СТУДЕНТСКА УНИЈА МАТЕМАТИЧАРА

Финално такмичење
А категорија
HS Algorithm 2024

шифра

Број бодова по задацима

1.		6.	
2.		7.	
3.		8.	
4.		9.	
5.		10.	

Укупан број поена:

СТУДЕНТСКА УНИЈА МАТЕМАТИЧАРА

Финално такмичење
HS Algorithm 2024

Упутство за израду задатака

Драги такмичари,

Испред вас се налазе задаци за финални део такмичења HS Algorithm. Тест се састоји од 10 задатака при чему је 5 задатака из математике, док је 5 задатака из информатике. У сваком задатку имаћете прилике да се упознате са темама које се изучавају на нашем факултету у оквиру основних академских студија на програмима Математика и Информатика.

Последњи задатак из сваке области је класичног типа - задатак решавате на папиру који касније предајете и **потребно је приказати детаљан поступак** да бисте добили све поене. У неким задацима ће заокруживање нетачног одговора на питање типа вишеструког избора донети негативне поене и то четвртину поена који су предвиђени за тај подзадатак (погледати табелу). Уколико питање оставите незаокружено, то се вреднује са 0 поена. Целокупан рад који на крају предајете мора бити исписан **хемијском оловком**, док слике можете цртати графитном оловком.

р.бр	**област**	**број поена**	**негативни поени**
1	Аналитичка геометрија	9	нема
2	Вероватноћа	9	има
3	Алгебра	9	нема
4	Анализа и линеарна алгебра	9	нема
5	Математичка анализа	14	нема
6	Дигитални запис података	9	има
7	Машинско учење	9	има
8	Алгоритми и структуре података	9	нема
9	Лексичка анализа	9	нема
10	Вештачка интелигенција	14	нема

Време за израду задатака је 180 минута. Такмичарска комисија ће два пута, након 60 и након 120 минута, проћи кроз све учионице и одговарати на ваша питања у вези поставки задатака.

СРЕЋНО!

СТУДЕНТСКА УНИЈА МАТЕМАТИЧАРА

Финално такмичење
А категорија
HS Algorithm 2024

шифра

Задатак 1. Аналитичка геометрија

Једна од важних грана аналитичке геометрије јесте **сферна геометрија**, која проучава облике на сфери. Док је класична геометрија, попут Еуклидове, заснована на равним линијама и равним површинама, сферна геометрија се бави фигурама на површини сфере.
Најзначајнија разлика је у томе што у сферној геометрији најкраћи пут између две тачке није права, већ најкраћа крива на сфери која повезује те две тачке. Многа тврђења из класичне геометрије не важе у сферној геометрији - на пример, косинусна теорема за троугао.
Сферна геометрија има значајну примену у астрономији и географији, јер се планета Земља може моделовати као сфера. Ова област геометрије омогућава да апроксимирамо дужине рута авионских летова. У астрономији, израчунавање растојања између звезда и њихових кретања по небу заснива се на примени сферне геометрије.

Уколико посматрамо раван и сферу, постоји неколико могућности за њихов међусобни положај. У крајње неинтересантном случају раван ће промашити сферу. У супротном, раван и сфера имају заједничких тачака, где је прва могућност да се ради о само једној заједничкој тачки и тада је у питању тангентна раван на сферу. Преостаје могућност када се раван и сфера секу по кругу. Лако је приметити да је пресечни круг највећи када раван пролази кроз центар сфере. Такав круг са сфере, чији се центар поклапа са центром сфере, зовемо **велики круг**. Географски примери су меридијани који су половине великог круга, док су све паралеле мали кругови, осим екватора који јесте велики круг. Велики кругови добијају на значају када схватимо да се најкраће растојање између две тачке на сфери реализује баш по луку великог круга који их спаја.

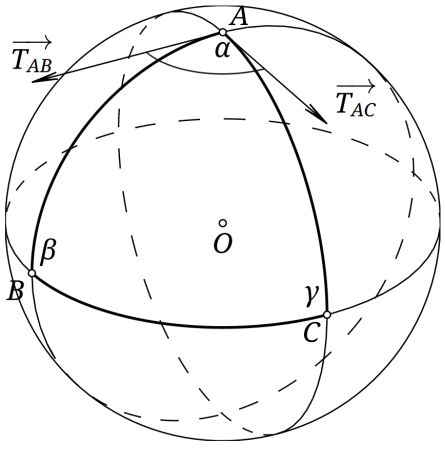
Посматрамо најједноставнију сферу у тродимензионом простору, сферу полупречника 1, са центром у координатном почетку:

$$\mathbb{S}^2 = \{(x_1, x_2, x_3) \ : \ x_1^2 + x_2^2 + x_3^2 = 1\}.$$

Сваки тачку са сфере $X \in \mathbb{S}^2$ можемо поистоветити са њеним вектором положаја у односу на центар сфере и уједно координатни почетак O, што је јединични вектор $\overrightarrow{OX}$. **Сферни троугао** ABC представљају тачке $A, B, C \in \mathbb{S}^2$, при чему захтевамо да су њихови вектори положаја, $\overrightarrow{OA}, \overrightarrow{OB}$ и $\overrightarrow{OC}$ линеарно независни.
Разлог због којег уводимо овај додатни услов линеарне независности је да избегнемо двосмислености. Наиме, уколико су вектори $\overrightarrow{OA}, \overrightarrow{OB}$ и $\overrightarrow{OC}$ линеарно зависни, то су тачке O, A, B, C копланарне (припадају истој равни), одакле следи да A, B, C припадају неком великом кругу. Аналогија са раванском геометријом је та да не можемо говорити о троуглу ABC уколико су тачке колинеарне.

Странице сферног троугла дефинишемо као краће лукове великих кругова који спајају темена. Дужине страница можемо обележити са a, b и c.
Угао између страница сферног троугла је угао између одговарајућих тангенти у теменима:

$$\alpha = \sphericalangle(\overrightarrow{T_{AB}}, \overrightarrow{T_{AC}}), \quad \beta = \sphericalangle(\overrightarrow{T_{BA}}, \overrightarrow{T_{BC}}), \quad \gamma = \sphericalangle(\overrightarrow{T_{CA}}, \overrightarrow{T_{CB}}).$$



Испоставља се да за сферни троугао важи

$$\cos a = \cos b \cos c + \sin b \sin c \cos \alpha,$$

што је пандан косинусној теореми еуклидске геометрије. Пандан синусне теореме је:

$$\frac{\sin a}{\sin \alpha} = \frac{\sin b}{\sin \beta} = \frac{\sin c}{\sin \gamma}.$$

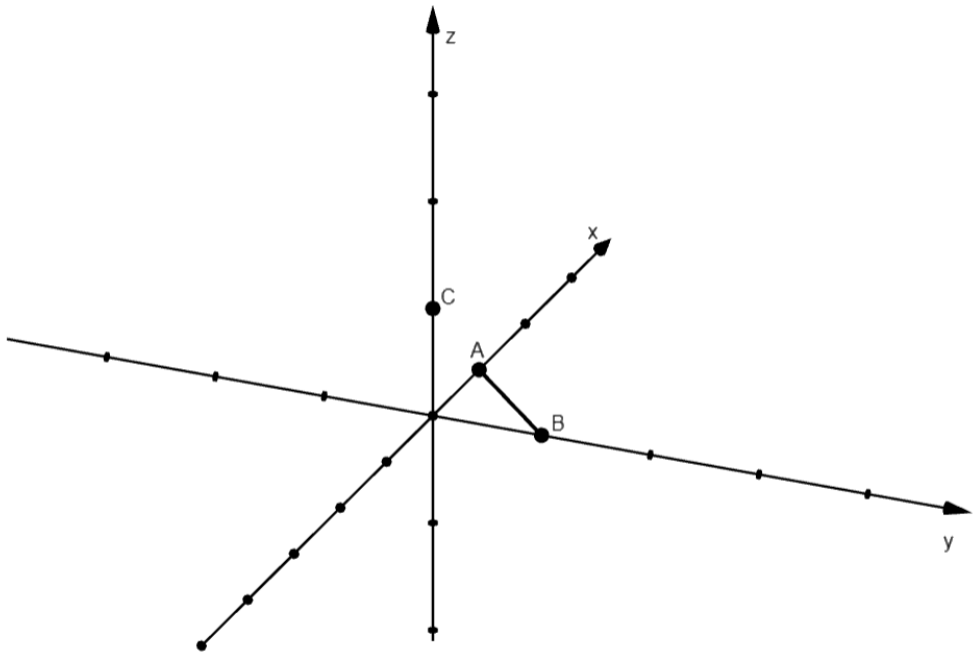
Постоји и дуална косинусна теорема, која даје однос између углова:

$$\cos \alpha = -\cos \beta \cos \gamma + \sin \beta \sin \gamma \cos a,$$
$$\cos \beta = -\cos \gamma \cos \alpha + \sin \gamma \sin \alpha \cos b,$$
$$\cos \gamma = -\cos \alpha \cos \beta + \sin \alpha \sin \beta \cos c.$$

Обим троугао се дефинише као збир свих страница $O = a + b + c$, док се може доказати да је површина сферног троугла $P = \alpha + \beta + \gamma - \pi$.

Потребно је да коначна решења упишите у за то предвиђене правоугаонике. Поступак се не оцењује. Негативне поене на овом задатку не можете да добијете.

(а) [**4 поена**] Дат је круг полупречника r_1 који садржи тачке $A(1,0,0)$, $B(0,1,0)$, $C(0,0,1)$ (видети слику испод). Означимо са d_1 растојање између тачака A и B **мерено по том кругу**. Означимо са d_2 дужину странице која спаја тачке A и B сферног троугла ABC (тј. растојање тачака A и B **мерено по јединичној сфери са центром у координатном почетку** - тачки са координатама $(0,0,0)$). Тада је $d_2 - d_1$ једнако:



(б) [**5 поена**] Сферни троугао ABC који се налази на јединичној сфери има углове $\alpha = \frac{\pi}{3}$, $\beta = \frac{2\pi}{3}$ и површину $P = \frac{\pi}{6}$. Његов обим је:

СТУДЕНТСКА УНИЈА МАТЕМАТИЧАРА

Финално такмичење
А категорија
HS Algorithm 2024

шифра

Задатак 2. Увод у вероватноћу

Вероватноћа је грана математике која се бави проучавањем случајних појава и њихових исхода. Она омогућава квантификацију несигурности и помаже у доношењу одлука у ситуацијама где исходи нису унапред познати. Основни задатак истраживача у области вероватноће је да идентификује основне карактеристике случајног експеримента, дефинише одговарајући простор вероватноће и развије моделе који омогућавају анализу и предвиђање будућих исхода. Вероватноћа има широку примену у свакодневном животу и различитим областима науке. На пример, користи се у играма на срећу за предвиђање шансе за добитак, у спортским статистикама за анализу учинка играча, и у метеорологији за процену вероватноће временских појава попут кише. Такође, вероватноћа игра кључну улогу у алгоритмима друштвених мрежа и видео игара, где помаже у генерисању насумичних догађаја или персонализованих препорука.

Пример. Замислимо торту облика квадрата на столу који је постављен у координатни систем. Доње лево поље има координате $(0,0)$, а горње десно $(1,1)$. Торта је подељена на следећи начин:

- Део изнад линије $y = \frac{2}{3}$ премазан је чоколадним преливом.
- Део испод линије $y = \frac{2}{3}$ премазан је преливом од јагода.
- Део за који важи $x + y > \frac{4}{5}$ има пуњење од вишања, док део за који важи $x + y < \frac{4}{5}$ има пуњење од малина.

Ако насумично убодемо торту чачкалицом, свака тачка на површини торте има једнаку шансу да буде убодена. Занима нас вероватноћа да убодемо део торте који је премазан чоколадом и има пуњење од вишања.

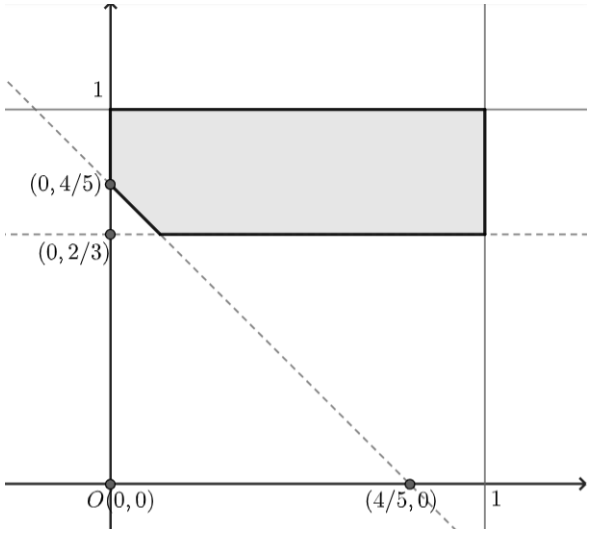
Решење. Вероватноћа да убодемо део торте који је премазан чоколадним преливом и пуњењем од вишања је дата као површина пресека ових области:

$$P(\text{чоколадни прелив и вишње}) = P(\text{чоколадни прелив} \cap \text{вишње}).$$

Ова вероватноћа се рачуна као површина пресечне области подељена са укупном површином торте.

Површина торте је 1. Део торте који је прекривен чоколадом налази се изнад линије $y = \frac{2}{3}$. Површина овог дела може се израчунати као правоугаоник дужине 1 и ширине $1 - \frac{2}{3}$, што даје површину $\frac{1}{3}$. С друге стране, део торте са вишњама налази се изнад линије $x + y = \frac{4}{5}$ и формира троугао. Површина овог троугла се израчунава формулом $\frac{1}{2} \cdot \text{страница} \cdot \text{висина}$, где су дужине страница и висина једнаке $\frac{4}{5}$, те је површина тог дела $\frac{1}{2} \cdot \frac{4}{5} \cdot \frac{4}{5} = \frac{8}{25}$. Коначно, површина пресека између чоколадног дела и дела са вишњама налази се унутар троугла дефинисаног пресеком линија $y = \frac{2}{3}$ и $x + y = \frac{4}{5}$. Овај пресек представља део површине где су оба услова истовремено задовољена. Коначо добијамо да је тражена вероватноћа:

$$P(\text{чоколадни прелив и вишње}) = \frac{1}{3} - \frac{1}{2} \cdot \frac{2}{15} \cdot \frac{2}{15} = \frac{73}{225}$$



Основни појмови у вероватноћи

Простор догађаја

Простор догађаја (Ω) је скуп свих могућих исхода неког случајног експеримента. Сваки елемент овог скупа представља један могући исход.

Пример: Код бацања шестостране коцкице, простор догађаја је $\Omega = \{1, 2, 3, 4, 5, 6\}$.

Случајни догађај

Случајни догађај је подскуп простора догађаја, који обухвата одређене исходе. На пример, ако размотрамо бацање коцкице:

- A: „Коцкица показује паран број" је случајни догађај $A = \{2, 4, 6\}$.

Ако је догађај једнак целом простору догађаја ($A = \Omega$), то је **сигуран догађај**. Ако је празан скуп ($A = \emptyset$), то је **немогућ догађај**.

Вероватноћа

Вероватноћа је функција која сваком догађају A придружује број из интервала $[0, 1]$. Функција P мора задовољити следеће особине:

- **Ненегативност:** За сваки догађај A важи $0 \leqslant P(A) \leqslant 1$.
- **Вероватноћа сигурног догађаја:** Вероватноћа да се догоди сигуран догађај је $P(\Omega) = 1$.
- **Вероватноћа немогућег догађаја:** Вероватноћа немогућег догађаја је: $P(\emptyset) = 0$.
- **Адитивност за дисјунктне догађаје:** За коначан број дисјунктних догађаја $A_1, A_2, \ldots, A_n$ важи

$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} P(A_i).$$

Израчунавање вероватноће помоћу површине

Ако се догађаји посматрају у неком геометријском простору, вероватноћа да случајно изабрана тачка припада некој области A одређује се као однос површине те области према површини целог простора.

$$P(A) = \frac{\text{Површина области } A}{\text{Површина целог простора}}.$$

Условна вероватноћа

Условна вероватноћа $P(A|B)$ представља вероватноћу да се догоди догађај A, под условом да се догодио догађај B. Дефинише се формулом:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad \text{ако је } P(B) > 0.$$

Принцип укључења и искључења

За коначан број догађаја $A_1, A_2, \ldots, A_n$, вероватноћа уније ових догађаја $P(A_1 \cup A_2 \cup \cdots \cup A_n)$ може се записати на следећи начин:

$$P(A_1 \cup A_2 \cup \cdots \cup A_n) = \sum_{i=1}^{n} P(A_i) - \sum_{1 \leqslant i < j \leqslant n} P(A_i \cap A_j) + \sum_{1 \leqslant i < j < k \leqslant n} P(A_i \cap A_j \cap A_k) - \cdots + (-1)^{n+1} P(A_1 \cap A_2 \cap \cdots \cap A_n).$$

Формула потпуне вероватноће

Ако је простор догађаја подељен на дисјунктне догађаје $H_1, H_2, \ldots, H_n$ који покривају цео простор ($H_1 \cup H_2 \cup \cdots \cup H_n = \Omega$), тада вероватноћа догађаја A може да се изрази као:

$$P(A) = \sum_{i=1}^{n} P(H_i) \cdot P(A|H_i).$$

Пример

Замислимо експеримент: Бацамо шестострану коцкицу. Израчунајмо вероватноћу догађаја: „Број је паран и мањи од 5".

Решење. Означимо A: „Испао је паран број" ($A = \{2, 4, 6\}$), а са B: „Испао је број мањи или једнак 3" ($B = \{1, 2, 3\}$). Јасно, вероватноће тих догађаја су $P(A) = \frac{3}{6} = 0.5$ и $P(B) = \frac{3}{6} = 0.5$.
Приметимо сада, вероватноћа да је испао паран број и да је тај број мањи или једнак 3) је $P(A \cap B) = \frac{1}{6}$ (пошто је $A \cap B = \{2\}$). Слично, вероватноћа да је испао паран број или број који је мањи или једнак 3 се може израчунати принципом укључења и искључења (а може и на друге начине, али овде ради илустрације приказујемо ово решење), чиме добијамо

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) = 0.5 + 0.5 - \frac{1}{6} = \frac{6}{6} - \frac{1}{6} = \frac{5}{6}.$$

Сада, условна вероватноћа $P(A|B)$ се рачуна на следећи начин: ако знамо да је број мањи или једнак 3 (B), тражимо вероватноћу да је број паран (A), што даје $P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{\frac{1}{6}}{\frac{3}{6}} = \frac{1}{3}$.
Коначно, израчунајмо вероватноћу догађаја C: „Број је паран и мањи од 5". Овај догађај можемо изразити помоћу партиције простора дефинисане догађајима H_1 : „Број је мањи или једнак 3" ($H_1 = \{1, 2, 3\}$) и H_2: „Број је већи од 3" ($H_2 = \{4, 5, 6\}$).
Применом формулу потпуне вероватноће добијамо $P(C) = P(C|H_1) \cdot P(H_1) + P(C|H_2) \cdot P(H_2)$. Вероватноћа $P(H_1)$, да је број мањи или једнак 3, је $P(H_1) = \frac{3}{6} = 0.5$, док је $P(H_2) = \frac{3}{6} = 0.5$, јер оба скупа имају по три елемента. Вероватноћа $P(C|H_1)$, да је број паран у H_1, је $P(C|H_1) = P(\{2\}) = \frac{1}{3}$, а $P(C|H_2)$, да је број паран у H_2, је $P(C|H_2) = P(\{4, 6\}) = \frac{2}{3}$. Заменом у формулу потпуне вероватноће добијамо

$$P(C) = \frac{1}{3} \cdot 0.5 + \frac{2}{3} \cdot 0.5 = \frac{1}{6} + \frac{2}{6} = \frac{3}{6} = 0.5.$$

Особине вероватноће за min **и** max

1. Вероватноћа за максимум:

$$P\{\max(A, B, C) < k\} = P(A < k \cap B < k \cap C < k),$$

што представља вероватноћу да су све величине мање од k. Овај израз одговара пресеку догађаја где су сви елементи испод границе k.

2. Вероватноћа за минимум:

$$P\{\min(A, B, C) > k\} = P(A > k \cap B > k \cap C > k),$$

што представља вероватноћу да су све величине изнад k. Овај израз одговара пресеку догађаја где су сви елементи изнад границе k.

Проблем са сегментима

Имамо штап дужине 1 који се насумично ломи на два места. Тиме добијамо три мања штапа.

Основни појмови

Да бисмо решили следеће задатке, прво ћемо објаснити основне појмове и принципе:

1. **Насумични ломови:**
   Штап се ломи на две случајне тачке X и Y унутар интервала $(0, 1)$. Ломови деле штап на три сегмента.

2. **Сегменти:**
   Дужине сегмената дефинишу се као:

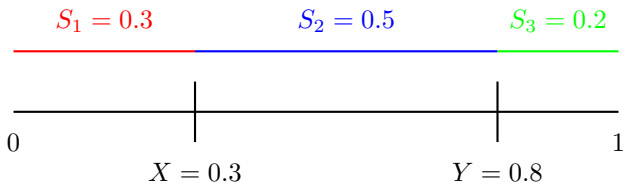
$$S_1 = \min(X, Y), \quad S_2 = |X - Y|, \quad S_3 = 1 - \max(X, Y).$$

3. **Уређени сегменти:**
   Након што израчунамо сегменте S_1, S_2, S_3, уређујемо их по величини неопадајуће:

$$L_1 = \min(S_1, S_2, S_3), \quad L_2 = \text{друга по величини од } S_1, S_2, S_3, \quad L_3 = \max(S_1, S_2, S_3).$$

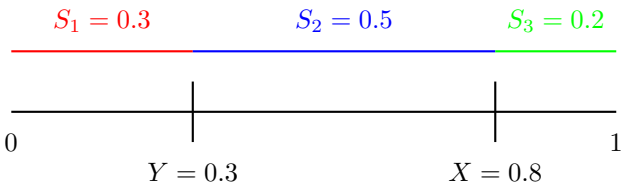
Случај 1: $X < Y$

Ако су тачке ломова $X = 0.3$ и $Y = 0.8$, сегменти су: $S_1 = 0.3$, $S_2 = 0.8 - 0.3 = 0.5$ и $S_3 = 1 - 0.8 = 0.2$.
Сортирано: $L_1 = 0.2$, $L_2 = 0.3$ и $L_3 = 0.5$.



Случај 2: $X > Y$

Ако су тачке ломова $X = 0.8$ и $Y = 0.3$, сегменти су: $S_1 = 0.3$, $S_2 = 0.8 - 0.3 = 0.5$ и $S_3 = 1 - 0.8 = 0.2$.
Сортирано: $L_1 = 0.2$, $L_2 = 0.3$ и $L_3 = 0.5$.



Неједнакост троугла

Да би се троугао могао формирати од три сегмента дужина L_1, L_2 и L_3 мора важити

$$L_1 + L_2 > L_3,$$

где су сегменти сортирани неопадајуће (тј. важи $L_1 \leqslant L_2 \leqslant L_3$).

(а) [**4 поена**] Колика је вероватноћа да се од три сегмента може формирати троугао?

а) $\frac{1}{2}$ б) $\frac{1}{4}$ в) $\frac{3}{4}$ г) 1 д) $\frac{1}{8}$

(б) [**4 поена**] Колика је вероватноћа да је највећи сегмент краћи од 0.6?

а) $\frac{7}{25}$ б) $\frac{13}{25}$ в) $\frac{17}{50}$ г) $\frac{6}{25}$ д) $\frac{13}{50}$

(в) [**1 поен**] Колика је вероватноћа да се од три сегмента може формирати троугао или да је највећи сегмент краћи од 0.6?

а) $\frac{13}{25}$ б) $\frac{77}{100}$ в) $\frac{1}{4}$ г) $\frac{53}{100}$ д) 1

СТУДЕНТСКА УНИЈА МАТЕМАТИЧАРА

Финално такмичење
А категорија
HS Algorithm 2024

шифра

Задатак 3. Алгебра

Алгебра је једна од основних грана математике која проучава структуре као што су групе, прстенови и поља, заједно са њиховим операцијама и релацијама. Групе описују симетрије и трансформације, прстенови се баве својствима операција сабирања и множења, док поља проширују ове концепте са делилачким структурама. Алгебра налази примену у многим областима, као што су криптографија, теорија бројева, информатика и физика, где помаже у моделирању и решавању сложених проблема. Њени концепти су темељни у модерној математици и науци.

Дефиниција 1. Група је алгебарска структура $(G, \cdot)$, где је G непразан скуп, а $\cdot$ бинарна операција на G која има следеће особине:

- **Ассоцијативност:** За све $x, y, z \in G$ важи $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.
- **Постојање неутралног елемента:** Постоји $e \in G$ такво да за све $x \in G$ важи $x \cdot e = e \cdot x = x$.
- **Постојање инверзног елемента:** За свако $x \in G$ постоји $x^{-1} \in G$ такво да је $x \cdot x^{-1} = x^{-1} \cdot x = e$.

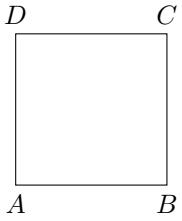
Дефиниција 2. Нека је G група. Подскуп $H \subseteq G$ се назива подгрупом од G ако важи:

- за све h_1 и h_2 из H, елемент $h_1 h_2$ се такође налази у H.
- за свако $h \in H$, и инверз h^{-1} се налази у H.

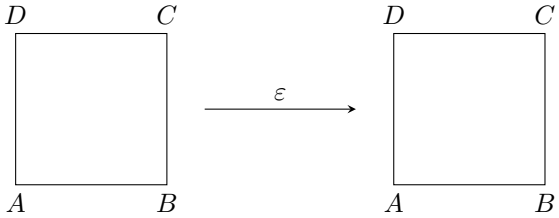
Дефиниција 3. Нека је G група. За подскуп $S \subseteq G$ кажемо да генерише групу G и пишемо $G = \langle S \rangle$ ако се сваки елемент из G може записати као коначан производ елемената из S и њихових инверза.

Дефиниција 4. Диедарска група D_n је група симетрија правилног n-многоугла ($n \geq 3$) у односу на композицију пресликавања.

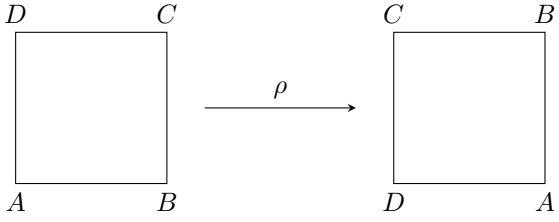
За било какав озбиљнији алгебарски рад са групом D_n биће потребно да боље разумемо и опишемо њене елементе. Посматраћемо квадрат $ABCD$ као на слици:



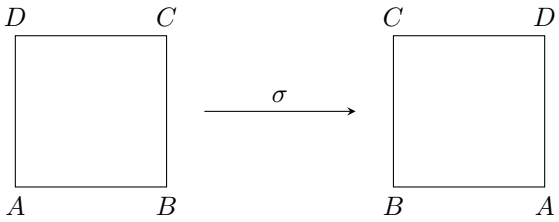
Симетрије овог квадрата могу бити ротације и рефлексије. При томе, међу ротације ћемо урачунати и идентичко пресликавање ε, које је неутрални елемент у групи D_4.



Са ρ ћемо означити ротацију за $\frac{\pi}{2}$ око центра квадрата у смеру супротном од кретања казаљке на сату.



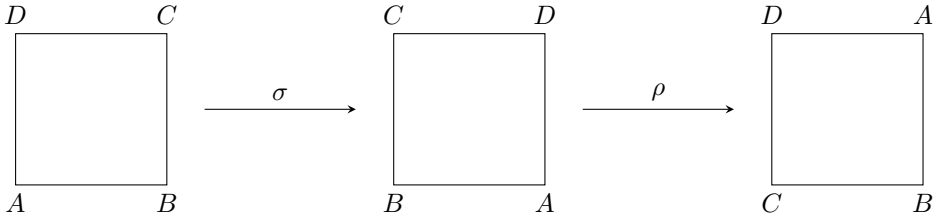
Остале ротације онда можемо изразити као ρ^2 и ρ^3, при чему је $\rho^4 = \varepsilon$.
Са σ ћемо означити рефлексију квадрата у односу на његову вертикалну осу.

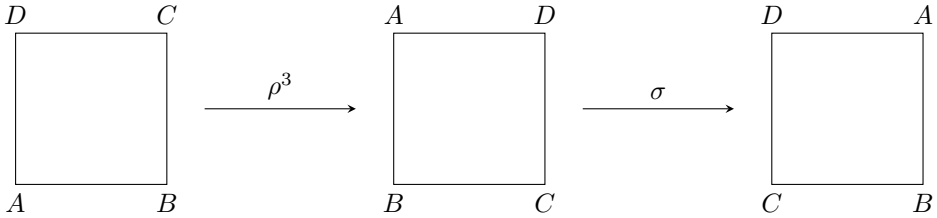


Имајући у виду геометријску интерпретацију, јасно је да је $\sigma^2 = \varepsilon$. Остале рефлексије можемо и добити тако што прво заротирамо квадрат, па су оне: $\sigma\rho$, $\sigma\rho^2$ и $\sigma\rho^3$.
Добијамо да је

$$D_4 = \{\underbrace{\varepsilon, \rho, \rho^2, \rho^3}_{\text{ротације}}, \underbrace{\sigma, \sigma\rho, \sigma\rho^2, \sigma\rho^3}_{\text{рефлексије}}\}.$$

Можемо приметити да је D_4 генерисана са ρ и σ, при чему знамо да је $\rho^4 = \varepsilon$ и $\sigma^2 = \varepsilon$. Што се тиче односа та два генератора, имамо:





Можемо закључити да је $\rho\sigma = \sigma\rho^3$, односно $\rho\sigma = \sigma\rho^{-1}$. На тај начин добијамо презентацију групе D_4 преко генератора и релација:

$$D_4 = \langle \underbrace{\rho, \sigma}_{\text{генератори}} \mid \underbrace{\rho^4 = \varepsilon,\ \sigma^2 = \varepsilon,\ \rho\sigma = \sigma\rho^{-1}}_{\text{релације}} \rangle.$$

Потпуно слично бисмо за свако $n \geq 3$ добили:

$$D_n = \langle \rho, \sigma \mid \rho^n = \varepsilon,\ \sigma^2 = \varepsilon,\ \rho\sigma = \sigma\rho^{-1} \rangle.$$

Дефиниција 5. Дејство групе G на скуп S је пресликавање $\cdot : G \times S \to S$ са особинама:

- $e \cdot x = x$ за све $x \in S$, где је e неутрални елемент групе G.
- $g \cdot (h \cdot x) = (gh) \cdot x$ за све $g, h \in G$ и $x \in S$.

Уз дејство групе G на скуп S дефинишемо следеће скупове:

- $\mathcal{O}_x = \{g \cdot x \mid g \in G\}$, *орбита елемента* $x \in S$.
- $\mathrm{Stab}(x) = \{g \in G \mid g \cdot x = x\}$, *стабилизатор елемента* $x \in S$.
- $\mathrm{Fix}(g) = \{x \in S \mid g \cdot x = x\}$, *фиксни скуп елемента* $g \in G$.

Подзадатак 1 [1 + 1 **поен**] Нека је $G = \left\{ \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix} \;\middle|\; a, b \in \mathbb{R}, a \neq 0 \right\}$ подгрупа групе $GL_2(\mathbb{R})$.

Са

$$\begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix} \cdot x = ax + b, \quad \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix} \in G,\ x \in \mathbb{R}$$

је дефинисано дејство групе G на $\mathbb{R}$.
Одредити стабилизатор за елемент $2 \in \mathbb{R}$, као и фиксни скуп за

$$\begin{bmatrix} -2 & 3 \\ 0 & 1 \end{bmatrix} \in G$$

при овом дејству.

Одговор: _____

Одговор: _____

Скуп свих орбита означавамо са S/G.

Тврђење (Бернсајдова лема) Нека је G коначна група која дејствује на коначан скуп S. Тада важи:

$$|S/G| = \frac{1}{|G|} \sum_{g \in G} |\mathrm{Fix}(g)|.$$

Сада када смо се упознали са теоријом можемо уз помоћ **дејства групе** и **Бернсајдове леме** много лакше решити неке занимљиве проблеме.

Пример. Одредити на колико начина се могу обојити темена правилног шестоугла са n боја ако су два бојења иста када се једно од другог може добити:

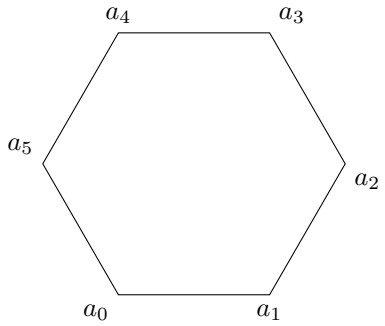
а) ротацијом шестоугла,

б) произвољном симетријом шестоугла.

Решење. а) Нека је S скуп свих могућих бојења темена шестоугла. Тада можемо записати:

$$S = \{(a_0, a_1, a_2, a_3, a_4, a_5) \mid 1 \leq a_i \leq n\},$$

при чему $a_i = k$ значи да је i-то теме обојено k-том бојом. Приметимо да је $|S| = n^6$.
Ситуацију можемо представити и шестоуглом на слици:



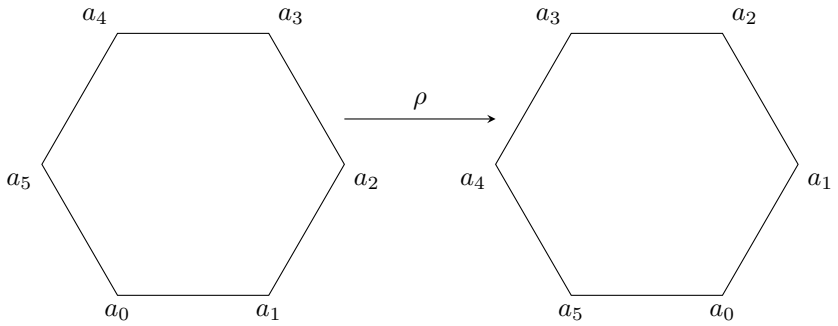
Подгрупа свих ротација $\langle \rho \rangle$ групе D_6 делује на скуп S ротирајући темена шестоугла. При томе су два бојења иста ако и само ако се налазе у истој орбити при овом дејству. Дакле, треба да израчунамо број орбита $|S/\langle \rho \rangle|$. Из Бернсајдове леме је:

$$|S/\langle \rho \rangle| = \frac{1}{|\langle \rho \rangle|} \sum_{g \in \langle \rho \rangle} |\mathrm{Fix}(g)| = \frac{1}{6} \sum_{i=0}^{5} |\mathrm{Fix}(\rho^i)|.$$

Све што је још потребно да урадимо је да израчунамо колико се елемената налази у фиксним скуповима које смо добили.

За $i = 0$ имамо да је $\rho^i = \varepsilon$, па сви елементи скупа S остају фиксирани. Дакле, $|\mathrm{Fix}(\varepsilon)| = n^6$.

За $i = 1$ имамо елемент ρ који представља ротацију за 60° око центра шестоугла у смеру супротном од смера кретања казаљке на сату. Његово дејство можемо представити сликом испод.Бојења у фиксном скупу су она која при овом дејству остају инваријантна.



То значи да прва и друга слика морају бити исте, одакле добијамо услове:

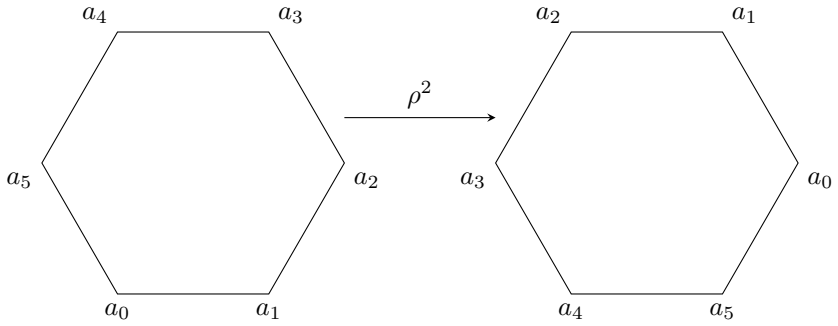
$$a_0 = a_5, \quad a_1 = a_0, \quad a_2 = a_1, \quad a_3 = a_2, \quad a_4 = a_3, \quad a_5 = a_4.$$

Закључујемо да важи:

$$a_0 = a_1 = a_2 = a_3 = a_4 = a_5 = \alpha, \quad 1 \leq \alpha \leq n,$$

па је: $|\mathrm{Fix}(\rho)| = n$.

Слично важи за $n = 2$, где имамо дејство елемента ρ^2, кога представљамо сликом:



Из једнакости прве и друге слике добијамо услове:

$$a_0 = a_4, \quad a_1 = a_5, \quad a_2 = a_0, \quad a_3 = a_1, \quad a_4 = a_2, \quad a_5 = a_3.$$

Закључујемо да је:

$$a_0 = a_2 = a_4 = \alpha, \quad a_1 = a_3 = a_5 = \beta, \quad 1 \leq \alpha, \beta \leq n,$$

па је: $|\mathrm{Fix}(\rho^2)| = n^2$.

Сличним разматрањем добијамо

$$|\mathrm{Fix}(\rho^3)| = n^3$$
$$|\mathrm{Fix}(\rho^4)| = n^2$$
$$|\mathrm{Fix}(\rho^5)| = n$$

Када све сумирамо, добијамо да је број тражених бојења једнак:

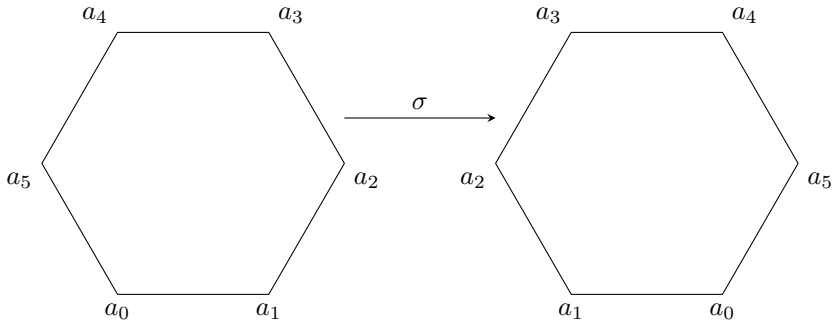
$$\frac{1}{6}\left(n^6 + n^3 + 2n^2 + 2n\right).$$

б) Све је исто као у делу **а)**, само што сада уместо дејства подгрупе $\langle\rho\rangle$, посматрамо дејство целе групе D_6. На основу Бернсајдове леме добијамо да је број тражених бојења једнак:

$$\frac{1}{|D_6|}\sum_{g\in D_6}|\mathrm{Fix}(g)| = \frac{1}{12}\sum_{g\in D_6}|\mathrm{Fix}(g)|.$$

Кардиналности фиксних скупова ротација смо већ одредили у делу **а)**, па остаје само још да их одредимо за рефлексије.

Рефлексија σ је у односу на вертикалну осу шестоугла, па је њено дејство представљено следећом сликом:

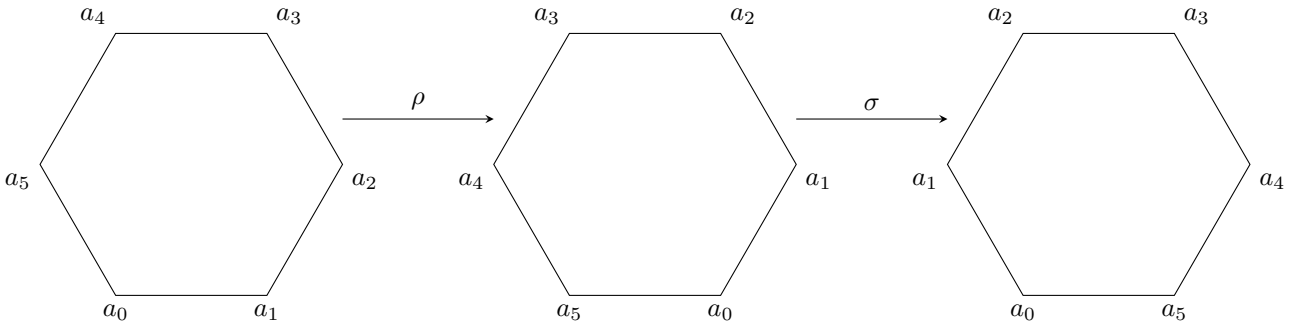


Из једнакости прве и друге слике закључујемо да важи:

$$a_0 = a_1 = \alpha, \quad a_2 = a_5 = \beta, \quad a_3 = a_4 = \gamma, \quad 1 \le \alpha, \beta, \gamma \le n,$$

па је: $|\mathrm{Fix}(\sigma)| = n^3$.

Дејство рефлексије $\sigma\rho$ представљено је следећом сликом:

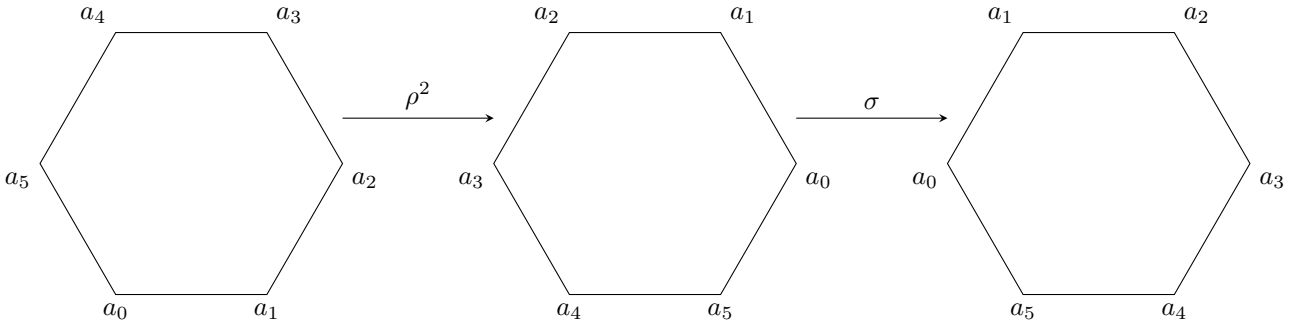


Из једнакости прве и треће слике закључујемо да важи:

$$a_0 = \alpha, \quad a_1 = a_5 = \beta, \quad a_2 = a_4 = \gamma, \quad a_3 = \delta, \quad 1 \le \alpha, \beta, \gamma, \delta \le n,$$

па је: $|\mathrm{Fix}(\sigma\rho)| = n^4$.

Дејство рефлексије $\sigma\rho^2$ представљено је следећом сликом:



Из једнакости прве и треће слике закључујемо да важи:

$$a_0 = a_5 = \alpha, \quad a_1 = a_4 = \beta, \quad a_2 = a_3 = \gamma, \quad 1 \le \alpha, \beta, \gamma \le n,$$

па је: $|\mathrm{Fix}(\sigma\rho^2)| = n^3$.
Сличним разматрањем добијамо

$$|\mathrm{Fix}(\sigma\rho^3)| = n^4$$
$$|\mathrm{Fix}(\sigma\rho^4)| = n^3$$
$$|\mathrm{Fix}(\sigma\rho^5)| = n^4$$

Када све сумирамо, добијамо да је број тражених бојења једнак:

$$\frac{1}{12}\left(n^6 + 3n^4 + 4n^3 + 2n^2 + 2n\right).$$

Подзадатак 2 [**3 поена**] Одредити на колико начина се могу обојити темена правилног петоугла са n боја, ако су два бојења иста када се једно од другог може добити ротацијом петоугла.

Подзадатак 3 [**4 поена**] Одредити на колико начина се могу обојити темена правилног осмоугла са n боја, ако су два бојења иста када се једно од другог може добити произвољном симетријом осмоугла.

СТУДЕНТСКА УНИЈА МАТЕМАТИЧАРА

Финално такмичење

А категорија

HS Algorithm 2024

шифра

Задатак 4. Линеарна алгебра и математичка анализа

Линеарна алгебра и математичка анализа представљају фундаменталне математичке области које се изучавају на свим математичким факултетима од самог почетка студија. Иако се веза између њих можда и не види одмах, заправо је веома јака и огледа се у својој пуној раскоши кроз теорију унитарних простора из које се надаље развијају теорија Фуријеових редова и Хилбертових простора. Фуријеови редови од саме појаве тог концепта налазе велику примену. Сам Фурије их је користио за решавање једначине топлоте, а данас се користе у обради и компресији сигнала, алгоритмима за препознавање говора и на бројним другим местима у индустрији. Вон Неуманн-овим математичким заснивањем квантне механике на темељу спектралне теорије оператора на Хилбертовим просторима, као и бурним развојем ове и сродних дисциплина, функционална анализа и теорија оператора постале су не само импозантне математичке теорије, већ и сам темељ савремене физике и једна од њених најмоћнијих апаратура.

Централни објекат за касније увођење појмова Фуријеовог реда и Хилбертових простора је скаларни производ са чијим сте се обичним обликом већ сусрели изучавајући аналитичку геометрију. У даљем тексту дефинисаћемо одређене појмове полуформално тежећи балансу између онога што вам је потребно да решите задатке у овом делу и прецизности како бисте имали свест о потпуној комплексности теорије којом се бавимо.

Дефиниција: Векторски простор V над пољем $\mathbb{R}$ (тренутно није битно шта значи да је $\mathbb{R}$ поље) је скуп вектора таквих да $\forall u, v \in V$ вреди $\alpha u + \beta v \in V$ за све реалне бројеве α, β.

За $U \subset V$ кажемо да је потпростор од V ако $\forall u_1, u_2 \in U$ вреди $\alpha_1 u_1 + \alpha_2 u_2 \in U$ за све реалне бројеве α_1, α_2. Пишемо $U \leqslant V$.

Пример: $\mathbb{R}^3$ посматран као скуп вектора у 3 димензије при чему сваки вектор сматрамо да има почетак у координатном почетку и да је одређен коринатама (x, y, z) краја тог вектора је векторски простор над пољем $\mathbb{R}$.

Дефиниција: Нека је X векторски простор над пољем $\mathbb{R}$. Функција $\|\cdot\| : X \to \mathrm{R}$ назива се норма ако испуњава:

- $\|x\| \geqslant 0$ и $\|x\| = 0$ ако и само ако $x = 0$
- $\|\lambda x\| = |\lambda| \|x\|$ за сваки реалан број λ
- $\|x + y\| \leqslant \|x\| + \|y\|$.

Тада се уређен пар $(X, \|\cdot\|)$ назива нормиран векторски простор. Функција норме замишљена је тако да имитира својства апсолутне вредности.

Дефиниција: Функција $\langle \cdot, \cdot \rangle : X \times X \to \mathbb{R}$ која испуњава својства:

- $\langle x, x \rangle \geqslant 0$ и $\langle x, x \rangle = 0$ ако и само ако $x = 0$
- $\langle x, y \rangle = \langle y, x \rangle$

- $\langle \lambda_1 x_1 + \lambda_2 x_2, y\rangle = \lambda_1 \langle x_1, y\rangle + \lambda_2 \langle x_2, y\rangle$ за све $\lambda_1, \lambda_2 \in \mathbb{R}$

назива се скаларни производ. У том случају се уређен пар $(X, \langle \cdot, \cdot \rangle)$ назива унитаран или пред-Хилбертов простор.

Тврђење: Сваки унитаран простор је нормиран, јер се норма може увести на природан начин са

$$\|x\| = \sqrt{\langle x, x\rangle}.$$

Нећемо се бавити доказивањем да овако дефинисана норма заиста задовољава аксиоме норме из једне од претходних дефиниција.

Примери:

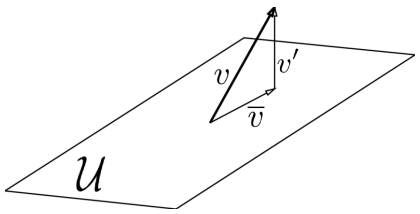
- Скаларни производ у $\mathbb{R}^3$ који сте учили у школи дефинисан са $\langle (x_1, x_2, x_3), (y_1, y_2, y_3)\rangle = x_1y_1 + x_2y_2 + x_3y_3$ заиста задовољава аксиоме скаларног производа из претходне дефиниције. Аналогно се дефинише скаларни производ у $\mathbb{R}^n$
- Није тешко показати да непрекидне функције на интервалу $[a, b]$ чине један векторски простор. На овом векторском простору можемо увести скаларни производ са:

$$\langle f, g\rangle = \int_a^b f(x)g(x)\mathrm{d}x$$

Дефиниција: За векторе $f, g \in X$ каже се да су ортогонални и то се означава са $f \perp g$, уколико је $\langle f, g\rangle = 0$. За скупове $F, G \subset \Theta$ каже се да су ортогонални, а означава са $F \perp G$, ако је $f \perp g$ за свако $f \in F$ и $g \in G$. Са $G^\perp$ означава се скуп свих $f \in H$ који су ортогонални на G и назива се ортогоналом скупа G.

Тврђење: $G^\perp \leqslant X$ тј. сваки ортогонал је потпростор од X односно затворен је у односу на линеарне комбинације.

Теорема о ортогоналној пројекцији: Ако је X векторски простор на коме је деифнисан скаларни производ (тј. унитарни простор) и U потпростор од X тада се сваки вектор $v \in X$ може представити у облику $v = \overline{v} + v'$ при чему је $\overline{v} \in U, v' \in U^\perp$. Вектор $\overline{v}$ назива се ортогонална пројекција вектора v на потпростор U, а растојање вектора v од потпростора U дефинише се као $d(v, U) = ||v'||$.



Потребно је да коначна решења упишите у за то предвиђене правоугаонике. Поступак се не оцењује. Негативне поене на овом задатку не можете да добијете.

а) [3 поена] Познато је да је пресликавање $\circ$: $\mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$:

$$(x_1, x_2, x_3) \circ (y_1, y_2, y_3) = 2x_1y_1 + 2x_2y_2 + 2x_3y_3 - x_1y_2 - x_2y_1 - x_2y_3 - x_3y_2$$

један скаларни производ на векторском простору $\mathbb{R}^3$. Ако је U скуп свих решења једначине $x - 2y + 3z = 0$ у скупу реалних бројева одредити растојање вектора $v = (0, 1, 4)$ од U.

Дефиниција: Скуп $\{e_1, e_2, \dots\}$ је ортонормирана база унитарног простора X ако се сваки вектор $v \in X$ може представити у облику $v = \alpha_1 e_1 + \alpha_2 e_2 + \dots + \alpha_n e_n + \dots$ и ако важи $e_i \perp e_j$ за $i \neq j$ и $||e_n|| = 1$ за свако n.

Пример: Посебно нам је од интереса одредити ортонормирану базу за простор непрекидних функција на $[-\pi, \pi]$ са скаларним производом дефинисаним са $\langle f, g\rangle = \int_a^b f(x)g(x)\mathrm{d}x$. Може се показати да функције $\{\frac{1}{\sqrt{2\pi}}, \frac{\cos x}{\sqrt{\pi}}, \frac{\sin x}{\sqrt{\pi}}, \frac{\cos 2x}{\sqrt{\pi}}, \frac{\sin 2x}{\sqrt{\pi}}, \dots, \frac{\cos nx}{\sqrt{\pi}}, \frac{\sin nx}{\sqrt{\pi}}, \dots\}$ чине ортонормирану базу за простор непрекидних функција на $[-\pi, \pi]$ са овим скаларним производом.

Заиста, за $m \neq n$ важи:

$$\langle \cos mx, \cos nx\rangle = \int_{-\pi}^{\pi} \cos mx \cos nx = \frac{1}{2}\left(\int_{-\pi}^{\pi} \cos(m+n)x dx + \int_{-\pi}^{\pi} \cos(m-n)x dx\right)$$
$$= \frac{1}{2}\left(\frac{\sin(m+n)x}{m+n}\Big|_{-\pi}^{\pi} - \frac{\sin(m-n)x}{m-n}\Big|_{-\pi}^{\pi}\right) = 0$$

За $m = n$ довољно је проверити

$$\|\cos mx\|^2 = \int_{-\pi}^{\pi} \cos^2 mx dx = \int_{-\pi}^{\pi} \frac{1+\cos 2mx}{2} dx = \frac{1}{2}\left(\int_{-\pi}^{\pi} 1 dx + \frac{\sin 2mx}{2m}\Big|_{-\pi}^{\pi}\right) = \pi \Rightarrow \|\cos mx\| = \sqrt{\pi}$$

Аналогно бисмо показали $\langle \sin mx, \sin nx\rangle = 0$ за $m \neq n$ и $||\sin mx|| = \sqrt{\pi}$.

Дефиниција: Фуријеов ред функције f дефинисане на интервалу $[-\pi, \pi]$ је

$$S(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n \cos nx + b_n \sin nx)$$

где су $\{a_0, a_1, \dots, a_n, \dots, b_1, b_2, \dots b_n, \dots\}$ Фуријеови коефицијенти дефинисани на следећи начин:

- $a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,dx$
- $a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nx\,dx$ за $n \geqslant 1$
- $b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin nx\,dx$ за $n \geqslant 1$

Теорема (Дирихле): Ако је непрекидна функција f део-по-део монотона (мења монотоност коначно много пута између тачака локалног максимума и минимума) тада важи $S(x) = f(x)$ за све $x \in (-\pi, \pi)$.

б) [3 поена] Израчунати Фуријеове коефицијенте $\{a_0, a_1, \dots, a_n, \dots, b_1, b_2, \dots b_n, \dots\}$ за функцију $f(x) = \cos\frac{x}{2}$ на интервалу $[-\pi, \pi]$. Напишите све коефицијенте које успете израчунати.

в) [3 поена] Израчунати суму

$$\sum_{n=1}^{\infty} \frac{(-1)^n}{4n^2 - 1}$$

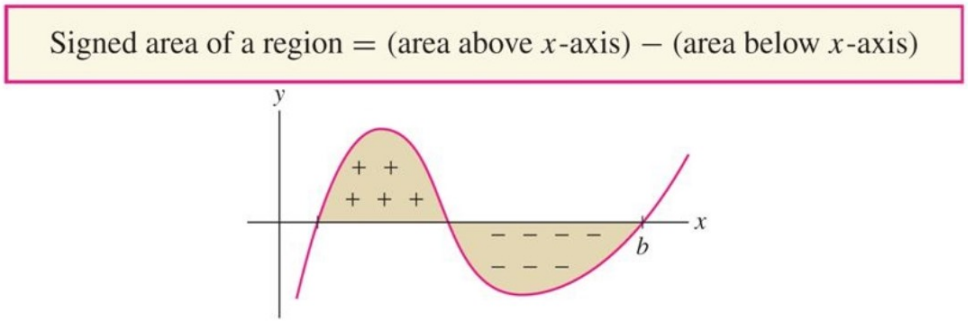
Немојте заокруживати бројеве на калкулатору!

За ученике који се раније нису сусрели са тим појмовима наводимо кратак увод у интеграле.

- **Извод функције** $F(x)$, у ознаци $F'(x)$, је математичка операција која представља брзину промене функције у одређеној тачки, тј. мери колико се функција $F(x)$ промени када се вредност независне променљиве x промени за врло малу вредност. Извод функције $F(x)$ представља проналажење нове функције $f(x)$ која описује брзину промене оригиналне функције у зависности од промене вредности променљиве x, тј. $f(x) = F'(x) = \frac{dF}{dx}$.

- **Интеграл функције** $f(x)$, у ознаци $\int f(x)\,dx$, је математичка операција која представља супротност изводу. Интеграл функције $f(x)$ представља проналажење нове функције $F(x)$, назване интегрална функција или **примитивна функција**, чији је извод једнак оригиналној функцији, тј. $F'(x) = f(x)$.

- **Одређени интеграл функције** $f(x)$, у ознаци $\int_a^b f(x)\mathrm{d}x$, представља означену површину испод графика функције $f(x)$ на интервалу $[a, b]$.



- **Њутн-Лајбницова формула** Ако је функција $f(x)$ непрекидна на интервалу $[a, b]$ тада је $\int_a^b f(x)\mathrm{d}x = F(x)|_a^b = F(b) - F(a)$ где је $F(x)$ функција таква да је $F'(x) = f(x)$.

Нека правила за рачунање извода и интеграла:

- $(\sin x)' = \cos x, (\cos x)' = -\sin x$
- $\int \sin x = -\cos x + C, \int \cos x = \sin x + C$

Нека правила која важе за изводе и интеграле:

Особине извода	Особине интеграла
$(Cf(x))' = Cf'(x)$	$\int Cf(x)dx = C\int f(x)dx$
$(f(x) \pm g(x))' = f'(x) \pm g'(x)$	$\int (f(x) \pm g(x))dx = \int f(x)dx \pm \int g(x)dx$
$(f(x) \cdot g(x))' = f'(x)g(x) + f(x)g'(x)$	$\int u(x)v'(x)dx = u(x)v(x) - \int v(x)u'(x)dx$
$\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$	$\int f'(g(x)) \cdot g'(x)dx = f(g(x)) + C$
$(f(g(x)))' = f'(g(x))g'(x)$	У претходном реду, функција $g(x)$ је смена и пишемо $g(x) = t$, $dt = g'(x)dx$ и то заменимо у интеграл

СТУДЕНТСКА УНИЈА МАТЕМАТИЧАРА

Финално такмичење

А категорија

HS Algorithm 2024

шифра

Задатак 5. Математичка анализа

Анализа је математичка област која се бави изучавањем граничних процеса и бесконачно малих величина. Вероватно сте чули за појмове као што су интеграл, извод или лимес (гранична вредност). Најбитнији појам у анализи јесте појам функције тј. пресликавања са којим сте се сусрели у средњошколском образовању. У анализи уобичајено посматрамо како се нека функција једне реалне промењиве понаша у „околини неке тачке a". Под „околином неке тачке a" мислимо на тачке које се налазе на неком (малом) растојању од тачке a. На пример, интервал $(1.95, 2.05)$ интуитивно можемо сматрати околином тачке 2. Међутим, поменули смо појам растојања једне тачке од друге тачке - у скупу реалних бројева $\mathbb{R}$ је природно посматрати растојање две тачке тј. два броја a и b као апсолутну вредност њихове разлике $|a-b|$ - растојање бројева 2 и 2.05 је $|2-2.05| = 0.05$. У скупу $\mathbb{R}^2$ растојање тачака $A(x,y)$ и $B(z,t)$ је $\sqrt{(x-z)^2+(y-t)^2}$.

Овај начин мерења растојања између тачака скупа $\mathbb{R}$ и $\mathbb{R}^2$ су нам интуитивно јасна и поприлично позната из свакодневног живота. Како у математици увек тежимо општости тј. универзалности, често покушавамо да меримо и растојање између неке две тачке (тј. нека два елемента) произвољног скупа X. Преношење интуиције о мерењу растојања у општијим ситуацијама може бити вишеструко корисно - многе озбиљније математичке теорије развијају се управо граде се на метричким просторима, метрика Лондонске железнице (енгл. taxicab metric) служи диспечерима да знају који такси да шаљу на адресу, али своје примене налази и у развијању хеуристика за вештачку интелигенцију, у обради слика користимо метрике да проценимо колико се разликују, у теорији релативности простор-време се моделује као метрички простор...

Шта су онда заправо формално метрика и метрички простори?

Посматрајмо особине растојања $\sqrt{(x-z)^2+(y-t)^2}$ између тачака $A(x,y)$ и $B(z,t)$ у $\mathbb{R}^2$. Важе следеће особине:

- растојање је ненегативно и једнако је 0 ако и само ако меримо растојање неке тачке од саме себе;
- растојање од A до B је једнако растојању од B до A;
- растојање од тачке A до B је мање него збир растојања тачака A и B од неке тачке $C(p,q)$ у $\mathbb{R}^2$.

Сада можемо на произвољном скупу X одредити услове да нека функција $d : X \times X \to \mathbb{R}$ буде растојање тј. метрика (d је од енглеског distance).

Дефиниција: Нека је X произвољан (непразан) скуп. Функција $d : X \times X \to \mathbb{R}$ је метрика на скупу X ако задовољава следећа својства:

1. За све $x, y \in X$ је $d(x,y) \geqslant 0$ и $d(x,y) = 0 \iff x = y$;
2. За све $x, y \in X$ је $d(x,y) = d(y,x)$;
3. За све $x, y, z \in X$ је $d(x,y) \leqslant d(x,z) + d(z,y)$.

Тада кажемо да је (X,d) један метрички простор.

Пример: *Дискретна метрика* на било ком скупу X дефинисана је са $g : X \times X \to \mathbb{R}$ са $g(x,y) = 1$ ако $x \neq y$ и $g(x,x) = 0$. Није тешко проверити да је (X, g) заиста метрички простор, тј. да овако дефинисано g задовољава сва 3 својства која треба испуњавати да би била метрика.

Подзадатак 1 [2 поена]: Нека је Y скуп свих бесконачних низова реалних бројева и нека су дата пресликавања

$$m : Y \times Y \to \mathbb{N}, \quad m(x,y) = \{\min i \mid x_i \neq y_i\},$$

$$d : Y \times Y \to \mathbb{R}, \quad d(x,y) = \begin{cases} \log\left(1 + \frac{1}{m(x,y)}\right), & x \neq y, \\ 0, & x = y \end{cases}.$$

Доказати да је са d задата метрика на скупу Y. Напомена: $\log x = \log_{10} x$ за $x > 0$.

Напомена: Сматрамо да низ x има елементе $(x_1, x_2, \dots)$, а низ y има елементе $(y_1, y_2, \dots)$.

У наредним подзадацима ћемо радити са овим метричким простором (Y, d).

Дефиниција: За низ тачака метричког простора $\{a_n\}_{n\in\mathbb{N}} \subseteq (X, d)$ кажемо да конвергира ако постоји $x \in X$ и за свако $\varepsilon > 0$ постоји $n_0 \in \mathbb{N}$ тако да за свако $n \geqslant n_0$ важи $d(a_n, x) \leqslant \varepsilon$. То означавамо са:

$$\lim_{n\to+\infty} a_n = x.$$

Подзадатак 2 [3 поена]: Да ли низ $(a_n)_{n\in\mathbb{N}}$ задат са

$$a_n = (\underbrace{1, 2, 3, \dots, n}_{n}, 0, 0, \dots)$$

конвергира у метричком простору (Y, d)? Уколико конвергира, одредити му лимес. Решење детаљно образложити.

Дефиниција: Нека је дат метрички простор (X, d). За тачку $x \in X$ и $\varepsilon > 0$ дефинишемо:

1. отворену куглу са центром у x полупречника ε:

$$B(x, \varepsilon) = \{y \in X \mid d(x,y) < \varepsilon\};$$

2. затворену куглу са центром у x полупречника ε:

$$B[x, \varepsilon] = \{y \in X \mid d(x,y) \leqslant \varepsilon\};$$

Дефиниција: Подскуп U метричког простора (X, d) је отворен ако за сваку тачку $x \in U$ постоји $\varepsilon_x > 0$ тако да је $B(x, \varepsilon_x) \subseteq U$, односно ако за сваки његов елемент постоји нека кугла са центром у том елементу која је цела садржана у U. Фамилију свих отворених скупова метричког простора (X, d) означаваћемо са $\mathcal{T}_d$.

Подскуп F метричког простора (X, d) је затворен ако му је комплемент отворен.

Напомена: У претходној дефиницији отвореног скупа писали смо ε_x како бисмо нагласили да оно може да зависи од тачке x тј. не мора да буде исто за различите тачке.

Тврђење: Није се тешко уверити да је произвољна унија отворених скупова отворен скуп.

Пример: Ако је X произвољан скуп и g дискретна метрика на X тада је $B(x, \frac{1}{2}) = \{y \in X \mid g(x,y) < \frac{1}{2}\} = \{x\}$ отворена кугла са центром у x полупречника $\frac{1}{2}$ јер је $g(x,y) = 1$ за $x \neq y$. Приметите да одавде следи да су једночлани скупови отворени у дискретној метрици.

Дефиниција: Две метрике d_1 и d_2 на истом скупу X су тополошки еквивалентне ($d_1 \sim_t d_2$) ако генеришу исте фамилије отворених скупова, односно ако важи:

$$U \in \mathcal{T}_{d_1} \Leftrightarrow U \in \mathcal{T}_{d_2}$$

Дефиниција: Две метрике d_1 и d_2 на истом скупу X су метрички еквивалентне ($d_1 \sim_m d_2$) ако постоје константе $\alpha, \beta > 0$ тако да за сваке две тачке $x, y \in X$ важи:

$$\alpha d_1(x, y) \leqslant d_2(x, y) \leqslant \beta d_1(x, y)$$

Тврђење. Ако су две метрике d_1 и d_2 на скупу X метрички еквивалентне тада да су и тополошки еквивалентне.

Подзадатак 3 [4 поена]: Испитати да ли је метрика d на метричком простору (Y, d) метрички и тополошки еквивалентна са дискретном метриком на скупу Y. Решење детаљно образложити.
Дефиниција: Метрички простор (X, d) је неповезан ако постоје подскупови $X_1, X_2 \subseteq X$ такви да важи:

- $X_1 \cup X_2 = X$
- $X_1 \cap X_2 = \emptyset$
- $X_1, X_2 \neq \emptyset$
- $X_1, X_2 \in \mathcal{T}_d$

тј. ако се цео простор може представити као дисјунктна унија непразних отворених скупова.
Простор је повезан ако није неповезан.
Подзадатак 4 [5 поена]: Да ли је метрички простор (Y, d) повезан? Решење детаљно образложити.

Задатак 6. Дигитални запис података

Приликом комуникације између два уређаја, који могу и не морају бити у оквиру истог рачунарског система, може доћи до непредвиђених грешака. Те грешке могу да се догоде или на самом преносном путу, или на локацијама где се налазе уређаји који учествују у комуникацији. Разлози за настанак грешака су разни, а најчешћи узроци су шумови (проузроковани електромагнетним сметњама), физичка оштећења преносног медијума, преоптерећеност мреже, и слично. Преносни сигнал, односно порука коју један уређај шаље другом, интерпретира се као низ нула и јединица, па се грешке приликом преноса манифестују кроз промену вредности једног или више битова.

Будући да је прецизан пренос података од суштинске важности, развијени су различити алгоритми за детекцију грешака. Неки од познатијих су контрола парности, контрола збира блока, циклична провера редунданси (CRC) и Хамингов SEC код. Последњи поменути алгоритам, Хамингов SEC код, за разлику од осталих, поред откривања постојања грешке, може и да изврши корекцију на биту на којем се појавила грешка.

У овом делу задатка бавићемо се управо тим алгоритмом. Алгоритам објашњавамо на примеру порука које се састоје од 12 битова, где првих 8 битова представља сам садржај поруке, док преостала 4 бита представљају контролне битове. Прималац, на основу добијене поруке, генерише своје контролне битове и упоређује их са контролним битовима које му је проследио пошиљалац. На основу тога ће закључити да ли је дошло до грешке на неком биту, и ако јесте, одрадити потребне корекције. Нека су битови поруке означени са $m_8, ..., m_1$, а контролни битови са $c_4, ..., c_1$, на следећи начин:

m_8	m_7	m_6	m_5	m_4	m_3	m_2	m_1	c_4	c_3	c_2	c_1

Сада је потребно формирати таблицу Хамингових SEC кодова. У првој колони су декадни бројеви од 12 до 1, у другој колони одговарајући бинарни еквиваленти, записани помоћу четири бита, а у трећој се сваком броју (од 12 до 1) додељује неки од битова $m_8, ..., m_1, c_4, ..., c_1$. Битови $c_4, ..., c_1$ се додељују оним бројевима који су степени броја два (тј. бројевима 1, 2, 4 и 8), а на преостала места се додају битови $m_8, ..., m_1$ на начин приказан у следећој табели:

12	1100	m_8
11	1011	m_7
10	1010	m_6
9	1001	m_5
8	1000	c_4
7	0111	m_4
6	0110	m_3
5	0101	m_2
4	0100	c_3
3	0011	m_1
2	0010	c_2
1	0001	c_1

Затим се израчунавају контролни битови $c'_4, ..., c'_1$ преко битова $m_8, ..., m_1$, тако што се редом у четвртој, трећој, другој и првој колони свих бинарних записа из табеле у одговарајућу формулу убаце они битови m_i чија је вредност 1 и изврши се њихова ексклузивна дисјункција. Другим речима, важи:

$$c'_4 = m_5 \oplus m_6 \oplus m_7 \oplus m_8$$

$$c'_3 = m_2 \oplus m_3 \oplus m_4 \oplus m_8$$

$$c'_2 = m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7$$

$$c'_1 = m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7$$

Коначно, израчунава се вредност $c_4c_3c_2c_1 \oplus c'_4c'_3c'_2c'_1$, чиме се добија четворобитна бинарна ниска $r_4r_3r_2r_1$. Уколико се овај број не налази у горе наведеној табели, или се пак налази, али се у одговарајућој врсти налази нека од вредности $c_4, ..., c_1$, није дошло до грешке. У супротном, добијена вредност упућује на ком од контролних битова $m_8, ..., m_1$ је дошло до грешке, те је ту вредност потребно инвертовати, како би се добио тачан садржај поруке.

(а) [**4 поена**] Прималац је добио три наредне дванаестобитне поруке:

1. 010011001110
2. 010001111100
3. 010011101010

Свака од примљених порука носи информацију о једном ASCII карактеру. Знајући да је ASCII код карактера 'A' 65, која трословна реч је састављена од карактера који су кодирани наведеним порукама?

а) LGN
б) LON
в) LOL
г) Ништа од наведеног

Размотримо сада поменути алгоритам цикличне провере редунданси (односно CRC методу), који, за разлику од Хамминговог SEC кода, не може да детектује позицију бита на ком се догодоила грешка, већ само да установи да је до исте дошло. Прималац сада, поред кодиране поруке, добија и *полином генератор*, на основу којих може или да установи да је дошло до грешке и да затражи поновно слање поруке, или да издвоји оригиналну поруку, ако се испостави да није дошло до грешке.

Полином генератор је произвољан полином чији су коефицијенти или 1, или 0. *Бинарни запис* полинома генератора $G(x) = a_nx^n + a_{n-1}x^{n-1} + ... + a_1x + a_0$ је бинарна ниска састављена од коефицијената полинома $G(x)$, тј. ниска $a_na_{n-1}...a_1a_0$. Имајући ове дефиниције у виду, поступак CRC методе можемо описати следећим корацима:

1. Пошиљалац бира произвољан полином генератор. Нека је степен изабраног полинома n.

2. Пошиљалац дописује n нула на крај поруке коју жели да пошаље (што представља множење поруке са x^n).

3. Извршава се дељење новодобијене поруке бинарним записом полинома генератора. Дељење бинарних бројева је лакше него дељење декадних, јер се уместо одузимања одговарајућих цифара врши њихова ексклузивна дисјункција, па нема ни позајмица. Водеће нуле које се добијају у међурезултатима се игноришу, а потом се спушта онолико цифара дељеника колико је неопходно да се добије број чија је дужина једнака дужини делиоца (под условом да постоји толико цифара у дељенику). Приликом овог дељења, од интереса је само остатак који се добија, не и количник. Пример оваквог дељења приказан је у наставку:

```
111001100000 / 11001
11001
  10111
  11001
   11100
   11001
     10100
     11001
      11010
      11001
         110
```

4. Порука која се шаље примаоцу добија се када се на оригиналну поруку надовеже остатак добијен претходним дељењем. Тај остатак је, евентуално, потребно допунити водећим нулама тако да буде записан на укупно n места.

5. Тако добијена порука се шаље примаоцу, заједно са полиномом генератором.

6. Прималац врши дељење примљене поруке одговарајућим бинарним записом добијеног полинома генератора, на исти начин који је описан у кораку 3.

7. Уколико је добијени остатак различит од 0, констатује се грешка у преносу и захтева се поновно слање поруке. У супротном, порука је исправно примљена, а оригинални садржај добија се одбацивањем последњих n битова примљене поруке.

(б) [**2.5 поена**] Прималац је добио поруку 110101101110. Који од наведених полинома може да буде полином генератор који је такође примљен, уколико је CRC алгоритам установио да је порука успешно испоручена?

а) $G(x) = x^3 + x^2$
б) $G(x) = x^5 + x + 1$
в) $G(x) = x^4 + x^2 + x$
г) ниједан од наведених полинома

(в) [**2.5 поена**] Пошиљалац жели да као садржај поруке пошаље осмобитни бинарни запис броја 215 (записаног у декадном бројном систему). За полином генератор изабрао је полином петог степена, код

кога су само два коефицијента једнака 1 и које је дељив полиномом $R(x) = x^2 - x + 1$. Облик за слање ове поруке је:

а) 11000

б) 1101011111000

в) 100100

г) ништа од наведеног

Задатак 7. Машинско учење

Кластеровање је метода машинског учења која се користи за груписање података у мање групе, познате као **кластери**. Подаци унутар истог кластера имају сличне особине и међусобно су ближи, док су у односу на податке из осталих кластера различитији и удаљенији. Ова техника представља **ненадгледану методу**, што значи да за податке унапред није познато којој групи припадају. Уместо тога, алгоритми кластеровања анализирају особине података и на основу њих одлучују којој групи (кластеру) припадају. Циљ је идентификовати природне групе и обрасце који постоје у подацима, чиме се омогућава боље разумевање сложених скупова података.

Алгоритми кластеровања су незаменљив алат у савременој науци и индустрији, захваљујући способности да групишу сличне податке и открију скривене обрасце. На пример, у биологији се користе за груписање организама на основу генетских сличности, док у друштвеним мрежама омогућавају идентификацију заједница корисника са заједничким интересовањима – замислите, на пример, груписање корисника који прате исте теме или инфлуенсере. Када се ради о сликама, кластеровање може да разврста фотографије према визуелним сличностима – на пример, све слике са истим лицима могу бити груписане у један фолдер, или слике са исте локације аутоматски препознате и постављене заједно. Такође, кластеровање може омогућити ефикасно претраживање огромних збирки чланака, вести или књига. На пример, у великим новинским архивама, алгоритми могу аутоматски груписати чланке који говоре о истим темама, као што су политика, спорт или култура, чинећи претраживање бржим и једноставнијим.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) је популаран алгоритам за кластеровање података заснован на густини тачака (података) у простору. Његова главна идеја је да идентификује густе просторе тачака као кластере, док тачке које су изоловане или се налазе у ретким областима означава као шум. Овај алгоритам користи два основна параметра:

- r - максимално растојање између тачака како би се сматрале блиским.
- n - минималан потребан број тачака блиских некој тачки x како би се простор око тачке x сматрао кластером.

DBSCAN се састоји из следећих корака:

Фаза означавања

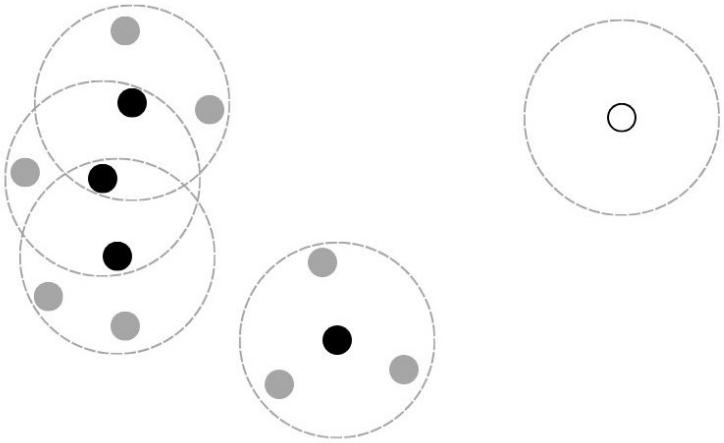
1. Бира се тачка x из скупа података који је доступан.
2. За тачку x се израчунава број N − колико тачака се налази на растојању мањем или једнаком од r, не убрајајући тачку x.
   - Уколико важи $N >= n$, тачка x чини **језгро кластера**.
   - Уколико важи $N < n$, и уколико се x налази на максималном растојању r од неке тачке која чини језгро кластера, тада тачка x представља **граничну тачку кластера**.
   - Уколико важи $N < n$, и уколико се x не налази на максималном растојању r од неке тачке која чини језгро кластера, тада тачка x представља **шум** (изоловану тачку).
3. Кораци 1. и 2. се понављају док се не обраде све тачке из скупа података.

Фаза формирања кластера

Свака тачка која је означена као језгро служи као почетна тачка за формирање кластера. Све тачке које се налазе на растојању мањем од r од ње, припадаће истог кластеру као и она. Уколико се међу тим тачкама налази још нека тачка језгра, тада ће се кластер проширити и на тачке које су на растојању мањем од r и од те тачке језгра.

Пример. На слици 1 може се видети пример DBSCAN алгоритма над конкретним подацима. Црном бојом означене су тачке које чине језгро, сивом бојом граничне тачке а тачке које су непопуњене представљају шум. На левом делу слике се уочава један кластер, који је настао надовезивањем три тачке језгра са њиховим околним граничним тачкама. У средишњем делу уочава је још један одвојени кластер, који чини једна тачка језгра са своје три граничне тачке. У горњем десном углу налази се шум, и он није део ни једног кластера. Закључујемо да се у подацима налазе два кластера, тј. два простора густих тачака који су

раздвојени ретким простором. Управо смо уз помоћ параметара r и n дефинисали појам густине простора. Са другачијим избором вредности тих параметара, број кластера би био другачији, јер би самим тим и појам густине простора био другачији. У овом примеру су испрекиданом линијом исцртане области на растојању r (r-околине) од тачака језгра и шума, а за параметар n је узета вредност 3.



Слика 1: DBSCAN алгоритам - пример

Силуета је метрика која се користи за процену квалитета кластеровања, а вредности силуете дају информацију о томе колико су добро одвојени кластери и колико су тачке унутар кластера сличне једна другој. Силуета се рачуна за сваку тачку и даје резултат између -1 и 1, при чему вредност близу 1 означава да је тачка добро постављена унутар свог кластера и далеко од других кластера, вредност близу 0 означава да је тачка на ивици између два кластера а вредност близу -1 означава да је тачка вероватно погрешно додељена кластеру.

$$s(x) = \frac{b(x) - a(x)}{\max(a(x), b(x))}$$

Где су:

- $a(x)$ - просечно растојање између тачке x и свих других тачака унутар тог истог кластера.
- $b(x)$ - просечно растојање између тачке x и свих других тачака из најближег суседног кластера.

Вредност силуете за цео скуп тачака рачуна се као просек силуета за све тачке:

$$S = \frac{1}{N} \sum_{i=1}^{N} s(i)$$

Приликом рачунања силуете, у обзир се не узимају подаци који представљају шум, тако да вредност N представља број података који нису шум.

Задатак. Замислите да радите на пројекту развоја туризма у Републици Србији. Ваш задатак је да групишете градове у туристичке кластере тако да градови унутар истог кластера треба да буду довољно близу како би могли ефикасно сарађивати на промоцији заједничких атракција, развоју смештајних капацитета и организацији међуградских тура. За овај задатак ћете управо користити DBSCAN алгоритам са параметрима $r = 80km$ и $n = 2$ и матрицу растојања између градова која је наведена у табели 1.

(а) [2 поена] Број кластера добијен кластеровањем са задатим параметрима је:

а) 1 б) 2 в) 3 г) 4

(б) [2 поена] Уколико Ј представља број градова који чине језгро кластера, Г број градова који представљају граничне тачке кластера и Ш број градова који представљају шум, тада је Ј+Г−Ш:

а) 6 б) 7 в) 8 г) 9

(в) [**2 поена**] Заокружи тврдњу која је тачна:

а) Повећањем вредности параметра r, број кластера расте.

б) Смедерево (СД) има најбољу вредност силуете у односу на остале градове.

ц) Тачка која је шум у својој r-околини може имати тачку која је гранична.

д) Све претходне тврдње су нетачне.

(г) [**3 поена**] Вредност силуете за цео скуп тачака припада интервалу:

а) $[0.5, 0.6)$ б) $[0.6, 0.7)$ в) $[0.7, 0.8)$ г) ни један од понуђених

Табела 1: Растојања између градова

	БГ	СД	ЈА	КИ	КШ	НИ	УЕ	НС	ПА	ЗР	ВР	ББ
БГ	0	70	140	130	200	240	200	80	160	90	340	170
СД	70	0	100	160	150	190	230	140	110	120	290	240
ЈА	140	100	0	310	60	100	150	230	20	210	200	180
КИ	130	160	310	0	350	400	340	100	330	50	550	330
КШ	200	150	60	350	0	80	130	270	40	250	160	150
НИ	240	190	100	400	80	0	250	330	90	310	120	270
УЕ	200	230	150	340	130	250	0	250	170	270	290	30
НС	80	140	230	100	270	330	250	0	260	60	450	230
ПА	160	110	20	330	40	90	170	260	0	240	180	190
ЗР	90	120	210	50	250	310	270	60	240	0	430	285
ВР	340	290	200	550	160	120	290	450	180	430	0	300
ББ	170	240	180	330	150	270	30	230	190	285	300	0

Задатак 8. Алгоритми и структуре података

Велики број проблема са којим се сусрећемо је индуктивне природе, и у њиховом решавању се јављају рекурзивне функције, то јест функције које позивају саме себе. У многим случајевима се дешава да током извршавања рекурзивне функције долази до преклапања рекурзивних позива тј. да се идентични рекурзивни позиви (рекурзивни позиви са идентичним параметрима) извршавају више пута. Ако се то дешава често, програми су по правилу веома неефикасни (у многим случајевима број рекурзивних позива, па самим тим и сложеност бива експоненцијална у односу на величину улаза). До ефикаснијег решења се често може доћи техником динамичког програмирања. Оно често временску ефикасност поправља ангажовањем додатне меморије у којој се бележе резултати извршених рекурзивних позива. Динамичко програмирање долази у два облика.

- Техника **мемоизације** или **динамичког програмирања наниже** задржава рекурзивну дефиницију али у додатној структури података (најчешће низу или матрици, ређе мапи тј. речнику) бележи све резултате рекурзивних позива, да бих их у наредним позивима у којима су параметри исти само очитала из те структуре.
- Техника **динамичког програмирања навише** у потпуности уклања рекурзију и ту помоћну структуру података попуњава исцрпно у неком систематичном редоследу. Дакле, рекурзивна конструкција се замењује индуктивном, тј. итеративном.

Док се код мемоизације може десити да се рекурзивна функција не позива за неке вредности параметара, код динамичког програмирања навише се израчунавају вредности функције за све могуће вредности њених параметара мањих од вредности која се заправо тражи у задатку. Иако се на основу овога може помислити да је мемоизација ефикаснија техника, у пракси је чешћи случај да је током одмотавања рекурзије потребно израчунати вредност рекурзивне функције за баш велики број различитих параметара, тако да се ове предности мемоизаиције у пракси ретко сређе.

Пример. Најбољи начин да разјаснимо технику динамичког програмирања је да је илуструјемо на погодно одабраном примеру. За то ћемо користити Фибоначијев низ, који је опште познати проблем и кроз чије се решавање могу илустровати већина основних концепата динамичког програмирања.

Основно рекурзивно решење Имплементацију можемо направити рекурзивно, директно из дефиниције (за $n = 0$ и $n = 1$ функција враћа 1, а у супротном враћа збир рекурентних позива за $n - 1$ и $n - 2$).

```
function fib(n):
  if n == 0 or n == 1:
    return 1
  else:
    return fib(n - 1) + fib(n - 2)
```

Неефикасност овакве имплементације се може видети у томе што се рекурзивни позиви за исте вредности врше више пута. Зато ћемо видети како динамичким програмирањем можемо доћи до ефикасније имплементације.

Мемоизација уз коришћење статичког низа Прва могућност је да применимо мемоизацију. Резултате рекурзивних позива ћемо памтити у некој помоћној структури (у овом случају, статичком низу), и онда ћемо на почетку сваког рекурзивног позива проверавати да ли је резлтат за текућу вредност већ израчунат. Пошто морамо некако обележити вредост у низу која још није рачуната, то радимо помоћу неке специјалне вредности, у овом случају, како су сви фибоначијеви бројеви позитивни, бирамо -1/

```
function fib(n, memo):
    // Ako je vrednost već izračunata, vraćamo je
    if memo[n] != -1:
        return memo[n]

    // Bazni slučajevi
    if n == 0 or n == 1:
        memo[n] = 1
        return memo[n]

    // Inače, računamo n-ti element niza memo i vraćamo ga
    memo[n] = fib(n - 1, memo) + fib(n - 2, memo)
    return memo[n]

function fin(n):
    // Pravimo niz memo od n+1 elemenata
    memo[n+1]

    // Inicijalizujemo elemente niza memo na -1
    init(n, memo)

    return fib(n, memo)

function init(n, memo):
    for i = 0 to n:
        memo[i] = -1
```

Динамичко програмирање навише Друга могућност је да применимо динамичко програмирање навише. Техника динамичког програмирања навише подразумева уклањање рекурзије (она је подложна многим техничким ограничењима) и да се све вредности у низу попуне неким редоследом. Пошто вредности на вишим позиција зависе од вредности на нижим, то радимо са лева на десно. Прво попуњавамо прва два елемента (базне случајеве) па остале попуњавамо у петљи на основу претходне две.

```
function fib(n):
    // Pravimo niz dp od n+1 elemenata
    dp[n+1]

    //Bazni slučajevi
    dp[0] = 1
    dp[1] = 1

    // popunjavamo dp niz od 2 do n
    for i = 2 to n:
        dp[i] = dp[i - 1] + dp[i - 2]

    print(dp[n])
```

У овом задатку треба да дописете одговоре на линију. Негативне поене не можете да добијете.

Задатак. Дате су вредности n ($n \leq 100$) различитих новчића. Наћи најмањи број новчића потребан да се добије укупна сума s ($s \leq 1000$).

(а) [**4 поена**] Допунити празнине у коду следећег рекурзивног решења овог задатка:

```
//funkcijom računamo najmanji broj novčića potreban da se
//naplati suma s kad su nam na raspolaganju n novčića u nizu novčići
function minBrojNovcica(novcici[], n, s):
    //ako je potrebna suma za naplatu 0 to radimo sa 0 novčića
    if s == 0:
        return 0

    //ako nemamo više novčića ne možemo naplatiti, INF predstavlja beskonačno tj da nema rešenja
    if n == 0:
        return INF


    //posmatramo poslednju vrstu novčića, želimo da proverimo da li
    //postižemo minimum kad koristimo novčić poslednje vrste ili
    //bez njega. promenljiva br predstavlja traženi broj

    br = 0

    1._____

    if s >= novcici[n - 1]:
       2._____


    return br
```

Одговор:
1._____
2._____

(б)[5 поена] Допунити празнине у коду решења које користи динамичко програмирање навише:

```
//funkcijom računamo najmanji broj novčića potreban da se
//naplati suma s kad su nam na raspolaganju n novčića u nizu novčići
//niz dp predstavlja najmanji broj novčića koji je potreban za naplatu sume s
//za svaki njegov element, tj. vrednost d[k] predstavlja najmanji broj novčića
//potreban za naplatu sume k

function minBrojNovcica(novcici[], n, sum):

    //Pravimo niz za dinamičko programiranje
    dp[MAX_SUMA + 1]

    //počinjemo analizu sa 0 vrsta novčića

    //vrednost 0 naplaćujemo sa 0 novičića
    dp[0] = 0

    //ostale vrednosti ne možemo naplatiti
    for i = 1 to sum:
        dp[i] = INF

    //popunjavamo niz dp tako što uključujemo novčiće vrstu po vrstu,
    //za svaku vrstu ažuriramo minimume tako što proverimo da li je moguće
    //uključiti novčić tekuće vrste, i ažuriramo minimum ako je potrebno


    for i = 1 to n:
        for j = 0 to sum:
            if j >= novcici[i - 1]:
                1._____


    //vraćamo najmanji broj novčića za sumu s
    return dp[s]
```

Одговор: 1. _____

Задатак 9. Лексичка анализа

Програмски језици, попут природних језика, имају **синтаксу** и **семантику**. У природном језику, синтакса одређује правила за слагање речи у реченице (нпр. редослед речи), док семантика одређује значење тих реченица. Слично, у програмирању, синтакса су правила која одређују како се пишу изрази и команде, док семантика одређује шта ти изрази и команде значе и како ће рачунар реаговати на њих. Регуларни изрази у програмским језицима баве се провером синтаксе програмског језика.

Правила регуларних израза

- `[abcde]` - У регуларном изразу се може појавити слово a, b, c, d или e.
- `[a-zA-Z]` - У регуларном изразу се може појавити било које латинично слово.
- `[0-9]` - У регуларном изразу се може појавити било која цифра.
- `[a-zA-Z0-9_]` - Означава било који знак који може бити слово, број или доња црта.
- `+` - Означава "један или више претходних знакова". На пример, `[0-9]+` значи да се може појавити било која комбинација цифара (нпр. `1`, `123`, `01234`, итд.).
- `*` - Означава "нула или више претходних знакова". На пример, `[0-9]*` значи да се може појавити било која комбинација цифара (нпр. `1`, `123`, `01234`), као и празна реч (реч без иједног карактера).
- `.` - Ознака за било који карактер (осим краја линије).
- `\s` - Ознака за било који размак (*space*, таб, нови ред).
- `|` - Ознака за или.($a|b$ означава да реч може да буде а или b).
- `?` - Означава да је претходни елемент опцион, тј. може се појавити нула или један пут. На пример, за регуларни израз $ba?$ речи које би биле прихваћене су ba и b, јер а може а и не мора да се појави у речи.

Специјални карактери

Неки карактери у регуларним изразима имају специјално значење, па их не можемо користити тек тако. На пример, уколико желимо да напишемо регуларни израз за децимални број, следећи запис би био погрешан:

$$[0-9]+.[0-9]+$$

Разлог је што симбол `.` има специјално значење у регуларим изразима и означава "било који карактер". Тако би овај израз, поред децималног броја као што је `123.456`, прихватио и низ као што је $123m456$.

Да бисмо одузели симболима њихово специјално значење, користимо симбол `\`. Правилно написан регуларни израз за децималне бројеве би био:

$$[0-9]+\.[0-9]+$$

Специјални карактери у регуларним изразима

Списак најчешће коришћених специјалних карактера: $., *, +, ?, \{, \}, ', \$, (,), [,]$

Пример регуларног израза за илустрацију

Пример 1: Регуларни израз: $[ab]*$

Ово значи:

- У речи се могу појављивати слова a или b a и b (`[ab]`).
- Можеш имати нула или више понављања (`*`).

Шта **не** би било прихваћено овим регуларним изразом: *abc*, *abd...*
Шта би било прихваћено: празна реч, *aaabbb*, *ababa*, *bbb*, *aaa...*

Пример 2: Регуларни израз: $(ab)*$

Ово значи:

- Дозволи реч *ab*.
- Можеш имати нула или више понављања речи (`*`).

Шта **не** би било прихваћено овим регуларним изразом: *aaa*, *bbbb*, *abaa*
Шта би било прихваћено: празна реч, *ab*, *abab*, *abab...*

Пример 3: Регуларни израз: $\backslash s * primer$

Ово значи:

- $\backslash s$ значи да је испред белина.
- $\backslash s*$ значи да може да има нула или више белина.

Шта не би било прихваћено овим регуларним изразом: празна реч, `нестопример`
Шта би било прихваћено: *primer*, *primer*, *primer* (обратити пажњу на белине)

Задатак - Наредба доделе у програмском језику C

Наредба доделе у језику C омогућава да се вредност неког израза додели променљивој. Ова наредба је основни начин за чување и манипулацију подацима у програму.

Наредба доделе у језику C има следећу структуру:

- На почетку се налази име променљиве(прочитати испод како написати регуларни израз за наредбу доделе).
- Испред и иза имена променљиве могу се наћи белине.
- Након тога следи знак једнакости (=). Испред и иза знака једнакости могу се налазити белине
- Иза знака једнакости, због једноставности, можемо претпоставити следеће:
  - Може стајати бројна константа
  - Може стајати променљива
  - Може стајати само један збир или производ бројих константи или имена променљивих (због једноставности задатка)
- На крају наредбе доделе мора стајати ";"

Примери валидних наредби доделе:

- $x = 5;$
- $\_counter = value + 1;$
- $number = 3 + 1;$
- $\_counter = value;$

Примери невалидних наредби доделе:

- $5x = 10;$ (име променљиве не може почињати бројем)

- $x =$; (недостаје израз са десне стране)
- $result = a + b$ (недостаје тачка-запета на крају)
- $result = a * b + c$;(збир од 3 сабирка)
- $result$?5;(уместо знака једнакости недозвољен карактер)

Да би име променљиве било валидно, мора да задовољава следећа правила:

- Може садржати само слова (мала и велика), бројеве и доњу црту (_).
- Мора почињати словом или доњом цртом, али не бројем.
- Не сме садржавати специјалне знакове као што су @, %, # - итд.

Примери валидних имена променљивих у *C* језику:

- x
- $broj\_1$
- $\_privatna$
- $temp2$

Примери невалидних имена променљивих:

- $3broj$ (почиње бројем)
- $\_\_nijeTacnaPromenljiva$ (почиње са две доње црте)
- $123netacno$ (почиње бројем)
- $ime - varijable$ (садржи црту -)
- $@ovonijepromenljiva$ (садржи знак @)

Напомена о резервисаним речима:
Регуларни израз сам по себи не може препознати резервисане речи у језику *C*. Ово значи да би, на пример, реч *int* била валидна према регуларном изразу, али додатна провера семантичке исправности (нпр. поређење са листом кључних речи) мора бити изведена у програму који користи овај регуларни израз. Током израде задатака дозвољено је да регуларни израз прихвата кључне речи програмског језика Ц (*int*, *void*...).

(а) [4 поена] Написати регуларни израз који препознаје наредбу доделе *C* програмског језика. :

Задатак - $\#include$ директиве у језику C

#include директива у језику *C* користи се за укључивање спољног кода у програм. Она омогућава повезивање са спољним фајловима, као што су:

Стандардне библиотеке (нпр. *stdio.h*, *stdlib.h*) које пружају функције за улаз/излаз, математичке операције и друге основне функционалности.

Кориснички хеадер фајлови (нпр. "*mylib.h*") који могу садржавати дефиниције функција, структура, константи, итд., које користиш у свом пројекту.

Да би нека реч била *#include* директива, мора да поштује следећа правила:

- Библиотеке са <> синтаксом (нпр. *#include* < *stdio.h* >).
- Библиотеке са "" синтаксом (нпр. *#include*"*myheader.h*").
- Име билиотеке може бити састављено искључиво од слова.
- На крају имена библиотеке мора постојати ".*h*" екстензија.

- Белине се могу наћи пре и после кључне речи *#include*.

Примери валидних директива:

- *#include* $< stdio.h >$
- *#include*"*utils.h*"
- *#include* $< math.h >$
- *#include*"*myheader.h*"
- *#include* $< lib.h >$

Примери невалидних директива:

- *#include* $< stdio >$ // фали екстензија *.h*
- *#include*"*utils*" // фали екстензија *.h*
- *#include* $< 123file.h >$ // име фајла не може почети бројем
- *#include*"$my - header.h$" // садржи црту - у имену библиотеке
- *#include* $< stdio.h >$ // ако није баш у облику «...» или "... ", већ нпр. има неки размак или други карактер
- *#include*"*myheader.hh*" // не одговара јер екстензија није *.h*
- *nesto* $< stdio.h >$ // не почиње са *#include*, и потенцијалним белинама

(б) [**2 поена**] Који од следећих регуларних израза препознаје све облике *#include* директива у *C* коду

Решење

а) $\backslash s * \#include \backslash s * ((< [a - zA - Z] + \backslash .h >) | ("[a - zA - Z] + \backslash .h"))$

б) $\backslash s * \#include \backslash s * [< "](([a - zA - Z] + \backslash .h)[> "]$

ц) $\backslash s * \#include \backslash s * [< "](([a - zA - Z] * \backslash .h)[> "]$

д) ништа од наведеног

Задатак - Коментари у језику C

Коментари у језику *C* користе се за објашњавање кода и олакшавање читања програма. Коментари се игноришу током компилације, па не утичу на извршавање програма. Једнолинијски коментар је у форми такав да на почетку има две косе црте // па затим следи низ карактера, и коментар се завршава карактером новог реда.Коментар такође може бити и празна линија (након црта које означавају почетак коментара облика "//").

// Ovo je komentar

/Ovo nije komentar jer nema dve kose crte na pocetku komentara

// Ovo nije jednolinijski komentar
jer se nalazi na vise linija

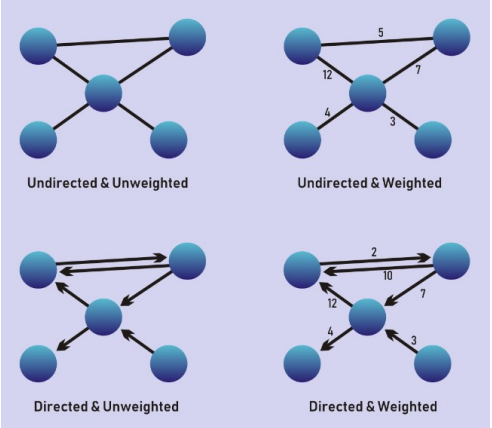
(в) [**3 поена**] Напиши регуларни израз који препознаје једнолинијски коментар у језику *C*.

Решење

Задатак 10. Вештачка интелигенција

Граф је структура која се састоји од скупа чворова и грана које повезују те чворове. Формално, граф се дефинише као $G = (V, E)$, где је:

- **V скуп чворова,**
- **E скуп грана које повезују парове чворова.**



Графови могу бити усмерени (гране имају смер) или неусмерени (гране немају смер). Такође, графови могу бити тежински или нетежински:

- **У тежинским графовима свакој грани је придружена одређена тежина (нпр. удаљеност, време, трошак...).**
- **У нетежинским графовима све гране имају једнаку тежину.**

Графови су корисни у многим свакодневним ситуацијама, попут планирања путева у навигационим апликацијама, оптимизације мрежа у телекомуникацијама и анализе друштвених односа на платформама попут *Facebook*-а. Такође се користе у логистици за планирање рута испоруке и у анализи података за повезивање сложених система. Њихова примена омогућава ефикасније доношење одлука и решавање сложених проблема.

Проблеми на графовима често укључују проналажење одређених путања, као што је најкраћа путања између два чвора. Један од најефикаснијих алгоритама за овај задатак и уједно један од најважнијих алгоритама вештачке интелигенције је алгоритам A* (А звезда). Алгоритам A* користи хеуристике (процене које воде претрагу ка циљу на основу приближних, али информативних података о могућем растојању или трошковима) како би усмерио претрагу ка циљу, испуњавајући, при одређеним условима, два важна својства:

- Потпуност – алгоритам ће увек пронаћи решење ако оно постоји.
- Оптималност – алгоритам гарантује решење са најмањом ценом (нпр. минимална удаљеност, најнижи трошак итд.). Оптимално решење не мора увек бити јединствено!

Алгоритам A* је варијанта алгоритма "Прво најбољи", која даје предност чворовима са најбољом оценом. Функција евалуације f (функција која оцењује квалитет стања) у овом алгоритму има следећу форму:

$$f(n) = g(n) + h(n),$$

где је:

- $g(n)$: тренутно позната цена пута од почетног чвора до чвора н,
- $h(n)$: процењена (хеуристичка) цена најјефтинијег пута од н до циљног чвора.

Док се претрага одвија:

- Вредност $g(n)$ се ажурира како алгоритам открива нове и јефтиније путеве.
- Вредност $h(n)$ остаје непромењена, јер представља процену засновану на унапред одређеној хеуристици.

Одабир добре хеуристике је пресудан за ефикасност алгоритма!

Основни појмови алгоритма A*:

- Затворена листа – листа посећених чворова за које су сигурно већ посећени сви суседи (односно сва непосредно доступна стања).
- Отворена листа – листа посећених чворова за које можда нису посећени сви њихови суседи.
- Родитељски чвор – представља чвор из којег се долази до тренутног чвора.

ОПИС АЛГОРИТМА А*

Улаз: Граф G, полазни чвор, циљни чвор и хеуристичка функција h.
Излаз: Пут од полазног до циљног чвора или неуспех.

- Иницијализуј:
  - Отворену листу са полазним чвором.
  - Затворену листу као празну.
- Докле год има чворова у отвореној листи:
  - Изабери чвор n из отворене листе са најбољом оценом $f(n)$.
  - Ако је n циљни чвор:
    - Врати најкраћи пут реконструишући га уназад од n до полазног чвора.
  - За сваког суседа m чвора n:
    - Ако m није ни у отвореној ни у затвореној листи:
      - Додај m у отворену листу.
      - Означи n као родитеља чвора m.
      - Израчунај $g(m)$ и $f(m)$ и придружи их чвору m.
    - Иначе (ако m већ припада некој листи), али је пут преко n јефтинији:
      - Промени информацију о родитељу чвора m на n.
      - Ажурирај вредности $g(m)$ и $f(m)$.
      - Ако је m у затвореној листи, пребацуј га у отворену.
  - Избаци n из отворене листе и убаци n у затворену листу.
- Ако је отворена листа празна, а циљ није достигнут, закључи да решење не постоји.

а) [7 поена] Претпоставимо да планирате пут од Београда до Бањалуке, али због бројних временских непогода морате да узмете хеуристичке процене са сајта министарства саобраћаја и трговине како бисте прилагодили план пута тренутним условима. Уколико су дате удаљености између путем повезаних градова, као и хеуристичка процена удаљености датих градова до Бањалуке, одреди:
Пут од Београда до Бањалуке _____
Редослед додавања градова у затворену листу. _____

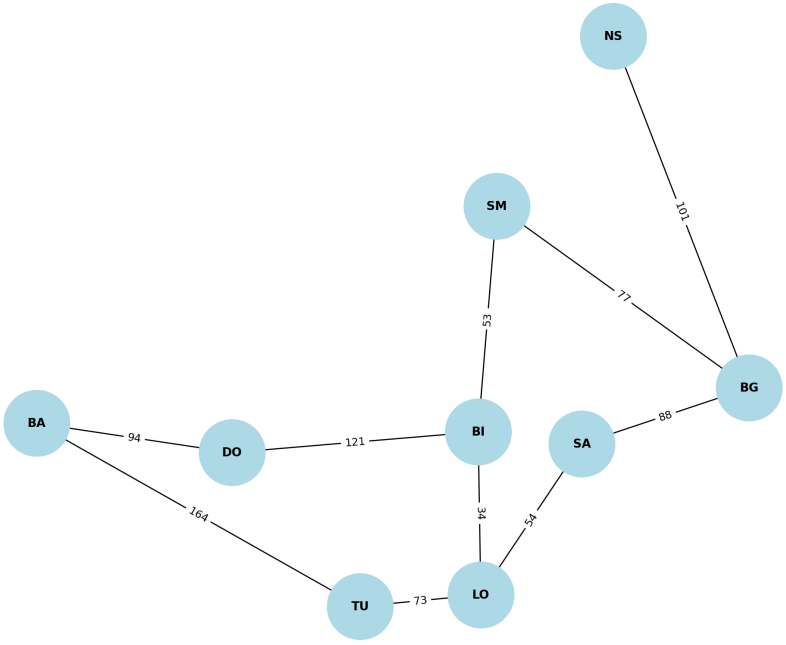
Градови: Београд (БГ), Нови Сад (НС), Сремска Митровица (СМ), Шабац (СА), Лозница (ЛО), Бијељина (БИ), Добој (ДО), Тузла (ТУ), Бањалука (БЛ).

Граф

Чворови	Удаљеност
БГ ↔ НС	101
БГ ↔ СМ	77
БГ ↔ ША	88
ША ↔ ЛО	54
ЛО ↔ БИ	34
ЛО ↔ ТУ	73
ТУ ↔ БЛ	164
СМ ↔ БИ	53
БИ ↔ ДО	121
ДО ↔ БЛ	94

Хеуристичке процене до Бањалуке

Град	Хеуристичка процена
БГ	350
НС	270
СМ	250
ША	210
ЛО	180
БИ	150
ДО	100
ТУ	200
БЛ	0



ХЕУРИСТИКА

Допустива хеуристика - хеуристика (h) која никада не прецењује стварно растојање између текућег чвора и циљног чвора, односно за сваки чвор важи:

$$h(n) \leq h^*(n),$$

где је $h^*(n)$ оптимална хеуристика (цена најкраћег пута од чвора н до циљног чвора).

Уколико се у алгоритму A* користи допустива хеуристика, ако је пронађен пут до циљног чвора, онда је он сигурно оптималан.

Конзистентна хеуристика - хеуристика (h) је конзистентна ако има вредност 0 за циљни чвор и за било која два суседна чвора n и m важи:

$$c(n, m) + h(m) \geq h(n),$$

где је $c(n, m)$ цена придружене (усмерене или неусмерене) грани (n, m).

Ако је хеуристика конзистентна, онда је она и допустива хеуристика.

Ако се у алгоритму A* користи конзистентна хеуристика h, онда:

- ако је пронађен пут до циљног чвора, онда је он сигурно оптималан,
- за чворове доступне из текућег чвора који су већ у затвореној листи, не треба се проверавати да ли њихова вредност g може да се ажурира.

б) [5 поена] Често се као хеуристичка процена удаљености 2 града користи ваздушна удаљеност, јер је сигурно мања од стварног пута и тиме обезбеђује својство оптималности, док је конзистентност задовољена из неједнакости троугла (јер је збир две странице троугла увек већи од треће). Под претпоставком да су следеће хеуристичке вредности заправо ваздушне удаљености између градова и Бањалуке, одреди нови редослед додавања градова у затворену листу.

Град	Хеуристичка удаљеност (км)
БГ	258
НС	215
СМ	187
ША	198
ЛО	164
БИ	160
ДО	71
ТУ	120

ДАЈКСТРА

Алгоритам Дајкстра се користи за проналажење најкраћег пута у тежинским графовима - графовима где ивице имају различите тежине, тј. представљају удаљености, цене, или било који други параметар који може бити представљен као број. Да би алгоритам функционисао, важно је да све тежине ивица буду ненегативне!

Рад Дајкстре се заснива на следећим корацима:

1. **Иницијализација**: За сваки чвор у графу постави удаљеност на бесконачност, осим за почетни чвор који има удаљеност 0.
2. **Избор чвора**: Почни од почетног чвора и изабери чвор са најмањом удаљеношћу који није још обрађен.
3. **Ажурирање удаљености**: За сваког суседа изабраног чвора израчунај удаљеност до њега кроз тренутни чвор. Ако је та удаљеност мања од тренутне удаљености тог суседа, ажурирај удаљеност.
4. **Понављање**: Понови кораке 2 и 3 за све чворове док не обрадиш све чворове или не нађеш најкраћи пут до циља.

в) [2 поена] Колико треба да износи хеуристичка функција од било ког чвора до циљног чвора како би се алгоритам A* понашао као Дајкстра?
