ОДСЕК ЗА СОФТВЕРСКО ИНЖЕЊЕРСТВО

АЛГОРИТМИ И СТРУКТУРЕ ПОДАТАКА 1

2024-2025

- домаћи задатак –

Опште напомене:

1. Домаћи задатак састоји се од једног програмског проблема. Студенти проблем решавају **самостално**, на програмском језику C или Python. Није дозвољено користити готове структуре података.
2. Пре одбране, сви студенти раде тест знања који се ради на рачунару коришћењем система *Moodle* (http://elearning.rcub.bg.ac.rs/moodle/). **Сви студенти треба да креирају налог и пријаве се на курс пре почетка лабораторијских вежби.** Пријава на курс ће бити прихваћена и важећа само уколико се студент региструје путем свог налога електронске поште на серверу **mail.student.etf.bg.ac.rs**.
3. Реализовани програм треба да комуницира са корисником путем једноставног менија који приказује реализоване операције и омогућава сукцесивну примену операција у произвољном редоследу.
4. Унос података треба омогућити путем читања са стандардног улаза и из датотеке.
5. Решења треба да буду отпорна на грешке и треба да кориснику пруже јасно обавештење у случају детекције грешке.
6. Приликом оцењивања, биће узето у обзир рационално коришћење ресурса. **Примена рекурзије се неће признати као успешно решење проблема које може освојити максималан број поена.**
7. За све недовољно јасне захтеве у задатку, студенти треба да усвоје разумну претпоставку у вези реализације програма. Приликом одбране, демонстраторе треба обавестити које претпоставке су усвојене и која су ограничења програма (на пример, максимална димензија матрице и слично). Неоправдано увођење ограничавајуће претпоставке повлачи негативне поене.
8. Одбрана домаћег задатка ће се обавити према распореду који ће накнадно бити објављен на сајту предмета.
9. За решавање задатака који имају више комбинација користити следеће формуле. (**R** – редни број индекса, **G** – последње две цифре године уписа):

   i = (R + G) mod 3

   j = (R + G) mod 2
10. Име датотеке која се предаје мора бити **dz1.c|py**. Дозвољено је и имати више фајлова (на пример .h и .c), али фајл који садржи главни програм задатка 2 треба да буде именован на наведени начин. Уколико се формат датотеке која се предаје не испоштује, студент губи 25 поена.
11. Предаја домаћих ће бити омогућена преко *Moodle* система. Детаљније информације ће бити благовремено објављене.
12. Предметни наставници задржавају право да изврше проверу сличности предатих домаћих задатака и коригују освојени број поена након одбране домаћих задатака, као и да пријаве теже случајеве повреде Правилника о дисциплинској одговорности студената Универзитета у Београду Дисциплинској комисији Факултета.

Задатак 1 – Уланчане листе [35 поена]

Написати интерактивни програм који илуструје рад са одговарајућим типом уланчаних листа.

У зависности од редног броја проблема ***i***, треба имплементирати и користити уланчану листу која је следећег типа:

0. Једноструко уланчана листа са заглављем
1. Једноструко уланчана кружна листа без заглавља
2. Двоструко уланчана листа без заглавља

Потребно је имплементирати следеће операције за рад са уланчаним листама:

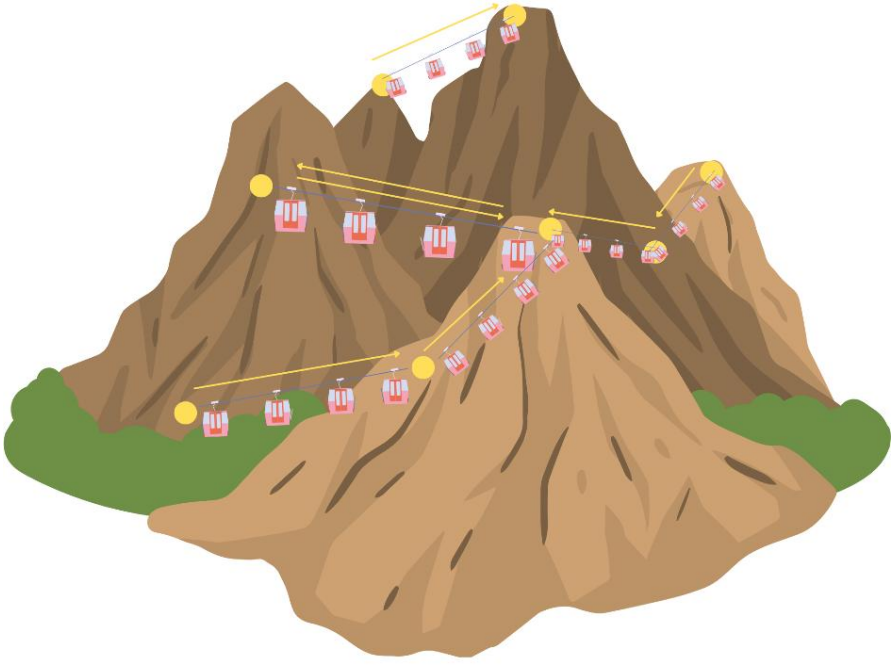
1. **[5 поена]** Креирање и уништавање уланчане листе
2. **[5 поена]** Испис уланчане листе
3. **[10 поена]** Уметање елемента на задату позицију
4. **[10 поена]** Брисање елемента са задатом вредношћу
5. **[5 поена]** Претрага листе на основу задате вредности

Садржај информационог поља чвора се оставља студенту на имплементацију и може бити прилагођен потребама задатка 2.

Корисник са програмом интерагује путем једноставног менија. Програм треба да испише садржај менија, а затим да чека да корисник изабере (унесе путем тастатуре) редни број неке од понуђених ставки, након чега, пре извршења, од корисника очекује да по потреби унесе додатне параметре. Поступак се понавља све док корисник у менију не изабере опцију за прекид програма.

Задатак 2 – Графови [65 поена]

За моделовање система жичара на једном планинском венцу се користи усмерени тежински граф. У графу су чворовима представљена места на планини која садрже разне активности за посетиоце. Грана између два чвора подразумева да постоји могућност да се жичаром превезе од прве до друге тачке на планини. Између неких места постоји жичара која превози само у једном смеру, а између неких она која превози у оба смера.



Свако место је одређено јединственим идентификатором (нпр. А2, где је А идентификатор планине, а 2 редни број жичаре на планини А). Тежина гране између два места моделује време потребно да се превезе жичаром, изражено у минутима.

Потребно је имплементирати следеће операције за рад са усмереним тежинским графом који се користи за моделовање жичара на овом планинском венцу:

1. **[10 поена]** Креирање и уништавање графа
2. **[10 поена]** Додавање чвора у граф и уклањање чвора из графа
3. **[10 поена]** Додавање и уклањање гране између два чвора у графу
4. **[5 поена]** Испис репрезентације графа

Потребно је и омогућити учитавање целог плана планинског венца из датотеке по следећем формату. Први ред датотеке садржи идентификаторе свих места одвојене запетама, а сви остали редови у датотеци представљају информације о жичарама у облику *id1-id2-vreme*, где су *id1* и *id2* идентификатори места, а *vreme* је време потребно да се од места *id1* стигне жичаром до места *id2*.

Приликом решавања задатог проблема, потребно је користити **листе суседности** као меморијску репрезентацију графова. За потребе овакве репрезентације, обавезно је користити имплементирану структуру уланчане листе из задатка 1. Уколико се у решењу користе стекови или редови, они такође морају бити имплементирани користећи листу из задатка 1.

Додатно, ради ефикасније организације обиласка планине, потребно је омогућити:

1. **[20 поена]** Одређивање свих места на планини до којих се може доћи жичаром од задате полазне тачке
2. **[10 поена]** Одређивање најкраћег пута између два задата места на планини коришћењем *Dijkstra*-иног алгоритма

У зависности од редног броја проблема ***j***, одређивање свих места на планини до којих се може доћи жичаром од задате полазне тачке реализовати коришћењем:

0. Обиласка графа по дубини (*DFS*)
1. Обиласка графа по ширини (*BFS*)

Корисник са програмом интерагује путем једноставног менија. Програм треба да испише садржај менија, а затим да чека да корисник изабере (унесе путем тастатуре) редни број неке од понуђених ставки, након чега, пре извршења, од корисника очекује да по потреби унесе додатне параметре. Поступак се понавља све док корисник у менију не изабере опцију за прекид програма.

Напомене

По потреби реализовати и додатне методе, где је то примерено.

За тестирање програма се могу користити датотеке које се налазе у оквиру посебне архиве.

Датотеке у језику Python

Преглед најбитнијих функција за рад са датотекама у језику Python је дат у наставку.

`open(ime_datoteke, režim_pristupa)`	Отвара датотеку задатог имена у одређеном режиму приступа.
`f.close()`	Затвара датотеку *f*.
`f.readline()`	Чита једну линију текста из *f* и враћа стринг.
`f.write(string)`	Уписује у датотеку *f* задати стринг.

`with open('ulaz.txt', 'r') as f:`
`  for line in f:`
`  print(line, end='')`
	Пример отварања датотеке у режиму читања, читања једне по једне линије текста из датотеке, исписа прочитаних линија и затварања датотеке.

`with open('izlaz.txt', 'w') as f:`
`  for line in lines:`
`  f.write(line)`
	Пример отварања датотеке у режиму уписа, уписа неколико стрингова у датотеку и затварања датотеке.

Датотеке у језику C

Преглед најбитнијих функција за рад са датотекама у језику C је дат у наставку.

`FILE *fopen(const char *imedat, const char *rezim);`	Отвара датотеку задатог имена у одређеном режиму приступа.
`int fclose(FILE *dat);`	Затвара датотеку *dat*.
`int fgetc(FILE *dat);`	Чита и враћа један карактер из датотеке *dat*.
`char *fgets(char *tekst, int n, FILE *dat);`	Чита једну линију из датотеке *dat* и смешта у низ карактера *tekst*.
`int fputc(int zn, FILE *dat);`	Уписује у датотеку *dat* задати карактер.
`int fputs(const char *tekst, FILE *dat);`	Уписује у датотеку *dat* задати низ карактера *tekst*.

`FILE *in = fopen("ulaz.txt", "r");`
`if (!in) return 0;`
 	
`char ch;`
`while ((ch = fgetc(in)) != EOF) {`
`  putchar(ch);`
`}`
`fclose(in);`
	Пример отварања датотеке у режиму читања, читања једног по једног карактера, исписа прочитаних карактера и затварања датотеке.

`FILE *out = fopen("izlaz.txt", "w");`
`if (!out) return 0;`
 	
`for (char *p = text; *p; p++) {`
`  fputc(*p, out);`
`}`
`fclose(out);`
	Пример отварања датотеке у режиму уписа, уписивања једног по једног карактера из низа карактера и затварања датотеке.