АЛГОРИТМИ И СТРУКТУРЕ ПОДАТАКА 1
2019-2020
- домаћи задатак 3 -

Опште напомене:

1. 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, уколико то већ нису учинили.** 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**.
2. Домаћи задатак 3 састоји се од једног програмског проблема. Студенти проблем решавају **самостално**, на програмском језику C.
3. Реализовани програми треба да комуницирају са корисником путем једноставног менија који приказује реализоване операције и омогућава сукцесивну примену операција у произвољном редоследу.
4. Решења треба да буду отпорна на грешке и треба да кориснику пруже јасно обавештење у случају детекције грешке.
5. Приликом оцењивања, биће узето у обзир рационално коришћење ресурса. Решења која употребљавају рекурзију не могу добити максималан број поена.
6. За све недовољно јасне захтеве у задатку, студенти треба да усвоје разумну претпоставку у вези реализације програма. Приликом одбране, демонстраторе треба обавестити која претпоставка је усвојена (или које претпоставке су усвојене) и која су ограничења програма (на пример, максимална димензија матрице и слично). Неоправдано увођење ограничавајуће претпоставке повлачи негативне поене.
7. Предаја домаћих ће бити омогућена преко Moodle система до **понедељка, 01.06.2020. у 17:00**. Даље информације око предаје и одбране ће бити благовремено објављене.
8. Формула за редни број алгоритама **i** које треба користити приликом решавања задатка је следећа: (R – редни број индекса, G – последње две цифре године уписа):

$$\mathbf{i = (R + G) \bmod 3 + 1}$$

9. Име датотеке која се предају мора бити **dz3p1.c**
10. Предметни наставници задржавају право да изврше проверу сличности предатих домаћих задатака и коригују освојени број поена након одбране домаћих задатака.

22.05.2020. године *Са предмета*

О графовима

Граф је нелинеарна структура података која се састоји од скупа чворова и скупа грана. Гране представљају односе (везе) између чворова. Графови се могу користити за моделирање произвољних нелинеарних релација. Постоје усмерени и неусмерени графови.

Репрезентација графа

Граф се може репрезентовати коришћењем матрица суседности или листа суседности.

Више информација о меморијским репрезентацијама графа се може пронаћи у материјалима са предавања и вежби, као и у књизи проф. Мила Томашевића „Алгоритми и структуре података“.

Задатак - Репрезентација програмског кода (100 поена)

Како би се убрзало извршавање програма, програмски преводиоци раде многобројне оптимизације над програмским кодом. Оптимизације се спроводе над различитим репрезентацијама кода, од којих једна може бити графовска репрезентација. Паралелизам на нивоу инструкција (*instruction level paralelism*) представља парадигму по којој је могуће извршити више инструкција (операција) паралелно на више извршних јединица и на тај начин скратити укупно извршавање програма. Постоје, наравно, бројна ограничења која онемогућавају потпуну паралелизацију извршавања у општем случају, као што су ресурси машине, доступност аргумената инструкција и сл. У реалној ситуацији, оптимално распоређивање инструкција у складу са постојећим ограничењима може бити временски захтевно и непрактично и због тога се користе различите хеуристике као допуна алгоритмима за распоређивање инструкција ради добијања што паралелнијег кода.

Циљ овог задатка је представљање мањег и поједностављеног сегмента кода графовском репрезентацијом и спровођење одређених алгоритама над таквом репрезентацијом кода на програмском језику C. Све операције у задатом програмском сегменту су унарне или бинарне. У општем случају операције могу бити и сложенији изрази, али они у оквиру овог задатка неће бити разматрани. Трајање операције у у процесорским циклусима је одређено типом операције и биће дато табеларно за све операције које се могу јавити у тест програмским сегментима.

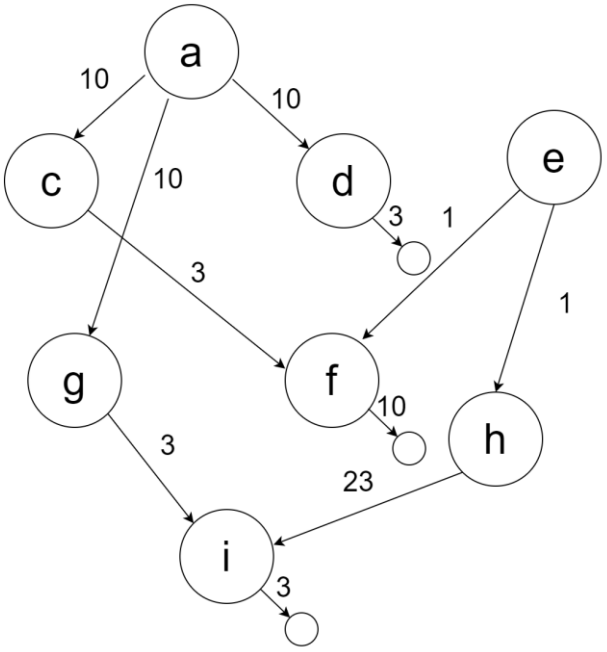
Програмски код се прослеђује на улаз (кроз командну линију или из датотеке) као скуп операција у виду стринга према следећем формату:

- Свака операција се налази у засебном реду
- Аргументи операција могу бити целобројне вредности или променљиве
- Променљиве се представљају малим словима енглеског алфабета (*a* - *z*)
- Ради поједностављења, резултат операције се увек уписује у нову промељиву (не постоји вишеструки упис у исту меморијску локацију)
- Не постоје инструкције скокова

Аргументи операције могу бити неке унапред спремне вредности или резултати неких претходно извршених операција. Уколико операција чека резултате извршавања неке друге операције кажемо да она зависи од те операције. Операција се може извршити онда када су сви њени аргументи доступни. Уколико у неком циклусу има више операција које имају спремне аргументе, онда су све кандидати за распоређивање за извршавање у том циклусу. У релаксираним условима задатка, када се каже да увек постоје доступни ресурси за извршавање операције када су њени аргументи спремни, то значи да се све овакве операције које задовољавају услов могу извршити у паралели у том циклусу.

Пример једног програмског сегмента и одговарајућа графовска репрезентација је дата у наставку. На слици је такође приказано и трајање операција чији резултат није аргумент ниједне операције.

```
1: a = 5 * 3
2: c = a + 7
3: d = a + 2
4: e = b xor 3
5: f = e * c
6: g = a + e
7: h = 82 / e
8: i = h - g
```



Операције које се могу јавити у коду су приказане у следећој табели као и њихово трајање у циклусима процесора.

ОПЕРАЦИЈА	ТРАЈАЊЕ
+, -	3
*	10
/	23
AND, OR, XOR, NOT	1

На овај начин дефинисан програмски сегмент се може моделовати усмереним ацикличним тежинским графом, где се операције моделују чворовима, а зависности међу операцијама гранама. Овакве зависности се називају зависностима по подацима, а граф се назива граф зависности по подацима (*data dependency graph*). Гране постоје само међу оним чворовима који представљају међусобно зависне операције и и усмерене су од чвора који репрезентује операцију која производи резултат ка чвору који представља операцију која зависи од тог резултата. Додељене тежине означавају трајање операција у циклусима процесора. Сваки чвор има свој идентификатор који представља редни број те операције у улазном програмском сегменту. По потреби могу се додати и додатне информације потребне за решавање задатака.

У складу са групом која се реализује, усвојити следећу репрезентацију графа:

1. Матрицу суседности
2. Листу суседности
3. Инверзну листа суседности

[30 поена] Основне операције над графом

Програм треба да садржи и користи помоћне функције за стварање графа, додавање чворова, додавање грана, брисање чворова, брисање грана и брисање целог графа. У складу са групом која се реализује и потребама задатка, усвојити и имплементирати структуру података за репрезентацију чвора графа над којом ће бити спровођене задате операције.

Укупан број задатих операција програмског сегмента директно одређује број чворова у графу. Идентификатори чворова се формирају према редном броју конкретне операције у том програму. Парсирањем појединачних операција додају се гране графа. На основу аргумената операције која се тренутно парсира потребно је увести грану за сваки аргумент који је производ извршавања неке од претходних операција. Потребно је осмислити начин мапирања резултата операције тако да се што ефикасније може увести нова грана.

[10 поена] Испис графа

Треба омогућити испис програмског кода на стандардни излаз или у текстуалну датотеку. Приликом исписа потребно је јасно приказати зависности међу операцијама у програмском сегменту.

[30 поена] Одређивање оптималних распореда операција

Трајање неког програмског сегмента могуће је скратити паралелним извршавањем више различитих операција. Као што је већ речено, најкраће трајање програмског сегмента одређује се на основу распореда операција у складу са ограничавајућим параметрима (ограничењима на основу зависности, доступним ресурсима машине за извршавање операција и сл.). Ради једноставности решења усвојићемо да је укупно трајање сегмента условљено једино критичним путем који постоји у добијеном графу, односно ограничењима која постоје на основу зависности. Потребно је реализовати проналажење критичног пута у добијеном графу (или више, уколико их има). Дужина критичног пута директно одређује трајање програмског сегмента који репрезентује на овај начин дефинисан граф.

Потребно је :

a) **[15 поена]** Одредити најкраће трајање конкретног програмског сегмента (дужину критичног пута)

b) **[5 поена]** Означити чворове који се налазе на критичном путу

c) **[10 поена]** Приказати различите могуће распореде свих операција тако да укупно трајање буде најкраће.

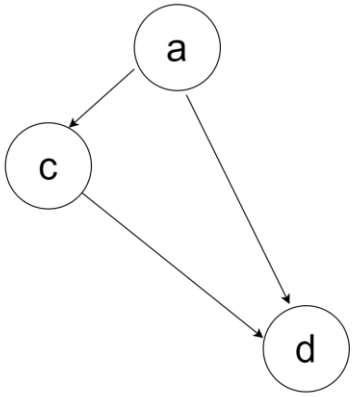
   Све операције, осим оних које се налазе на критичном путу могуће је варирати у распоређивању. Алгоритам за одређивање критичног пута даје за сваку операцију (чвор) информацију када најраније и најкасније може бити распоређена. То значи да је све операције које су ван критичног пута могуће распоредити у различитим циклусима у тим добијеним интервалима. У овиру овог задатка, потребно приказати различите могуће распореде и то тако да се за различите распореде испише за сваки циклус које операције се тада распоређују.

[20 поена] Одеђивање транзитивних зависности између операција

Транзитивна зависност представља индиректну зависност међу операцијама, нпр. У следећем примеру:

```
1: a = 5 * 6
2: c = a + 7
3: d = a * c
```

, операција 2 зависи од резултата прве операције по аргументу *a*, операција 3 зависи од операције 2 по аргументу *c*, али такође и од операције 1 по аргументу *a*. Ова зависност од операције 1 до операције 3 је транзитивна зависност због тога што постоји путања у графу 1 → 2 → 3 која директно диктира редослед операција. У том смислу, ова зависност 1→ 3 не носи никакву нову информацију и због тога је транзитивна. Последица је да операција 3 такође зависи од операције 1, али индиректно, као што је то приказано на слици.



Потребно је за задату операцију одредити на које све операције утиче индиректно. За реализацију овог проблема у складу са редним бројем додељене групе имплементирати неки од следећих алгоритама (могуће је и комбиновање са неким другим алгоритмима)

1. *Warshall* алгоритам
2. Обилазак графа по дубини (*DFS*)
3. Обилазак графа по ширини (*BFS*)

[10 поена] Комуникација са корисником

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, уз обавезно обавештење шта се од корисника очекује да унесе. Поступак се понавља све док корисник у менију не изабере опцију за прекид програма. Све наведене операције треба реализовати путем одговарајућих потпрограма чији је један од аргумената показивач на структуру података која имплементира граф са којим се ради.