МАТФ 2015, такмичење из информатике
Математички факултет, Београд

Задатак 1. Кутије

Продавац Мика је веома популаран у свом граду, тако да мештани купују већину намирница у његовој продавници. Он сваког јутра набавља огромне количине робе спаковане у кутије, које празни и тиме робом снабдева свој локал. Након тога, испражњене кутије оставља иза продавнице правећи тако велики низ кутија који се сваког дана повећава. Међутим, због велике најезде купаца, он никако није имао времена да кутије сложи, па је крајем месеца одлучио да узме слободан дан и све кутије унесе у стари магацин. Ипак, приметио је да је његов магацин премали да би све кутије могле да стану у њега, па се досетио начина да колико-толико смањи њихов број. Будући да су све његове кутије поређане у низ, одлучио је да што више може кутија спакује једну у другу. Додатно, Мика је одлучио да постави следеће услове при њиховом паковању:

- Уколико је кутија A спакована у кутију B, онда се кутија A мора обавезно налазити у почетном низу пре кутије B.
- Свака кутија се може произвољно ротирати.
- Две стране сваке кутије морају бити хоризонталне, односно паралелне са подом.

Помозите Мики да одреди најдужи низ кутија које се могу спаковати једна у другу.

Улаз: Са стандардног улаза се у првом реду учитава цео број n, $n \leq 20000$. Након тога се у $(i+1)$-ом реду учитавају три природна броја која представљају дужине страница i-те кутије, $1 \leq i \leq n$. Дужине страница нису веће од 10000.

Излаз: На прву и једину линију стандардног излаза потребно је исписати тражени максималан број кутија.

Пример:

Улаз:
```
4
10 20 5
12 30 7
5 5 4 1 1 3
```

Излаз:
```
2
```

Објашњење: Прва кутија се може спаковати у другу, док се у трећу и четврту кутију не може спаковати ниједна претходна. Закључујемо да је најдужи низ од две кутије које се могу спаковати једна у другу.

Задатак сачувати као *kutije.cpp* или *kutije.java*. Временско ограничење износи 1 секунд.

Решење: У задатку је потребно пронаћи најдужи растући подниз (не обавезно суседних) кутија који се могу спаковати једна у другу. Налажење најдужег растућег подниза је познати проблем динамичког програмирања и може се решити у временској сложености $O(n \log n)$ (видети четврти проблем овог туторијала о динамичком програмирању). Потребно је још одредити услов под којим се једна кутија може сместити у другу. Како је у задатку наглашено да су две странице сваке кутије паралелне са подом, довољно је размотрити случај под којим се један правоугаоник може уписати у други, а затим извршити проверу за све парове одговарајућих страница кутија. Нека су странице првог правоугаоника означене са a и b, а дијагонала са d. Слично, нека су странице другог правоугаоника A и B, дијагонала D и нека важи $a \leq b$ и $A \leq B$. Ако је $a > A$, први правоугаоник се не може уписати у други. Иначе, ако је $a \leq A$ и $b \leq B$, први правоугаоник може да се упише. Ако је $a \leq A$, $b > B$ и $d > D$, како је дијагонала првог правоугаоника већа од дијагонале другог, он се не може уписати. Остаје још да се размотри случај за који је $a \leq A$, $b > B$ и $d \leq D$. Уочимо круг пречника d са центром у пресеку дијагонала другог правоугаоника. Он сече све странице другог правоуганика. Уколико је најмање растојање између пресечних тачки веће од a, први правоугаоник се може уписати, а иначе не може стати у други правоугаоник.

Задатак 2. Војници

Капетан Лукић има чету од n војника различите висине, којом командује већ дуги низ година. Он је веома строг официр, па његови војници, поред тога што увек морају бити спремни да изврше његова наређења без грешке, на његов захтев они се морају постројити у врсту по висини, почев од најнижег ка највишем. Једног јутра, након што је постројио своје војнике, капетан Лукић је отишао до продавнице како би своје војнике наградио сладоледом за њихову послушност. Убрзо пошто је отишао, војници су се опустили и запричали. Међутим, након одређеног времена, када је капетан Лукић већ требало да стигне, приметили су да су неки од њих несвесно заменили места. Како се капетан може вратити сваког тренутка, њима је потребно да утврде начин на који би могли да се врате на првобитне позиције, али у што мањем броју замена. Сваком заменом само два суседна војника могу променити места. Помозите војницима да се што пре врате у првобитан положај и тиме не нљуте њиховог капетана.

Улаз: Са стандардног улаза се у првом реду учитава цео број n, $n \leq 200000$. Након тога се у $(i+1)$-ом реду учитава ненегативан цео број који представља висину i-тог војника, $1 \leq i \leq n$. Подразумева се да су сви војници различите висине и да висина ниједног од њих не прелази 10^6 (за мерење висине војника се користи посебна мерна јединица коју је измислио капетан Лукић).

Излаз: На прву и једину линију стандардног излаза потребно је исписати минималан број замена војника на суседним местима, како би поново били поређани од најнижег ка највишем.

Пример:

Улаз:
```
5
182
185
181
184
183
```

Излаз:
```
5
```

Објашњење: Најмањи број инверзија за сортирање низа је 5. Премештања се могу одвијати на следећи начин: $(182, 185, 181, 184, 183)$, $(182, 181, 185, 184, 183)$, $(181, 182, 185, 184, 183)$, $(181, 182, 184, 185, 183)$, $(181, 182, 184, 183, 185)$, $(181, 182, 183, 184, 185)$.

Задатак сачувати као *vojnici.cpp* или *vojnici.java*. Временско ограничење износи 1 секунд.

Решење: У задатку је потребно одредити минималан број инверзија суседних елемената потребних за сортирање низа. За ово ћемо искористити идеју *MergeSort* алгоритма, чија је временска сложеност $O(n \log n)$. Наиме, у сваком кораку алгоритма, низ се дели на два подниза приближно једнаких дужина, затим се та два подниза сортирају, да би се на крају њихови елементи распоредили тако да се добије сортирани низ. Број замена је тада једнак збиру броја замена за сортирање левог и десног подниза и броја замена за распоређивање елемената. Прва два параметра се одређују рекурзивно, а за последњи је довољно посматрати колико је корака потребно сваком елементу десног подниза да би се нашао на одговарајућем месту у левом поднизу. Укупан број корака по сваком таквом елементу представља број замена за распоређивање елемената.

Задатак 3. Хотели

Будући да је последњег лета у држави Андамантији туризам нагло порастао, њен председник је одлучио да изгради што је више могуће модерних хотела, како би задржао туристе. У Андамантији се налази n градова, а у сваком граду је довољно изградити по највише један хотел. (До тог тренутка у Андамантији се није налазио ниједан модеран хотел, а ниједан туриста није желео да одседа у старим хотелима.) Будући да је терен веома неприступачан, туриста који је смештен у једном граду може само да посети оближња места која су на удаљености не већој од r, док је остала места готово немогуће посетити. У буџету има новца на располагању за изградњу навише k хотела, $k \leq n$. Ваш задатак је да одредите колико је највише градова могуће посетити.

Улаз: Са стандардног улаза се у првој линији учитавају цели бројеви n, k и r, при чему је $1 \leq n \leq 50$, $1 \leq k \leq n$ и $0 \leq r \leq 100$. У сваком од преосталих n редова се налазе два ненегативна цела броја раздвојена размаком, који представљају редом x и y координату одговарајућег града. Вредности координата нису веће

од 1000.

Излаз: На прву и једину линију стандардног излаза потребно је исписати укупан максималан број градова које туристи могу да посете. Град може бити посећен уколико постоји бар један хотел који је од њега на растојању мањем или једнаком r.

Пример:

Улаз:
```
3 1 2
1 2
2 2
10 10
```

Излаз:
```
2
```

Објашњење: Уколико се хотел изгради у граду са координатама $(1, 2)$ или $(2, 1)$, у оба случаја ће прва два града бити захваћена, док се изградњом хотела у граду са координатама $(10, 10)$ може посетити једино град у ком је хотел изграђен.

Задатак сачувати као *hoteli.cpp* или *hoteli.java*. Временско ограничење износи 3 секунде.

Решење: Дати проблем је NP-тежак, па је пожељно користити хеуристички приступ за његово решавање. Овде ће бити приказано решење засновано на локалном претраживању. Наиме, нека једно решење проблема представља фиксиран скуп од k хотела изграђених у различитим градовима. Јасно је да се тада једноставно у $O(n^2)$ може одредити који се хотел може посетити, узимајући у обзир само оне хотеле који се налазе на растојању не већем од r од било ког изграђеног хотела. На почетку се скуп од k изграђених хотела одабере произвољно. Затим се одговарајући број пута (овде је довољно 1000 понављања) понавља следећи поступак:

- Изабере се произвољан хотел међу изграђених k хотела и означи се као неизграђен.
- Изабере се произвољан хотел међу преосталих $n - k$ хотела и означи се као изграђен.
- На тај начин се генерише ново решење од k изграђених хотела.
- Уколико је број захваћених хотела већи од тренутног најбољег, ново решење се узима као полазно за следећу итерацију.

Резултат представља вредност последњег решења, добијеног након 1000 итерација. Сложеност је $O(kn^2)$, где је k укупан број итерација. Алгоритам се додатно може убрзати тако што се ефикасније имплементира прелазак из тренутног у ново решење. Прецизније, ново решење се добија од претходног једноставном заменом два хотела, тако да је притом довољно размотрити ново стање само оних хотела који су на растојању не већем од r од ова два хотела. За сваки фиксиран хотел, скуп његових оближњих хотела се на почетку може иницијализовати.

Задатак 4. Корен

Јоца је за свој пети рођендан добио на поклон дигитрон са напредним функцијама. Временом су му све функције дигитрона досадиле, осим $\lfloor\sqrt{n}\rfloor$, која израчунава највећи цео број који није већи од $\sqrt{n}$. Међутим, како његов дигитрон има места за само 10 цифара, убрзо му је и та функција досадила, па је пожелео да израчуна вредност $\lfloor\sqrt{n}\rfloor$ за много већи аргумент n.

Улаз: У првој линији стандардног улаза се учитава ненегативан цео број n који може имати до 500 цифара.

Излаз: На стандардни излаз је потребно исписати вредност $\lfloor\sqrt{n}\rfloor$.

Пример:

Улаз:
```
84
```

Излаз:
```
9
```

Задатак сачувати као *koren.cpp* или *koren.java*. Временско ограничење износи 3 секунде.

Решење: Како је n велики број, није могуће директно одређивање његовог корена. Овде ће бити приказано једноставно решење засновано на бинарној претрази чија је сложеност $O(\log^3 k)$, где је k број цифара од n. Нека је $p = n/2$ почетни кандидат за решење. Ако је $p^2 \geq n$ и $(p+1)^2 < n$, p је тражено решење. Иначе, ако је $p^2 < n$, довољно је разматрати број $p/2$ као новог кандидата за решење, а ако је $(p+1)^2 > n$, за новог кандидата се узима вредност $(n+p)/2$. Претрага се зауставља када се пронађе одговарајуће решење, које задовољава почетни услов. Додатно, потребно је имплементирати множење великих бројева и дељење великих бројева бројем 2. (У Јави се може искористити класа *BigInteger* која има уграђене функције за аритметичке операције са великим бројевима.)

Задатак 5. Низ

Млади математичар је добио задатак да израчуна вредност n-тог елемента рекурентног низа. Међутим, након краћег размишљања, шватио је да низ може да буде веома дугачак и да би му за израчунавање ипак био потребан рачунар. Због тога је замолио вас да му помогнете у томе.

Низ $(x_n)_{n\in\mathbb{N}_0}$ је дефинисан са прва четири члана $x_i = a_i$, $i \in \{0,1,2,3\}$ и рекурентном релацијом

$$x_n = b_0 x_{n-1} + b_1 x_{n-2} + b_2 x_{n-3} + b_3 x_{n-4}, \quad n \geq 4,$$

при чему су a_i и b_i, $i \in \{0,1,2,3\}$ ненегативни цели бројеви не већи од 1000.

Улаз: У првој линији стандардног улаза се налазе бројеви a_0, a_1, a_2 и a_3 раздвојени размаком. Слично, у другом реду улаза су вредности b_0, b_1, b_2 и b_3. У трећој линији се учитава индекс n, $n \leq 2 \cdot 10^9$.

Излаз: На стандардни излаз је потребно исписати ненегативан цео број који представља вредност n-тог елемента низа $(x_n)_{n\in\mathbb{N}_0}$. Решење исписати по модулу 10000.

Пример:

Улаз:
```
1 1 2 3
1 1 1 1
6
```

Излаз:
```
25
```

Задатак сачувати као *niz.cpp* или *niz.java*. Временско ограничење износи 0.1 секунд.

Решење: Ако је $n < 4$, решење је вредност a_n. Нека је $n \geq 4$. Приметимо да је

$$\begin{bmatrix} x_4 \\ x_3 \\ x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix},$$

као и

$$\begin{bmatrix} x_5 \\ x_4 \\ x_3 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_4 \\ x_3 \\ x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^2 \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}.$$

У последњем изразу се квадрирање матрице извршава по модулу 10000. Одавде следи да се једноставном индукцијом може показати да је

$$\begin{bmatrix} x_n \\ x_{n-1} \\ x_{n-2} \\ x_{n-3} \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^{n-3} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}.$$

Одавде следи да се n-ти члан низа може пронаћи налажењем $(n-3)$-ег степена матрице димензија 4×4. За ово се може искористити модификација познатог алгоритма за степеновање временске сложености $O(\log n)$.

Задатак 6. Згодна матрица

За реалну матрицу A димензија $m \times n$ дефинисану са

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

кажемо да је *згодна* ако су испуњена следећа два услова:

- $m, n \geq 2$,
- $a_{11} + a_{mn} \leq a_{1n} + a_{m1}$.

За матрицу кажемо да је *баш згодна* ако је свака њена подматрица димензија $p \times q$, $p, q \geq 2$ згодна. Ваш задатак је да, за дату матрицу A одредите њену највећу подматрицу која је баш згодна. Подматрица матрице A се може састојати само од узастопних врста, односно колона.

Улаз: У првом реду улаза налазе се природни бројеви m и n ($2 \leq m, n \leq 1000$) који представљају димензије матрице. У сваком од наредних m редова налази се n бројева који представљају елементе m-тог реда. Елементи матрице су цели бројеви из интервала $[10^6, 10^6]$.

Излаз: На једини ред излаза треба исписати максимални број елемената које садржи нека баш згодна подматрица улазне матрице. Ако не постоји ниједна таква подматрица, исписати 0.

Пример:

Улаз:
```
3 3
1 3 1
2 1 2
1 1 1
```

Излаз:
```
4
```

Задатак сачувати као *zgodnamatrica.cpp* или *zgodnamatrica.java*. Временско ограничење износи 1 секунд.

Решење: Математичком индукцијом се може показати да је матрица баш згодна ако и само ако је свака њена подматрица димензије 2×2 згодна. Имајући ово у виду, може се конструисати нова матрица $B = [b_{ij}]$ димензија $(n-1) \times (n-1)$ на следећи начин:

$$b_{ij} = \begin{cases} 1, & \text{ако је } a_{ij} + a_{i+1,j+1} \leq a_{i,j+1} + a_{i+1,j}, \\ 0, & \text{иначе.} \end{cases}$$

Сада се задатак своди на проблем налажења правоугаоника највеће површине који се састоји само од јединица у матрици B. Ово је познати алгоритам динамичког програмирања који се може решити у $O(n^2)$. За детаље о поменутом алгоритму, видети овај туторијал.

Задатак 7. Путовање

Ана и Марко су одлучили да крену на путовање и да притом њихова маршута буде што дужа. Будући да им се укуси не подударају у потпуности, они су одлучили да независно једно од другог саставе листе жеља, а затим их упореде и да на основу тога саставе своју коначну маршуту. Притом свако од њих може пожелети и да више пута посети неки град, али свако жели да се придржава редоследа градова са своје листе жеља. Помозите Ани и Марку да се определе која је маршута најпогоднија за њих, тако што ћете израчунати укупан број маршута максималне дужине које њих двоје могу да прођу.

Улаз: У првом реду стандардног улаза налази се Анина, а у другом Маркова маршута. Свака маршута је предсављена ниском од највише 80 карактера, који су представљени малим словима енглеске абецеде. Притом се неки карактери могу понављати. Сваки карактер представља одређени град.

Излаз: На једини ред стандардног излаза треба исписати укупан број маршута максималне дужине. Гарантује се да ће он бити мањи или једнак 1000.

Пример:

Улаз:
```
абцабцаа
ацбацба
```

Излаз:
```
7
```

Објашњење: Све маршуте максималне дужине које Ана и Марко могу посетити су: абаба, абаца, абцба, ацбца, ацаба, ацаца и ацбаа.

Задатак сачувати као *putovanje.cpp* или *putovanje.java*. Временско ограничење износи 1.5 секунди.

Решење: У овом задатку је потребно одредити број различитих најдужих заједничких подстрингова за два фиксирана стринга $a_1a_2\dots a_n$ и $b_1b_2\dots b_m$. Налажење најдужег заједничког подниза је стандардни алгоритам динамичког програмирања. Нека је c_{ij} дужина најдужег заједничког подстринга за низове $a_1\dots a_i$ и $b_1\dots b_j$. Тада је

$$c_{ij} = \begin{cases} 0, & \text{ако је } i=0 \text{ или } j=0, \\ c_{i-1,j-1}+1, & \text{ако је } i,j>0 \text{ и } a_i = b_j, \\ \max\{c_{i-1,j}, c_{i,j-1}\}, & \text{ако је } i,j>0 \text{ и } a_i \neq b_j. \end{cases}$$

Дужина најдужег заједничког подстринга је вредност c_{nm}. Сада се тражени број најдужих подстрингова може одредити помоћу следећег псеудокода:

```
NZP(i, j, niz)
S = 0
if i = 0 or j = 0 then
   S = S + 1
   return S
end if
if a_i = b_j then
   Dodati a_i na pochetak niza niz
   NZP(i − 1, j − 1, niz)
   return S
end if
if c_{i−1,j} = c_ij then
   NZP(i − 1, j, niz)
end if
if c_{i,j−1} = c_ij then
   NZP(i, j − 1, niz)
end if
```

Задатак 8. Тутанкамон

Чувени цар Тутанкамон је владао Египтом десет година, од 1333. до 1323. године пре нове ере. За време своје владавине имао је један веома необичан начин на који је кажњавао непослушне робове. Најпре би поставио n стубова који се налазе дуж праве линије. Роба би поставио поред њих, а затим би узео ланац, завезао један крај ланца за роба, затим око стубова обавијао ланац на произвољан начин и на крају други крај ланца такође причврстио за роба. На тај начин, робу је било готово немогуће да побегне и његово кретање је било ограничено на околину око стубова. Међутим, његови сународници, који су и сами били робови, али су били послушни, ноћу би се дошетали и приметили да је ланац доста чврст и да не може да се откине, али да су постављени стубови од дрвета и да могу да се исеку, при чему онда роб може заједно са ланцом на себи да побегне, јер стубови више не представљају препреку. Међутим, како у то време нису постојале моторне тестере, процес сечења дрвета је трајао нешто дуже, па је због тога потребно да исеку што мањи број стабала, довољан да помогну непослушном робу да побегне.

Улаз: Са стандардног улаза се у првој линији учитавају четири цела броја, n, m, x_r и y_r. Због једноставности, ланац се сматра полигоном од m темена, који у општем случају не мора бити конвексан и може имати самопресецања. Почетно и крајње теме полигона је положај роба, тј. (x_r, y_r). Од другог до $(n+1)$-ог реда се учитавају редом координате стубова. Притом су сви стубови поређани у вертикалну линију, тј. имају исту x координату. Гледано одозго, роб се налази десно од стубова (вредност x координате му је већа).

У последњих $m+1$ редова се налазе координате темен полигона. Прво и последње теме полигона је управо (x_r, y_r). Важи да је $1 \leq n \leq 10$, $3 \leq m \leq 10000$, а све координате су ненегативни реални бројеви не већи од 10000.

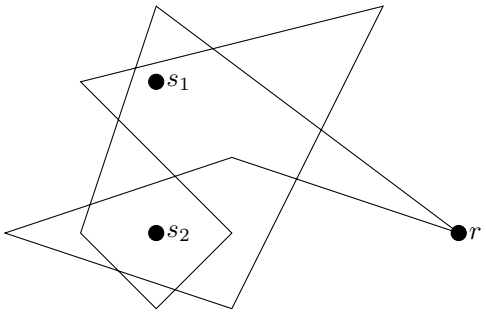
Излаз: На прву и једину линију стандардног излаза треба исписати тражени минимални број стубова који треба пресећи.

Пример:

Улаз:
```
2 10 6 1
2 3
2 1
6 1
2 4
1 1
2 0
3 1
1 3
5 4
3 0
0 1
3 2
6 1
```

Излаз:
```
1
```

Објашњење: Положај стубова, роба и ланца је приказан на слици. Уколико се склони било који од два стуба, роб ће бити у могућности да побегне.



Задатак сачувати као *tutankamon.cpp* или *tutankamon.java*. Временско ограничење износи 3 секунде.

Решење: Приметимо најпре да, будући да се сви стубови налазе у вертикалној линији, ма какав се облик полигона налазио лево или десно од вертикалне линије, није од значаја. Једини делови полигоне линије који су овде од интереса су они који секу линију у којој се налазе стубови. Посматрајмо сада низ тачака, гледајући одозго према доле, који се налази на вертикалној линији стубова, а чији су елементи пресечне тачке. Дати низ тачака има исту x координату, а у зависности од y координате сваке тачке, може се конструисати стринг, где сваки карактер репрезентује одређену тачку у низу. Притом су све пресечне тачке које се налазе између свака два фиксирана стуба означене истим словом енглеске абецеде. Уколико се у новодобијеном стрингу појаве два иста суседна елемента, они се бришу. Овај поступак се понавља док стринг не постане празан или уколико скраћивање више није могуће. Роб може да побегне ако и само ако је стринг постао празан након скраћивања суседних елемената. Описани алгоритам је довољно проверити за сваку фиксирану комбинацију стубова. Како стубова има n, број комбинација износи 2^n, па је укупна временска сложеност $O(2^n m)$.