ПРОБЛЕМ Пљачка века

Тројица окорелих криминалаца су извели пљачку века у далекој земљи Бајтовији, на планини Велебајт. Они су улетели у рудник и из њега изнели N још увек необрађених дијаманата. На крају успешне акције, њих тројица треба да поделе плен. Пошто су то криминалци, то је врло компликована процедура у којој свако жели да добије што више. Проблем је што свако на свој начин процењује свој удео у целој акцији, и не само то, свако на свој начин процењује вредност дијаманата које су украли. Да би лакше поделили плен, они су позвали програмере са такмичења. Морате бити врло пажљиви приликом поделе. Сваки дијамант треба доделити тачно једном од лопова. Потребно је направити поделу тако да сваки од лопова буде задовољан. Лопов ће бити задовољан ако сума процењених вредности дијаманата који су додељени њему није мања од онога што он мисли да заслужује, наравно, све према његовој процени. Примера ради, уколико један лопов процењује суму вредности дијаманата на 10 милиона, и ако сматра да је његов удео 37%, онда он неће бити задовољан са поделом у којој дијаманти које он добија по његовој процени вреде мање од 3.7 милиона. Као награду за добро одрађену поделу, сваки од криминалаца исплатиће у доларима 5% суме вредности дијаманата коју је он добио, према својој процени. Оно што не желимо, то су незадовољни криминалци, тако да је потребно направити поделу тако да се нико не осећа оштећеним, а потом направити такву поделу да програмерски хонорар буде што већи.

УЛАЗ. (Улазни подаци се налазе у датотеци `pljacka.in`) У првом реду се налази природан број N, укупан број дијаманата. У наредном реду се налазе 3 природна броја A, B и C, раздвојена размаком. Они приказују процењени удео за сваког од лопова. У сваком од наредних N редова, налази се по 3 броја, $A(i)$, $B(i)$ и $C(i)$, у милионима долара, који говоре колико сваки од 3 лопова процењује вредност i-тог дијаманта.

ИЗЛАЗ. (Излазне податке уписати у датотеку `pljacka.out`) У сваки од N редова излаза, треба исписати слово `A`, `B` или `C` у зависности од тога који лопов добија који дијамант. Ако има више решења, штампати било које. Уколико нема решења, у једном реду исписати `ODE GLAVA!`.

Ограничења.

- Укупан број дијаманата: $3 \leq N \leq 25$
- Субјективни удео сваког од лопова (у процентима): $1 \leq A, B, C \leq 100$
- Процењена вредност дијаманата: $1 \leq A(i), B(i), C(i) \leq 40$

Пример 1.

`pljacka.in`
```
5
30 30 60
10 5 5
1 5 10
10 5 10
5 10 5
5 5 5
```

`pljacka.out`
```
A
C
C
B
C
```

Пример 2.

`pljacka.in`
```
3
50 50 50
7 7 7
4 5 6
6 5 4
```

`pljacka.out`
```
ODE GLAVA!
```

ПРОБЛЕМ Бандере

Електричар Миле је најбољи мајстор кога сте икада видели. Разуме се у све, брзо поправља апарате, а пре свега није скуп. Једног дана је отишао у оближњи град Абенишбе да поправи неколико бандера. Када је стигао затекао је огромну мрежу. Бандере су биле разбацане по граду, без икаквог распореда. Миле је решио да преузме ствар у своје руке: повадио је све бандере и жели да их постави у једну линију, тако да је растојање између сваке две суседне бандере једнако $1m$. Наравно, све везе морају да остану на својим местима. Дужина жице коју треба да развуче између две бандере је једнака растојању између њих у метрима (најкраће растојање). Миле жели да постави бандере тако да је је укупна дужина жице коју ће потрошити на повезивање буде што је могуће мања. Како овај проблем није Милетов фах, обратио се вама - младим програмерима, да му помогнете и покажете му распоред бандера, тако да минимизује укупну дужину жице. Када сазна пермутацију бандера, Миле ће зачас да их постави и повеже.

За овај задатак потребно је да предате излазне датотеке за 10 улазних датотока `bandere.01.in, bandere.02.in, ..., bandere.10.in`, који се налазе у архиви на рачунару. Излазне датотеке се памте у формату `bandere.01.out, bandere.02.out, ..., bandere.10.out`, при чему број у називу излазне датотеке одговара броју у називу улазне датотеке.

Улаз У првом реду улазне текстуалне датотеке налазе се два природна броја n и m, који представљају број бандера и број међусобних веза. У свакој од следећих m линије се налазе два природна броја a и b ($1 \leq a, b \leq n$), који представљају по једну везу - бандере са редним бројевима a и b су повезане жицом.

Излаз У првом реду излазне текстуалне датотеке треба записати `# bandere, nn` где уместо `nn` треба записати број тест примера. У другом реду излазне датотеке уписати минималну дужину свих жица. У следећем реду треба исписати једну пермутацију бројева од 1 до n, која представља распоред бандера за дату дужину.

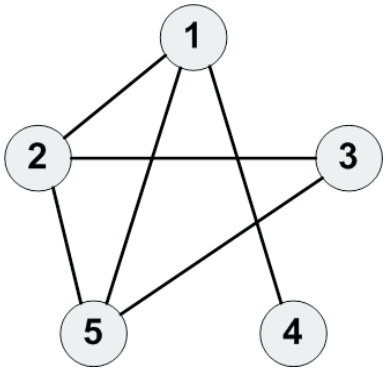
Пример 1.

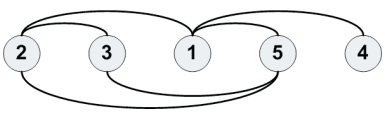
`bandere.00.in`

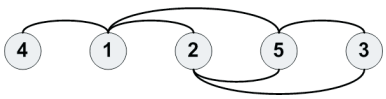
```
5 6
1 2
1 4
1 5
2 3
2 5
3 5
```

`bandere.00.out`

```
# bandere 00
11
2 3 1 5 4
```







Објашњење. На слици је приказана мрежа бандера коју треба "линеаризовати". Решење из примера има укупну дужину свих жица 11: дужина прве жице између бандере 1 и 2 је $2m$, док је дужина треће жице за бандере 1 и 5 једнака $1m$. Сличним расуђивањем добијамо да је укупна дужина жица 2+2+1+1+3+2 = 11. Минимална дужина је једнака 8, на слици десно.

Начин бодовања. Број бодова за сваки тест пример се одређује на следећи начин: нека је дужина жице у вашем решењу VAL, а OPT је минимална дужина међу свим такмичарима, укључујући и комисијска решења. Број бодова које добијате се рачуна по формули (заокруживање се врши на први већи цео број):

$$10^{(3-\frac{VAL}{OPT})/2}$$

Другим речима, ако је ваше решење идентично са најбољим добијате 10 поена. У случају да је решење дупло лошије - добићете 1 поен. За горњи тест пример добијамо $10^{(3-\frac{11}{8})/2} = 6.49$, па је број бодова 7. Ако је дужина коју наведете у излазном фајлу различита од стварне дужине за дату пермутацију добијате 0 поена за тај тест пример.

ПРОБЛЕМ Балегови

Као и његов колега Драганче на прошлом такмичењу, и професор Ђурић је схватио да математика и програмирање нису били исплативи како је мислио. Због тога је одлучио да промени професију и да се убудуће бави трговином. На прошлогодишњем савезном такмичењу сазнали сте да он баш није склон савременим технологијама (свакако се сећате, он још програмира користећи бушене картице), и стога није желео да даје паре за неку савремену вагу, која би, по његовим речима, била много бре неозбиљна. Због тога је одлучио да је сам направи, и то је и учинио. Вага професора Ђурића има један тас на који се могу стављати балегови. Балегови су објекти који имају целобројну тежину. Професор Ђурић је, наравно, својеручно направио и разне балегове, а обезбедио се и у случају да се неки балег загуби: наиме, сваки балег може се заменити неким двама (ни мање ни више) тако да се укупна тежина не промени. Како му се сав овај посао око прављења балегова јако допао, одлучио је да се управо они продају у његовој радњи. Међутим, године га притискају, те је заборавио да на балеговима означи тежине; на сваком је забележио само редне бројеве балегова којима се може заменити. Недуго потом, дошла му је и прва муштерија, која жели да купи комплет балегова чија је укупна тежина једнака нули. С обзиром на то што сви знамо да је муштерија увек у праву, помозите професору Ђурићу да испуни своју дужност као ваљаног трговца. Уколико му не помогнете, не само што ћете остати без бодова, већ ће вас сигурно и бар два пута опсовати.

УЛАЗ. (Улазни подаци се налазе у датотеци `balegovi.in`) У првом реду улазне датотеке налази се природан број n, број различитих балегова које професор Ђурић има (при том сматрати да од сваког поседује довољно примерака). Наредних n редова карактеришу балегове професора Ђурића редом, и то на следећи начин: у првом од њих налазе се два природна броја који представљају редне бројеве балегова којима се први балег може заменити; у следећем реду се налазе два природна броја који представљају редне бројеве балегова којима се други балег може заменити, итд.

ИЗЛАЗ. (Излазне податке уписати у датотеку `balegovi.out`) У првом реду излазне датотеке треба уписати колико је балегова професор Ђурић продао. У наредном реду треба уписати њихове редне бројеве, раздвојене празнином. Уколико професор Ђурић може да задовољи муштерију на више начина, одабрати било који.

Ограничења.

- $1 \leq n \leq 10^6$;
- тежине балегова налазе се у распону од $-2^{1000000}$ до $2^{1000000}$;
- временско ограничење за извршавање програма је 1 sec;
- меморијско ограничење за извршавање програма је 32 MB.

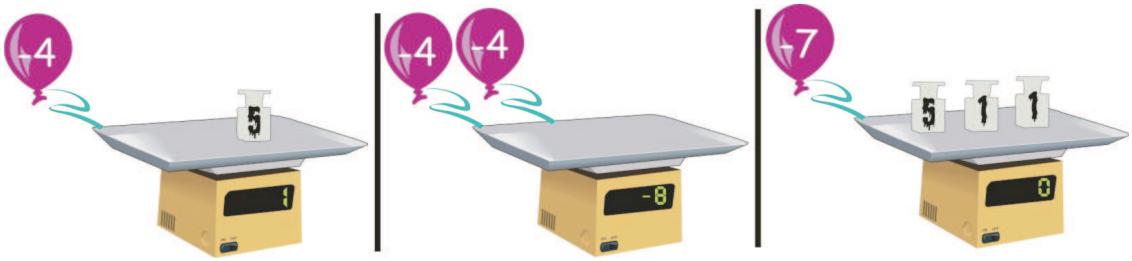
Пример 1.

`balegovi.in`

```
7
4 7
1 1
2 1
3 5
1 6
4 4
2 3
```

`balegovi.out`

```
4
7 5 1 1
```



Објашњење. Професор Ђурић има балегове тежина $1, 2, 3, -4, -7, -8, 5$ (редом). Први балег (чија је тежина 1) може се представити помоћу балегова 4 и 7 (чије су тежине -4 и 5), што је приказано на слици лево. Такође, примера ради, шести балег (чија је тежина -8) може се представити помоћу два једнака балега 4 (чија је тежина -4), што је илустровано сликом у средини. Слично важи за остале балегове. На десној слици видите како је професор Ђурић услужио муштерију: продао му је балегове 7 и 5 и два балега број 1 (чије су тежине $5, -7$ и два пута 1, па им укупан збир јесте једнак нули, као што је тражено). Неопходно је имати у виду да није познато које балегове професор Ђурић има, и зато понуђено решење мора испуњавати тражени услов за сваки могући сет балегова који одговара улазним подацима (погледати и следећи пример).

Пример 2.

`balegovi.in`

```
17
12 12
15 8
7 11
3 3
14 14
9 13
4 2
16 16
14 10
6 1
7 7
13 5
10 1
12 1
4 17
2 2
3 4
```

`balegovi.out`

```
6
5 1 1 13 10 1
```

Објашњење.

И сет

$$(-12, -5, 3, 6, -36, 54, 1, -20, 24, 42, 2, -6, 30, -18, 15, -10, 9)$$

и сет

$$(8, -15, 9, 18, 24, -36, 3, -60, -16, -28, 6, 4, -20, 12, 45, -30, 27)$$

задовољавају почетне услове (постоји још оваквих сетова). Будући да не знамо који од њих професор Ђурић поседује, решења примера ради (4, 12) и (3, 6, 17) нису исправна, јер чине збир нула само у првом, односно другом случају. Тражи се решење које у сваком могућем случају даје збир нула; једно од њих је приказано.

ПРОБЛЕМ Атоми

Пошто је Драганче успешно решио све задатке које му је професор Ђурић задао, математика и програмирање су му постали досадни. Зато је одлучио да се мало за промену посветио хемији. Пошто је сву потребну теорију брзо савладао, решио је да испроба како све то ради у пракси, па је купио справу која може да саставља једињења по жељи корисника. Унутар справе се налази трака која је издељена на редом нумерисана поља од 1 до N, и корисник може да постави атом на неко поље или да га уклони. На једном пољу траке може да се налази само један атом. Уколико се два атома налазе на суседним пољима, они су спојени. Међутим, оно што справа нема је провера стабилности једињења. Наравно, Драганчету то не ствара проблем, јер све што му је потребно да би израчунао стабилност је да зна колико је дугачак најдужи ланац на неком делу траке (ланац чине узастопно повезани атоми). Како број атома може бити веома велик, а Драганче не жели да проведе цео дан у бројању атома, замолио је вас да му помогнете у одређивању дужине најдужег ланца на делу траке који га интересује.

УЛАЗ. (Улазни подаци се налазе у датотеци `atomi.in`) У првом реду улазне датотеке се налазе два броја, N и M, $1 \leq N \leq 1000000$, $1 \leq M \leq 500000$. Број N представља дужину траке, а M укупан број операција. У наредних M линија налазе се описи операција које Драганче врши, а операције могу бити:

- 0 i - $(1 \leq i \leq N)$ уклања атом са поља i (уколико је поље већ празно, ништа се не дешава)
- 1 i - $(1 \leq i \leq N)$ поставља атом на поље i (уколико се атом већ налази на том пољу, ништа се не дешава)
- 2 a b - ($1 \leq a \leq b \leq N$) Драганче се пита колика је дужина најдужег ланца ако се посматрају само атоми који се налазе на интервалу $[a, b]$.

ИЗЛАЗ. (Излазне податке уписати у датотеку `atomi.out`) За свако питање (упит типа "2 a b") у нов ред датотеке треба исписати одговор.

Пример 1.

`atomi.in`	`atomi.out`

`8 5`
`1 3`
`1 5`
`2 3 5`
`1 4`
`2 1 8`
	`1`
	`3`

После друге операције, трака изгледа овако:
`00X0X000`
па је одговор 1.
После четврте операције трака изгледа овако:
`00XXX000`
па је одговор 3.

Пример 2.

`ulaz.in`	`ulaz.out`

`8 8`
`1 1`
`1 2`
`1 3`
`1 4`
`2 2 5`
`0 2`
`1 7`
`2 1 3`
	`3`
	`1`

После четврте операције трака изгледа овако:

XXXX0000
али пошто нас у петом питању занима само интервал од поља 2 до поља 5, одговор је 3.
После седме операције трака изгледа овако :
X0XX00X0
па је одговор на осмо питање 1.

Напомена. У 60% примера ће важити $M \leq 1000$.

ПРОБЛЕМ Игра

Дато је комплетно бинарно стабло дубине H. У листове се могу уписивати вредности 0 или 1. Остали чворови израчунавају своју вредност на основу вредности њихове деце. Чвор узима вредност 0 ако оба његова детета имају вредност 1. У супротном, чвор узима вредност 1. Професор Ђурић и Велики Штрумф играју следећу игру. На почетку игре, вредности чворова нису познате. У сваком потезу, Велики Штрумф пита за вредност неког листа, а професор Ђурић одређује произвољну вредност (0 или 1). Циљ Великог Штрумфа је да што пре утврди вредност у корену стабла, а циљ професора Ђурића је да се одигра што више потеза.

Професор Ђурић је добио информацију да постоји стратегија његове игре која ће натерати Великог Штрумфа да пита за вредности свих листова пре него што са сигурношћу може утврдити вредност корена. Међутим, он није сазнао и саму стратегију. Помозите професору Ђурићу и направите програм који ће играти уместо њега и омогућити му да штрумфује у летњем хладу смејући се Великом Штрумфу.

Ваш програм мора укључити библиотеку `vstrumf.h` у случају `C/C++` програма, односно `vstrumf.tpu` у случају `Pascal` програма, у којима је Велики Штрумф имплементирао своју стратегију. Ове библиотеке садрже следеће функције:

`vstrumf.h`
`int pocniIgru();`
`int pitajVelikogStrumfa(int vrednost);`
`vstrumf.tpu`
`function pocniIgru: integer;`
`function pitajVelikogStrumfa(vrednost: integer): integer;`

Функција `pocniIgru()` враћа вредност H ($1 \leq H \leq 15$), која представља дубину стабла (дужина пута од корена до листа). Ваш програм мора прво позвати ову функцију.

Функција `pitajVelikogStrumfa()` служи за одигравање једног потеза. Она враћа питање Великог Штрумфа, и то је редни број листа за чију вредност је он заинтересован. Листови су нумерисани бројевима од 1 до 2^H слева надесно. Функицција враћа вредност -1 ако је Велики Штрумф одредио вредност у корену, и тада се игра завршава (сви накнадни позиви су ирелевантни). Ви прослеђујете функцији одговор професора Ђурића, и то је вредност коју сте одредили за лист за који је Велики Штрумф претходно питао (у претходном позиву). Та вредност је небитна при првом позиву функције, с обзиром да Велики Штрумф још није поставио ниједно питање.

Типичан сценарио игре је следећи. Ваш програм позива функцију `pocniIgru()` и сазнаје дубину стабла H. Затим, ваш програм позива функцију `pitajVelikogStrumfa()`, при чему прослеђује вредност 0 или 1 произвољно (при првом позиву ова вредност се игнорише), а као повратну информацију добија редни број листа за чију вредност је заинтересован Велики Штрумф. Ваш програм надаље позива функцију `pitajVelikogStrumfa()` све док се као повратна вредност не појави -1. При сваком позиву, прослеђујете одоговор на претходно питање Великог Штрумфа, а добијате његово наредно питање.

Велики Штрумф никада не вара, тј. не завршава игру пре него што је апсолутно сигуран да зна вредност у корену стабла. Такође, он никада неће двапут питати за вредност истог листа. Велики Шрумф ће вам, познат по својој праведности, доделити тачан број поена који сте освојили. Очигледно, број одиграних потеза једнак је броју позива функције `pitajVelikogStrumfa()` до оног позива који враћа вредност -1.

Бодовање

Уколико ваш програм натера Великог Штрумфа да пита за вредности свих листова, освајате максимум поена на том тест примеру. У супротном, можете освојити максимално 60% поена на том примеру, сразмерно броју потеза који је одигран. Дакле, ако је N број свих листова, а игра се заврши после $N-1$ откривених вредности, освајате 60% поена на том тест примеру. Уколико се игра заврши после X откривених вредности, освајате $\frac{X}{N-1}60\%$ поена на тест примеру, при чему се заокруживање врши на први цео број не мањи од израчунате

вредности.

Напомена.

- Временско ограничење по тест примеру је 1s. Игра мора бити завршена у овом временском периоду, укључујући и рад библиотеке.
- Осим комуникације са библиотеком, ваш програм не сме вршити друге улазно/излазне операције, као што је, рецимо, уписивање или читање из датотеке. У супротном, Велики Штрумф вас дисквалификује из игре.
- Ваш програм сме проследити функцији `pitajVelikogStrumfa()` само вредности 0 или 1. У супротном, игра се прекида и Велики Штрумф вам не додељује поене на том тест примеру.
- Уколико ваш програм позове функцију `pitajVelikogStrumfa()` пре позива функције `pocniIgru()` или након завршетка игре (тј. након повратне вредности -1), кршите правила игре и не добијате поене.
- Ако се програм успешно оконча, онда ћете у радном фолдеру моћи да видите излазну датотеку у којој ћете видети колико поена сте освојили у том извршавању програма.

Пример 1

позив	резултат
`pocniIgru()`	2
`pitajVelikogStrumfa(0)`	1
`pitajVelikogStrumfa(1)`	4
`pitajVelikogStrumfa(1)`	2
`pitajVelikogStrumfa(0)`	3
`pitajVelikogStrumfa(0)`	-1

Објашњење. У овом примеру, Велики Штрумф не може одредити вредност у корену стабла пре него што пита и за последњи лист. Програм осваја максимум бодова на овом примеру.

Пример 2

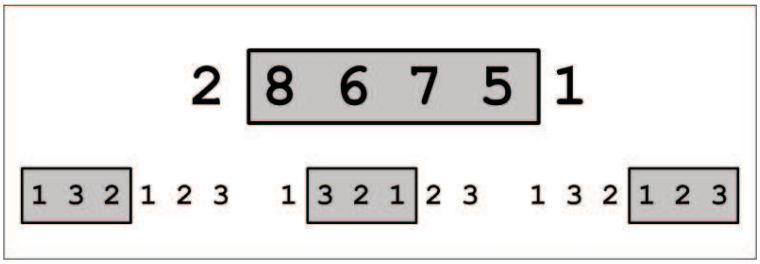
позив	резултат
`pocniIgru()`	3
`pitajVelikogStrumfa(0)`	1
`pitajVelikogStrumfa(0)`	3
`pitajVelikogStrumfa(0)`	-1

Објашњење. У овом примеру, Велики Штрумф са сигурношћу одређује вредност у корену након само две познате вредности. Програм осваја $\frac{2}{7}60\% = 17.14\%$ бодова на овом примеру.

ПРОБЛЕМ Шифра

Сви смо некада имали проблема при памћењу бројева телефона, датума рођења, шифри... Сличан проблем има и најбогатији патак на свету, стари Баја. Наиме, Баја је за свој родјендан од Проке проналазача добио сеф најновије генерације. Како је Прока приметио да Баји полако попушта памћење, шифру му је записао на папиру. Међутим, свестан сталних напада опаке браће Булдог и опасности која вреба уколико се они домогну тог папира, Баја је мало *кодирао* шифру. Ваш задатак је да помогнете Баји да што пре нађе дужину његове шифре као и број могућих комбинација.

Шифра представља најдужи узастопни подниз који чини пермутацију бројева неког сегмента $[a, b]$ (узастопни елементи низа који чине пермутацију).



Текст уз слику: Решење примера 1 и 2.

У првом примеру услов пермутације задовољавају поднизови (2), (8), (8,6,7), (6), (6,7), (6,7,5), (7), (5) и (1), али они немају максималну дужину (која је у том примеру 4).

У другом примеру, три подниза чине могућа решења.

УЛАЗ. (Улазни подаци се налазе у датотеци `sifra.in`) У првом реду датотеке `sifra.in` налази се број n, који представља број елеменената низа. У сваком од следећих n редова, налази се по један број. Дати низ представља бројеве исписане на Бајином папиру.

ИЗЛАЗ. (Излазне податке уписати у датотеку `sifra.out`) У првом и једином реду датотеке `sifra.out` исписати два броја одвојена размаком: дужину најдужег узастопног подзниза који је пермутација и број могућности.

Ограничења.

- $1 \leq n \leq 5000$,
- елементи низа су позитивни цели бројеви мањи од 2^{30},
- временско ограничења за извршавање програма је 1 sec.

Пример 1.

`sifra.in`	`sifra.out`

`6`
`2`
`8`
`6`
`7`
`5`
`1`
	`4 1`

Пример 2.

---

sifra.in
```
6
1
3
2
1
2
3
```

sifra.out
```
3 3
```