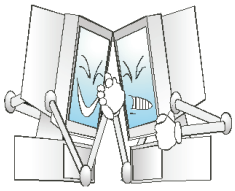


Проблем 2. Мрвице

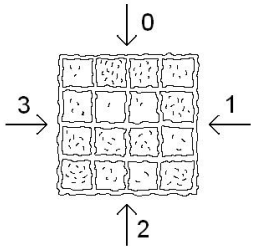
Бака је направила велику торту квадратног облика за своје унуке. Као украс, уздуж и попреко је ставила шлаг преко торте, поделивши је на $n \times n$ квадратних поља једнаке величине. На крају је на свако поље просула одређен број чоколадних мрвица.

Бака сече торту на следећи начин: Прво одабере једну од четири стране торте са које ће да крене да сече, а затим одабере између којих редова, односно колона, ће да сече торту. После тога се не мења ни правац, ни смер сечења док се сечење не заврши. Уколико је ово прво сечење у том правцу и смеру, бака сече од ивице торте. Ако је раније већ секла у том правцу и смеру, онда сече од места завршетка последњег сечења у том правцу и смеру. Након што почне да сече парче, не зауставља се до његове супротне ивице, односно док га потпуно не пресече. Ништа се не сече ако је торта већ пресечена целом дужином у датом правцу.

Пошто сви њени унуци много воле чоколадне мрвице, бака не жели да неко добије ни премало, а ни превише мрвица на парчету.

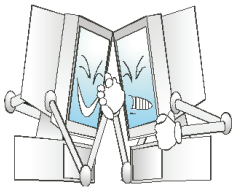
Вама је дато је q упита једног од следећа два типа:

- 1 a b - торта се сече у смеру a (једном од четири као на слици). Ако је хоризонтално сечење, торта се сече између редова b и $b+1$. Ако је вертикално сечење, торта се сече између колона b и $b+1$.
- 2 l h - потребно је одговорити колико парчади тренутно постоји таквих да је број мрвица на парчету у интервалу $[l, h]$.



Улаз. (Улазни подаци се учитавају из датотеке `mrvice.in`.) У првом реду улаза се налазе бројеви n и q ($1 \leqslant n \leqslant 500, 1 \leqslant q \leqslant 100.000$), димензија торте и број упита. У следећих n редова налази се по n бројева, број мрвица на сваком пољу торте. Свако поље ће имати најмање једну, а највише $2 \cdot 10^9$ мрвица. Затим следи q редова који представљају већ описане упите формата t i j, где $t \in \{1, 2\}$. Ако је $t = 1$, онда $0 \leqslant i \leqslant 3$ и $1 \leqslant j \leqslant n-1$. Ако је $t = 2$, онда $1 \leqslant i \leqslant j \leqslant 10^{15}$. Упити се извршавају у редоследу датом на улазу.

Излаз. (Излазни подаци се исписују у датотеку `mrvice.out`.) За сваки упит типа 2 из улаза исписати у нови ред излаза одговор на тај упит (одговоре исписивати у одговарајућем редоследу). Постојаће бар један упит типа 2.



Пример 1.

mrvice.in
```
4 6
3 12 8 5
4 1 2 9
7 6 6 3
10 4 2 8
1 2 1
1 3 3
2 9 15
1 3 3
2 13 14
2 17 100
```

mrvice.out
```
2
2
1
```

Објашњење. За дати пример, сечења су приказана на сликама:

3	12	8	5
4	1	2	9
7	6	6	3
10	4	2	8

После 1. сечења

3	12	8	5
4	1	2	9
7	6	6	3
10	4	2	8

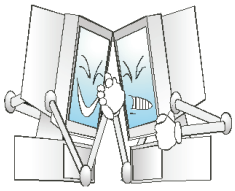
После 2. сечења

3	12	8	5
4	1	2	9
7	6	6	3
10	4	2	8

После 3. сечења

Напомена.

- У 20% тест примера биће $n, q \leqslant 100$.
- У 40% тест примера број упита типа 1 неће прећи 1.000.



Проблем 6. Аутобуси

Дато је m градова, поређаних у низу, и сваки од њих је означен неким бројем (i-ти град је означен бројем a_i, $1 \leqslant a_i \leqslant n$). Наш човечуљак креће из неког града означеног бројем 1, затим мора да оде до неког града означеног бројем 2, итд, да заврши у неком граду означеном бројем n.

Из сваког града на сваких сат времена крећу по два аутобуса, један вози у суседни град с леве стране а други у суседни град са десне стране (осим из првог и последњег града, где постоји само по један аутобус). У свету нашег човечуљка дан траје p сати, и сати су нумерисани од 0 до $p-1$. Познато је да сви аутобуси који у t сати крећу на лево возе l_t сати, а они који крећу на десно возе d_t сати.

Човечуљак у сваком граду може да изабере да крене неким аутобусом или може да чека у том граду произвољан број сати. Потребно је одредити број T, најмањи број сати који је потребан човечуљку да направи тражени обилазак (гарантује се да ће решење постојати). Човечуљак креће у 0 сати из произвољног града означеног бројем 1.

Улаз. (Улазни подаци се учитавају из датотеке `autobusi.in`.) У првом реду улаза се налазе три броја, m, n и p. У следећем реду налази се m бројева, то су вредности a_i, а у следећа два реда по p бројева - у првом од њих се налазе вредности l_i а у другом d_i.

Излаз. (Излазни подаци се исписују у датотеку `autobusi.out`.) У првом реду излаза треба исписати израчунати број T.

Ограничења. $1 \leqslant n, m, p \leqslant 10^5$, $1 \leqslant l_i, d_i \leqslant p$. У 30% тест примера биће $p = 1$, а у 40% $n, m, p \leqslant 1.000$, при чему укупно 60% тест примера задовољава бар један од ова два услова.

Пример 1.

`autobusi.in`
```
6 3 4
1 2 2 3 1 3
1 4 2 4
3 2 4 3
```

`autobusi.out`
```
7
```

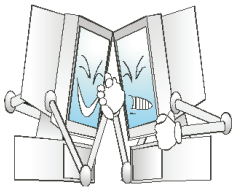
Објашњење. Имамо 6 градова, дан се састоји од 4 сата. Аутобуси који крећу у 0 сати на лево возе $1h$ а на десно $3h$, они који крећу у један сат на лево возе $4h$, а на десно $2h$, итд. Решење је на слици (X означава град у коме смо на почетку одговарајућег сата, а стрелица означава смер у коме се возимо у току тог сата).

	1	2	2		3		1	3	
$0h$						←	X		почињемо из петог града, у $0h$ крећемо аутобусом на лево
$1h$					X				стижемо у четврти град у $1h$, ту чекамо још $1h$
$2h$				←	X				у $2h$ крећемо аутобусом на лево
$3h$				←					
$0h$			X	→					у $0h$(сутрадан) стижемо у трећи град, крећемо аутобусом на десно
$1h$				→					
$2h$				→					
$3h$					X				у $3h$ стижемо у четврти град (након $7h$ путовања)

Пример 2.

`autobusi.in`
```
10 4 6
2 4 4 4 2 3 1 3 1 4
2 5 1 3 6 4
1 3 2 4 5 2
```

`autobusi.out`
```
12
```



Проблем 5. ПалаП

Дат је низ речи (w_i) дужине n. Уз помоћ ових речи могуће је креирати n^2 парова (w_i, w_j) речи, чијом конкатенацијом добијамо реч $w_{i,j}$. Написати програм који испитује колико има парова чијом конкатенацијом добијамо палиндром.

Реч s је палиндром уколико се чита исто и са леве и са десне стране (нпр *aabaa*, *a*, *abcba*).

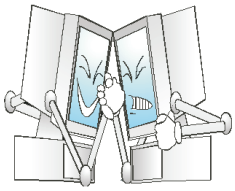
Улаз. (Улазни подаци се учитавају из датотеке `palap.in`.) У првом реду улаза налази се природан број n ($n \leqslant 5.000$) који означава број речи. Сваки од наредних n редова садржи по једну реч. Речи су састављене од малих слова енглеског алфабета и њихова дужина не прелази 500.

Излаз. (Излазни подаци се исписују у датотеку `palap.out`.) У првом и једином реду излаза исписати број парова чијом се конкатенацијом добија палиндром.

Пример 1.

palap.in	palap.out
3 a aa ab	5

Објашњење. Парови који креирају палиндроме су $(1,1), (1,2), (2,1), (2,2)$ и $(3,1)$.



Проблем 3. Асфалт

У једној држави постоји n градова и m двосмерних ауто-путева између неких од њих. Сваки град има неку вредност v из скупа $\{1, 2, \ldots, n\}$, при чему не постоје два града са истом вредношћу. Ове вредности представљају индексе популарности и град са вредношћу n је **главни град**.

Одлучено је да неке од путева у овој држави треба асфалтирати, при чему ће асфлатирање финансирати градови. Сваки град је рекао да ће асфалтирати (тачно један) пут који води од њега до његовог суседа са **највећом** вредношћу (јер је то у интересу сваког града). Међутим, уколико је вредност неког града већа од свих његових суседа он **не асфалтира** ниједан пут.

Ми смо странци и не знамо вредности градова, али имамо мапу и знамо који су путеви асфалтирани. Уколико има **тачно** $n-1$ асфалтираних путева, одредити све градове који могу бити главни. **Гарантује се да постоји бар један такав град, тј. мапа је задата коректно** .

Улаз. (Улазни подаци се учитавају из датотеке `asfalt.in`.) У првом реду улаза налазе се два природна броја, n и m, који представљају, редом, број градова и број ауто-путева ($n \leqslant 20.000$, $m \leqslant 200.000$). У наредних m редова следи опис мапе: по три броја a_i, b_i и c_i у сваком реду који означавају да постоји (неусмерен) пут између градова a_i и b_i, при чему је тај пут асфалтиран ако је $c_i = 1$, односно неасфалтиран ако је $c_i = 0$. Градови су нумерисани од 1 до n и између свака два града постоји највише један ауто-пут.

Излаз. (Излазни подаци се исписују у датотеку `asfalt.out`.) У првом реду излаза исписати број градова који су кандидати за главни град. У следећем реду исписати редне бројеве тих градова у произвољном редоследу, одвојене размаком.

Пример 1.

`asfalt.in`

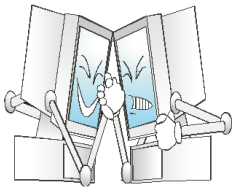
```
4 4
1 2 1
2 3 1
3 4 1
4 1 0
```

`asfalt.out`

```
2
2 3
```

Објашњење. Асфалтирани путеви су: 1-2, 2-3 и 3-4. Уколико би град 1 био главни град онда би сусед града 4 који има највећу вредност био баш град 1 и град 4 би асфалтирао пут 4-1. Међутим, овај пут није асфалтиран па град 1 не може бити главни. Слично и за град 4. Градови 2 и 3 могу бити главни, нпр. за $v(1) = 1$, $v(2) = 4$, $v(3) = 3$ и $v(4) = 2$, град 2 постаје главни и асфалтирање је коректно. Слично и за град 3.

Напомена. У 30% тест примера је $n \leqslant 200$ а у 50% тест примера $n \leqslant 2.000$.



Проблем 1. Игра

Мирко и Славко играју следећу игру: Мирко почиње игру замишљањем неке речи. Међутим, он не каже Славку коју је реч замислио, већ му каже само прво слово те речи. Сада Славко замишља реч која почиње словом које је Мирко рекао и каже Мирку прва два слова своје речи. Мирко сада опет смишља реч која почиње словима које је Славко рекао (нова реч може, али и не мора бити реч коју је на почетку замислио) и каже Славку прва три слова. Ова процедура се наставља све док слова која каже Мирко или Славко не формирају комплетну реч. Играч који први формира комплетну реч је изгубио игру. То значи да иако је Мирко, на пример, замислио реч 'лепота' и каже Славку прва четири слова 'лепо' – он губи, јер је реч 'лепо' такође реч српског језика.

Како би игра била равноправна, оба играча могу користити само речи из унапред дефинисаног речника. Ако оба играча играју оптимално, тада је могуће одредити ко ће на крају игре бити победник. Оптимално играње подразумева да сваки од играча настоји да добије игру у што је могуће мање потеза (знајући да ће и његов противник такође играти оптимално); уколико играч не може да добије игру, онда жели да изгуби што је могуће касније.

Крајњи исход игре је реч из речника коју каже играч који губи. Ако у неком моменту није битно за даљи развој игре које слово ће играч одабрати – он бира увек слово које долази раније у енглеској абецеди (лексикографски најмање). Одредити који играч побеђује у овој игри и реч којом се игра завршава.

Улаз. (Улазни подаци се учитавају из датотеке `igra.in`.) У првом реду се налази природан број n ($1 \leqslant n \leqslant 5000$). У следећих n редова се налази по једна реч састављена од малих слова енглеске абецеде, дужине мање или једнаке од 50, која представља реч из заједничог речника за оба играча.

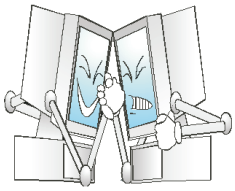
Излаз. (Излазни подаци се исписују у датотеку `igra.out`.) У првом реду исписати који играч побеђује (1 за Мирка и 2 за Славка), а у другом реду исписати реч којом се игра завршава, ако оба играча играју оптимално.

Пример 1.

`igra.in`
```
8
avioni
banana
berza
selo
seobe
kosa
kola
kuca
```

`igra.out`
```
1
kola
```

Објашњење. Ако Мирко каже слово `'b'`, тада ће Славко победити бирањем речи `'berza'` тако што каже слово `'e'`. Слично ако Мирко каже слово `'s'`, сигурно губи. Најзад, Мирко побеђује ако замисли било коју реч на `'a'` или `'k'`. Ако би одабрао реч '`avioni`' победио би у шест потеза, док у осталим случајевима побеђује у четири потеза. Зато Мирко бира слово `'k'`. Затим Славко затим бира слово `'o'`, па онда Мирко слово `'l'` (зато што `'l'` долази пре `'s'` по лексикографском уређењу) и Славко на крају бира слово `'a'`. Дакле, побеђује Мирко (први играч), а тражена реч на крају игре је `'kola'`.



Пример 2.

igra.in	igra.out
3 zzz zzza ttt	2 ttt

Проблем 4. Сорти

Једна продавница компјутерске и сличне опреме продаје 10 различитих производа: *iPad*, *iPod*, *iPhone*, *iTunes*, *AAA* алкалне батерије, *LCD* ТВ, црно-беле ТВ, квалитетне *NetCat* таблет рачунаре, јефтине *mp*4 плејере и сиве боје за штампаче. Ови производи се пакују у идентичне кутије и на свакој од њих стоји налепница са целим бројем између 0 и 9 која означава о ком производу је реч.

Сваког јутра у стовариште продавнице стиже n производа упакованих у кутије и оне се распоређују у низ на једној јако дугачкој полици (у редоследу доласка). Газди продавнице је јако битно да те кутије буду сортиране по налепницама, од најмање до највеће. Зато је ангажовао роботића за сортирање - *Sortija*. *Sorti* имаману: све што може да уради у једном потезу је да у низу **замени било које две кутије које су на растојању тачно** d (растојање између две кутије је број кутија између њих плус 1). Газда је увидео овај недостатак па му је додао још једну опцију: **замена налепница било које две кутије** (нема везе што купци неће добити коректан производ, битно да су они сортирани). Када кутија добије погрешну налепницу, она постаје **фалична**.

Помозите *Sortiju* да сортира кутије али тако да на крају број фаличних кутија буде **најмањи могућ**.

Улаз. (Улазни подаци се учитавају из датотеке `sorti.in`.) У првом реду улазне датотеке налазе се два природна броја, n и d, који представљају, редом, број кутија са производима и константан распон *Sortijevih* руку ($1 \leq d < n \leq 10^6$). У следећем реду налази се стринг дужине n састављен искључиво од цифара 0-9 који представља почетни распоред кутија.

Излаз. (Излазни подаци се исписују у датотеку `sorti.out`.) У првом реду излазне датотеке исписати број x - минималан могућ број фаличних кутија на крају сорта. Затим у следећих x редова исписати по два броја ind_i и num_i који означавају да кутија која се на почетку налазила на позицији ind_i сада има (погрешну) налепницу num_i (није битно од које кутије је добила ту налепницу). Уколико има више решења, штампати било које.

Пример 1.

`sorti.in`	`sorti.out`
`8 5`	`3`
`07349622`	`3 9`
	`4 3`
	`5 4`

Објашњење. Могуће је сортирати тако да само 3 кутије буду фаличне: кутији на позицији 3 (налепница 3) ставимо налепницу 9 са кутије на позицији 5, кутији на позицији 4 ставимо налепницу 3 са кутије на позицији 3 и кутији на позицији 5 - налепницу 4 са кутије на позицији 4. Затим заменимо кутије на позицијама 2 и 7 и на позицијама 3 и 8 (које су на растојању $d = 5$). Није могуће сортирати са мање фаличних кутија.

Напомена. За тачно израчунат број x (из првог реда) добија се 50% поена на том тест примеру. Такође, у 30% тест примера на полици ће бити само *iPad* и *iPod*, тј. улазни стринг ће садржати само цифре 0 и 1.