Задатак Collecting Diamonds

⏳ 3 секунде 💾 4 MB

Налазиште дијаманата је откривено у Родопским планинама. Ради једноставности, претпоставићемо да налазиште има N дворана, које су означене целим бројевима од 0 до $N-1$. Постоји M једносмерних ходника који повезују неке дворане тако да из сваке дворане постоји најмање један ходник који излази из ње. Сваки ходник има одређен број дијаманата који може бити ископан проласком кроз њега. Овај број **се не мења** проласком кроз ходник – остаје исти за наредне проласке.

Могуће је да ходник повезује дворану са самом собом, и да између истих дворана може бити више ходника (могуће у истом смеру). Такође се не гарантује да су дворане повезане; тј. могуће је да постоји пар дворана (x, y) тако да се у y не може стићи из x.

Петар ће проћи кроз K ходника да ископа дијаманте. Изабраће дворану s из које ће кренути, затим ће се преместити у другу дворану пролазећи ходником који почиње у s, и тако даље док не прође кроз тачно K ходника. Приметимо да може понављати дворане и ходнике, и да се број дијаманата које сакупља у ходницима не мења након понављања. Приметимо да ће увек постојати пут да прође кроз K ходника узастопно.

Петар ће изабрати s и путању којом ће се кретати на следећи начин: Прво, жели да максимизује број дијаманата које ће скупити из првог ходника кроз који пролази. Између свих таквих опција, максимизоваће број дијаманата који ће скупити из другог ходника. Ово понавља K пута. Другим речима, Петар жели да изабере лексикографски највећи низ. Пита се колики је укупан број дијаманата који ће сакупити ако на овај начин изабере пут. Помози му да израчуна.

Детаљи имплементације

Треба да имплементираш функцију `calculate_diamonds`:

```
long long int calculate_diamonds(int N, int M, int K,
   std::vector<int> u, std::vector<int> v, std::vector<int> d)
```

- N: број дворана у налазишту дијаманата;
- M: број ходника између дворана;
- K: број ходника кроз који ће Петар проћи;
- u, v, d: вектори M целих бројева, који представљају почетне дворане, крајње дворане, и дијаманте у ходницима.

Ова функција ће бити позвана једном за сваки тест и мора да врати један број – укупан број дијаманата који ће Петар сакупити користећи овакву стратегију.

Ограничења

- $1 \leq N \leq 2\,000$
- $1 \leq M \leq 4\,000$
- $1 \leq K \leq 10^9$
- $0 \leq d[i] \leq N$
- $1 \leq d[i] \leq 10^9$ за свако $0 \leq i < M$
- Гарантује се да постоји најмање један ходник који полази из сваке дворане.
- **Приметите необично мало ограничење меморије од 4 MB.**

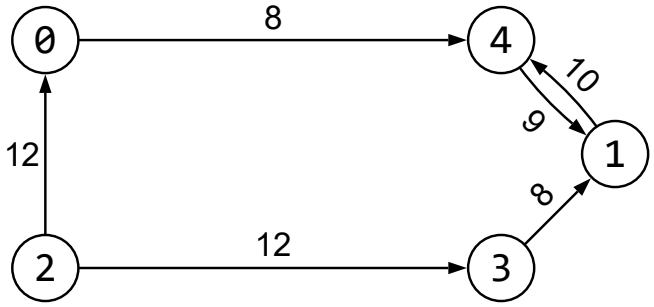
Подзадаци

Подзадатак	Поени	Обавезни подзадаци	N	M	K	Додатна ограничења
0	0	–	–	–	–	Примери.
1	11	0	≤ 10	≤ 20	≤ 10	-
2	10	$0-1$	≤ 100	$\leq 1\,000$	≤ 1000	-
3	26	$0-2$	≤ 100	$\leq 1\,000$	$\leq 10^9$	-
4	11	–	$\leq 2\,000$	$= N$	$\leq 10^9$	Свака дворана има тачно један ходник који почиње из ње и тачно један који се у њој завршава.
5	10	–	$\leq 2\,000$	$\leq 4\,000$	$\leq 10^9$	Сви $d[i]$ су различити.
6	11	–	$\leq 2\,000$	$\leq 4\,000$	$\leq 10^9$	Постоји тачно један $d[i] = 2$ ($0 \leq i < M$) и све друге вредности d су једнаке 1.
7	21	$0-6$	$\leq 2\,000$	$\leq 4\,000$	$\leq 10^9$	-

Пример 1

Размотрите следећи позив и илустрацију, за $N = 5$, $M = 6$, и $K = 4$:

```
calculate_diamonds(5, 6, 4,
   {2, 0, 4, 2, 3, 1}, {0, 4, 1, 3, 1, 4}, {12, 8, 9, 12, 8, 10})
```

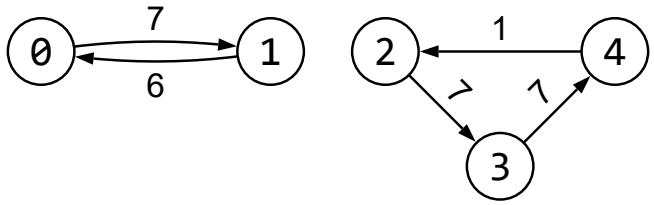


Петар ће изабрати да прође кроз следеће ходнике: $2 \xrightarrow{12} 3 \xrightarrow{8} 1 \xrightarrow{10} 4 \xrightarrow{9} 1$. Укупан број дијаманата који ће сакупити је 39, који треба функција да врати.

Пример 2

Размотрите следећи позив и илустрацију, за $N = 5$, $M = 5$, and $K = 4$:

```
calculate_diamonds(5, 5, 4,
       {0, 1, 2, 3, 4}, {1, 0, 3, 4, 2}, {7, 6, 7, 7, 1})
```



Постоји 5 начина за пролазак кроз 4 ходника:
$(1)\ 0 \xrightarrow{7} 1 \xrightarrow{6} 0 \xrightarrow{7} 1 \xrightarrow{6} 0$;
$(2)\ 1 \xrightarrow{6} 0 \xrightarrow{7} 1 \xrightarrow{6} 0 \xrightarrow{7} 1$;
$(3)\ 2 \xrightarrow{7} 3 \xrightarrow{7} 4 \xrightarrow{1} 2 \xrightarrow{7} 3$;
$(4)\ 3 \xrightarrow{7} 4 \xrightarrow{1} 2 \xrightarrow{7} 3 \xrightarrow{7} 4$;
$(5)\ 4 \xrightarrow{1} 2 \xrightarrow{7} 3 \xrightarrow{7} 4 \xrightarrow{1} 2$.

Начини (2) и (5) не максимизују број дијаманата првог ходника. Од начина (1), (3), и (4) једино начин (3) максимизује број дијаманата из другог ходника тако да је ово најбоља опција за Петра. Приметите да начин (3) не максимизује број дијаманата из трећег ходника, нити максимизује укупан број дијаманата, али то је једини лексикографски највећи низ. Укупан број дијаманата који ће Петар сакупити је 22, што би требала функција да врати.

Пример грејдера

Формат улаза је следећи:

- линија 1: три цела броја – вредности за N, M, и K.
- линија $1 + i$: три цела броја $u[i]$, $v[i]$, $d[i]$ – који представљају ходник који почиње из дворане $u[i]$ и завршава се у дворани $v[i]$ са $d[i]$ дијаманата за ископавање.

Формат излаза је следећи:

- линија 1: један цео број – повратна вредност позване функције.

Задатак Grid

⏳ 0.3 секунде 💾 256 MB

Симона сања о огромном богатству. Понуђено јој је да учествује у игри са великом наградом.

Симона ће бити постављена у поље $(0, 0)$ у мрежи A димензије $N \times M$ која је попуњена позитивним целим бројевима. Мора да дође до поља $(N-1, M-1)$. Да би то урадила, дозвољено јој је да више пута уради следећи потез: помери се из тренутног поља (x, y) у било које друго поље $(x+d, y)$ или $(x, y+d)$, тако да важи $d > 0$. За сваки такав потез, Симона ће бити награђена са $|A_{x,y} - A_{x',y'}| - C$ новчића, где су x', y' њене нове координате, а C је константна цена одређена пре почетка авантуре. Имајте на уму да ако је израз $|A_{x,y} - A_{x',y'}| - C$ негативан број, Симона ће изгубити тај број новчића. Имајте на уму да је могуће завршити игру са негативним бројем новчића.

Помозите Симони да одреди максималан број новчића које може имати на крају игре.

Напомена. $|a| = a$ *ако је* $a \geq 0$ *и* $|a| = -a$ *у супротном.*

Детаљи имплементације

Треба да имплементирате функцију `max_profit`:

```
long long max_profit(int N, int M, int C,
std::vector<std::vector<int>> A)
```

- N, M: димензије мреже;
- C: фиксирана константа за тест пример;
- A: вектор вектора целих бројева димензије $N \times M$, који представљају дводимензионалну мрежу (индексирану по реду, а затим по колони).

Ова функција ће бити позвана једном за сваки тест пример и мора вратити максималан број новчића са којима се игра може завршити.

Ограничења

- $1 \leq N, M$
- $N \cdot M \leq 5 \times 10^5$
- $1 \leq A_{i,j} \leq 10^6$ за $0 \leq i < N$ и $0 \leq j < M$
- $0 \leq C \leq 10^6$

Подзадаци

Подзадаци	Поени	Потребни подзадаци	Додатна ограничења
0	0	–	Пример.
1	9	–	$N = 1, M \leq 200$
2	5	–	$N = 1, A_{i,j} \leq A_{i,j+1}$
3	8	–	$N = 1, C = 0$
4	10	1	$N = 1, M \leq 5 \times 10^4$
5	7	$1 - 4$	$N = 1$
6	15	1	$N, M \leq 200$
7	9	2	$A_{i,j} \leq A_{i+1,j}, A_{i,j+1}$
8	12	3	$C = 0$
9	12	$0 - 1, 4.6$	$N \cdot M \leq 5 \times 10^4$
10	13	$0 - 9$	–

Пример

Погледајмо следећи позив:

```
max_profit(5, 6, 4, {{20, 24, 31, 33, 36, 40},
                     {25, 23, 25, 31, 32, 39},
                     {31, 26, 21, 24, 31, 35},
                     {32, 28, 25, 21, 26, 28},
                     {36, 35, 28, 24, 21, 27}})
```

У овом примеру, оптимална путања је $(0,0) \xrightarrow{7} (0,2) \xrightarrow{2} (1,2) \xrightarrow{10} (1,5) \xrightarrow{8} (4,5)$, а број новчића зарађених овом путањом је $7 + 2 + 10 + 8 = 27$. Ваша функција мора вратити 27.

```
max_profit(2, 2, 100, {{1, 2}, {3, 4}})
```

У овом примеру, функција мора вратити: -197. Имајте на уму да одговор може бити негативан.

Пример грејдера

Улазни формат је следећи:

- line 1: три цела броја - вредности N, M и C.
- lines $2 - (N + 1)$: M цели бројеви – вредности $A_{i,j}$.

Излазни формат је следећи:

- line 1: један цео број – повратна вредност из позива функције.

Задатак Prison

⏳ 2 секунде 💾 1024 MB

Алиса и Боб су неправедно осуђени на затвор са максималним обезбеђењем. Сада морају да испланирају бекство. Да би то урадили, морају бити у могућности да комуницирају што је могуће ефикасније (конкретно, Алиса мора да шаље информације свакодневно Бобу). Међутим, не могу се састати уживо. Једини начин на који могу размењивати информације је путем белешки написаних на салветама. Сваког дана Алиса жели да пошаље Бобу нову информацију - број између 0 и $N-1$. За сваки ручак, Алиса добија три салвете и на свакој салвети записује број између 0 и $M-1$ (могуће је да неки број напише више пута) и оставља их на својој столици. Затим, њихов непријатељ, Чарли, уништава једну од салвета и измеша преостале две. Коначно, Боб проналази преостале две салвете и чита бројеве на њима. Мора тачно декодирати оригинални број који је Алиса желела да му пошаље. На салветама је ограничен простор, тако да је M фиксиран. Међутим, циљ Алисе и Боба је да максимизују проток информација, тако да имају право да изаберу N колико год желе велико. Помозите Алиси и Бобу тако што ћете за сваког од њих применити стратегију, покушавајући да максимузујете број N.

Детаљи имплементације

Пошто је ово комуникациони задатак, ваш програм ће се покретати у два одвојена извршавања (једно за Алису и једно за Боба) која не могу делити податке или комуницирати на било који други начин осим оног описаног овде. Потребно је да имплементирате три функције:

```
int setup(int M);
```

Она ће бити позвана једном на почетку Алисиног извршавања вашег програма и једном на почетку Бобовог извршавања. Додељује јој се вредност M и мора вратити жељени број N. Оба позива функције `setup` морају вратити исти N.

```
std::vector<int> encode(int A);
```

Она имплементира Алисину стратегију. Биће позвана са бројем за енкодирање A ($0 \leq A < N$) и мора вратити три броја W_1, W_2, W_3 ($0 \leq W_i < M$) у које је A енкодиран. Ова функција ће бити позвана укупно T пута – једном дневно (вредности A се могу понављати кроз дане).

```
int decode(int X, int Y);
```

Она имплементира Бобову стратегију. Биће позвана са два од три броја које враћа `encode` у неком редоследом. Мора вратити исту вредност A коју је `encode` примио.

Ова функција ће такође бити позвана T пута – што одговара T позива функције `encode`; биће истим редоследом. Сви позиви функције `encode` ће се догодити пре свих позива функције `decode`.

Ограничења

- $M \leq 4300$
- $T = 5000$

Бодовање

За одређени подзадатак, део поена S које добијете зависи од најмањег N који враћа `setup` на било ком тест примеру у том подзадатку. Такође зависи од N^*, што је циљна вредност N која вам је потребна да бисте добили пун број поена за подзадатак:

- Ако решење даје нетачан одговор на било којем тест примеру, онда је $S = 0$.
- Ако $N \geq N^*$, онда је $S = 1.0$.
- Ако $N < N^*$, онда је $S = \max\left(0.35 \max\left(\frac{\log(N) - 0.985\log(M)}{\log(N^*) - 0.985\log(M)}, 0.0\right)^{0.3} + 0.65\left(\frac{N}{N^*}\right)^{2.4}, 0.01\right)$.

Подзадаци

Подзадатак	Поени	M	N^*
1	10	700	82017
2	10	1100	202217
3	10	1500	375751
4	10	1900	602617
5	10	2300	882817
6	10	2700	1216351
7	10	3100	1603217
8	10	3500	2043417
9	10	3900	2536951
10	10	4300	3083817

Пример

Размотримо следећи пример у ком је $T = 5$. Посматрамо шему енкодирања где Алиса шаље три једнака броја да енкодира 0 или три различита броја да енкодира 1.

Приметимо да Боб може да декодира оригинални број из било која два од три броја које је Алиса послала.

Извршање	Позив функције	Повратна вредност
Alice	`setup(10)`	`2`
Bob	`setup(10)`	`2`
Alice	`encode(0)`	`{5, 5, 5}`
Alice	`encode(1)`	`{8, 3, 7}`
Alice	`encode(1)`	`{0, 3, 1}`
Alice	`encode(0)`	`{7, 7, 7}`
Alice	`encode(1)`	`{6, 2, 0}`
Bob	`decode(5, 5)`	`0`
Bob	`decode(8, 7)`	`1`
Bob	`decode(3, 0)`	`1`
Bob	`decode(7, 7)`	`0`
Bob	`decode(2, 0)`	`1`

Пример грејдера

За пример оцењивача, сви позиви `encode` и `decode` биће у истом извршавању вашег програма. Додатно, функција `setup` ће бити позвана само једном по извршавању (за разлику од два пута као у систему за оцењивање).

Улаз је само један цео број – M. Затим ће исписати N које је ваш `setup` вратио. Затим ће позвати функције `encode` и `decode` (у том редоследу) T пута са насумично генерисаним бројевима од 0 до $N-1$ и насумично генерисаним изборима два од три броја из `encode` који ће се проследити у `decode` (и њихов редослед). Исписаће поруку о грешци ако ваше решење није исправно.

Задатак Navigation

⏳ 3 секунде 💾 512 MB

Дат је **повезан неусмерени једноставан кактус граф**[1] са $N \leq 1000$ чворова и M грана. Чворови су обојени (боје су означене ненегативним целим бројевима од 0 до 1499). На почетку, сви чворови имају боју 0. **Детерминистички робот без меморије**[2] истражује граф крећући се од једног чвора до другог. Мора посетити све чворове барем једном, а затим ће се искључити.

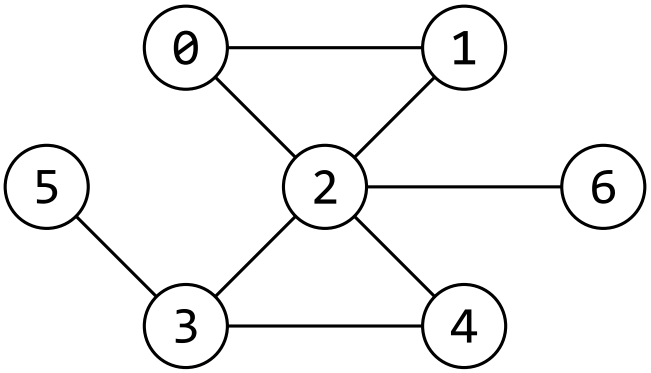
Робот почиње од произвољног чвора у графу. У сваком кораку, робот види боју тренутног чвора и боје свих суседних чворова **у неком фиксном редоследу за тренутни чвор** (тј. поновна посета чвору ће дати роботу исти низ суседних чворова, чак и ако су њихове боје другачије од претходних). Робот може да изврши једну од следеће две радње:

1. Доноси одлуку да се искључи.
2. Бира нову (можда исту) боју за тренутни чвор и бира суседни чвор на који ће се померити. Суседни чвор је јединствено означен индексом од 0 до $D-1$, где је D број суседних чворова.

Ако изабере другу радњу, боја тренутног чвора се промени у другу (или се задржава постојећа) и робот се помера на изабрани суседни чвор. Ово се понавља док се робот не искључи или не достигне ограничење броја итерација. Робот ће победити ако посети све чворове, а затим се искључи у ограниченом броју итерација од $L = 3000$ корака (у супротном, робот губи).

Потребно је да одредите такву стратегију робота која ће решити проблем за било који кактус граф. Поред тога, требало би да минимизујете број различитих боја које ваше решење користи. Боја означена са 0 се увек рачуна као да је коришћена.

[1] *Повезани неусмерени једноставан кактус граф је повезани неусмерени једноставан граф (сваки чвор се може достићи из било ког другог чвора; гране су двосмерне; нема повратних петљи на чворовима или дуплих грана) у коме свака грана припада само једном простом циклусу (прост циклус је циклус који садржи сваки чвор највише једном). Пример је на слици испод.*



[2]*Робот се сматра детерминистичким и без меморије, ако његова радња зависи искључиво од тренутних улаза (тј. не чува информације између два корака) и увек бира исту радњу када му се дају исти улази.*

Детаљи имплементације

Стратегија робота треба да буде имплементирана као следећа функција:

```
std::pair<int, int> navigate(int currColor, std::vector<int> adjColors)
```

Функција узима као параметре боју тренутног чвора и боје свих суседних чворова (у одређеном редоследу). Функција треба да врати пар чији је први елемент нова боја тренутног чвора, а други елемент је индекс суседног чвора на који робот треба да се помери. Ако робот треба да се искључи, функција треба да врати пар $(-1, -1)$.

Позиви функција ће се понављати да би се изабрала акција робота. Пошто је робот детерминистички, ако је функција `navigate` претходно позвана са неким параметрима, неће бити поново позвана са истим параметрима; уместо тога, користиће се вредност коју је функција претходно вратила. Поред тога, сваки тест може да садржи $T \leq 5$ подтестова (различити графови и/или различите почетне позиције) који се могу извршавати истовремено (тј. ваш програм може да прима наизменичне позиве који се односе на различите подтестове). Коначно, позиви функције `navigate` могу се десити у **одвојеним извршавањима** вашег програма (али је могуће да се понекад догоде у оквиру истог извршавања). Укупан број извршавања вашег програма је $P = 100$. Због свега наведеног, ваш програм не би требало да преноси податке између различитих позива.

Ограничења

- $3 \leq N \leq 1000$
- $0 \leq \text{Боја} < 1500$
- $L = 3000$
- $T \leq 5$
- $P = 100$

Бодовање

Део S поена које добијате за подзадатак зависи од C – максималног броја различитих боја које ваше решење користи (укључујући боју 0) на било ком тесту унутар тог подзадатка или било ког другог захтеваног подзадатка:

- Ако ваше решење не прође било који подтест, онда је $S = 0$.
- Ако је $C \leq 4$, онда је $S = 1.0$.
- Ако је $4 < C \leq 8$, онда је $S = 1.0 - 0.6\frac{C-4}{4}$.
- Ако је $8 < C \leq 21$, онда је $S = 0.4\frac{8}{C}$.
- Ако је $C > 21$, онда је $S = 0.15$.

Подзадаци

Подзадатак	Поени	Захтевани подзадаци	N	Додатна ограничења
0	0	–	≤ 300	Пример.
1	6	–	≤ 300	Граф је циклус.[1]
2	7	–	≤ 300	Граф је звезда.[2]
3	9	–	≤ 300	Граф је пут.[3]
4	16	$2-3$	≤ 300	Граф је стабло.[4]
5	27	–	≤ 300	Сви чворови имају највише 3 суседна чвора, а чвор где робот почиње има 1 суседног чвора.
6	28	$0-5$	≤ 300	–
7	7	$0-6$	–	–

[1]Граф који је циклус има гране: $(i, (i+1) \bmod N)$ за $0 \leq i < N$.
[2]Граф који је звезда има гране: $(0, i)$ за $1 \leq i < N$.
[3]Граф који је пут има гране: $(i, i+1)$ за $0 \leq i < N-1$.
[4]Стабло је граф без циклуса.

Пример

Размотримо пример графа са слике у задатку, који има $N = 7$, $M = 8$ и гране $(0, 1)$, $(1, 2)$, $(2, 0)$, $(2, 3)$, $(3, 4)$, $(4, 2)$, $(3, 5)$ и $(2, 6)$. Поред тога, пошто је редослед елемената у листама суседних чворова битан, навешћемо их у следећој табели:

Чвор	Суседни чворови
0	2, 1
1	2, 0
2	0, 3, 4, 6, 1
3	4, 5, 2
4	2, 3
5	3
6	2

Претпоставимо да робот почиње на чвору 5. Онда имамо један могући (неуспешан) низ интеракција:

#	Боје	Чвор	Позив `navigate`	Повратна вредност
1	0, 0, 0, 0, 0, 0, 0	5	`navigate(0, {0})`	`{1, 0}`
2	0, 0, 0, 0, 0, 1, 0	3	`navigate(0, {0, 1, 0})`	`{4, 2}`
3	0, 0, 0, 4, 0, 1, 0	2	`navigate(0, {0, 4, 0, 0, 0})`	`{0, 3}`
4	0, 0, 0, 4, 0, 1, 0	6	[1]`navigate(0, {0})`	`{1, 0}`
5	0, 0, 0, 4, 0, 1, 1	2	`navigate(0, {0, 4, 0, 1, 0})`	`{8, 0}`
6	0, 0, 8, 4, 0, 1, 1	0	`navigate(0, {8, 0})`	`{3, 0}`
7	3, 0, 8, 4, 0, 1, 1	2	`navigate(8, {3, 4, 0, 1, 0})`	`{2, 2}`
8	3, 0, 2, 4, 0, 1, 1	4	`navigate(0, {2, 4})`	`{1, 1}`
9	3, 0, 2, 4, 1, 1, 1	3	`navigate(4, {1, 1, 2})`	`{-1, -1}`

Овде је робот користио укупно 6 различитих боја: 0, 1, 2, 3, 4 и 8 (имајте на уму да се боја 0 рачуна као коришћена чак и ако робот никада није вратио боју 0, јер су сви чворови на почетку обојени бојом 0). Робот је извршио 9 итерација пре него што се искључио. Међутим, није успео, јер се зауставио без посете чвору 1.

[1]Имајте на уму да се позив функције `navigate` у итерацији 4 заправо неће догодити. Разлог је тај што је позив идентичан позиву у итерацији 1, тако да ће грејдер искористити вредност коју је вратила функција у том позиву. Међутим, ово се и даље рачуна као једна итерација робота.

Пример грејдера

Пример грејдера неће извршавати ваш програм више пута, тако да ће сви позиви на `navigate` бити у оквиру једног извршавања вашег програма.

Формат улаза је следећи: Прво се уноси T (број подтестова). Затим, за сваки подтест:

- линија 1: два цела броја – N и M;
- линија $2 + i$ (за $0 \leq i < M$): два цела броја – A_i и B_i, који представљају два чвора повезана граном i ($0 \leq A_i, B_i < N$).

Пример грејдера ће затим исписати број различитих боја које је ваше решење користило и број итерација које су биле потребне пре него што се робот искључио. Ако је ваше решење било неуспешно, исписаће поруку о грешци.

По дифолту ће пример грејдера исписати детаљне информације о томе шта робот види и ради у свакој итерацији. Ово можете онемогућити променом вредности `DEBUG` из `true` у `false`.

Задатак Reactions

⌛ 1 секунд 💾 256 MB

Ники спроводи експерименте о хемијској реактивности. Припремио је N експеримената, који су индексирани од 0 до $N-1$. Треба да изабере експеримент који ће прво извести, а затим ће спровести све експерименте са индексима који су већи или једнаки индексу експеримента којег је првог изабрао. Другим речима, ако одлучи да почне од експеримента са индексом S, извешће експерименте $S, S+1, \ldots, N-1$ овим редоследом.

Пре избора почетног експеримента, он има посуду са раствором. Температура тог раствора је једнака 0 степени. Током i-тог експеримента ($0 \leq i \leq N-1$), он изводи следећа два корака овим редоследом:

1. Мења температуру раствора за дати цео број степени (може се повећати/смањити за произвољан број или остати непромењена);
2. Изводи експеримент и проверава да ли се реакција одвија.

Познато је да се у i-том експерименту температура мења за D_i степени – температура се повећава ако је $D_i > 0$, смањује се ако је $D_i < 0$ или остаје непромењена ако је $D_i = 0$. Реакција у i-том експерименту се одвија само ако је тренутна температура (након промене) већа или једнака T_i. Треба напоменути да промена температуре из првог корака опстаје без обзира на то да ли се реакција одвија или не.

Ники жели да се одвија највећи број реакција како би могао да прикупи што више података. Помозите му тако што ћете израчунати овај број.

Детаљи имплементације

Треба да имплементирате функцију `reactions`:

```
int reactions(int N, std::vector<int> D, std::vector<long long> T)
```

- N: број планираних експеримената;
- D: вектор од N целих бројева, где D_i представља промену температуре за i-ти експеримент;
- T: вектор од N целих бројева, где T_i представља минималну температуру раствора да би се реакција одвила током i-тог експеримента.

Ова функција ће бити позвана једном за сваки тест. Треба да врати максималан број реакција које се могу догодити ако је почетни експеримент изабран најоптималније могуће.

Ограничења

- $1 \leq N \leq 500\,000$
- $-10^9 \leq D_i \leq 10^9$
- $-10^{15} \leq T_i \leq 10^{15}$

Подзадаци

Подзадатак	Поени	Неопходни подзадаци	Додатна ограничења
0	0	–	Примери.
1	15	0	$N \leq 2000$
2	15	0	Постоји највише 20 индекса i за које је $D_i < 0$.
3	20	–	$D_i \leq 0$ за свако $0 \leq i < N$
4	20	0	Одговор је највише 20.
5	30	$0 - 4$	–

Пример 1

Размотримо следећи позив:

```
reaction(5, {1, 1, -3, 1, 1}, {1, 3, 5, 1, 2})
```

Ако Ники одлучи да почне од експеримента са индексом 3, температура раствора ће постати 1 што задовољава ограничење за одвијања те реакције. Током следећег експеримента температура се повећава на 2 и реакција се поново одвија. Пошто не постоји начин да се одвију више од 2 реакције, функција би требало да врати 2.

Пример 2

Размотримо следећи позив:

```
reaction(5, {1, -3, 0, 3, 2}, {0, -2, -1, 0, 3})
```

Функција треба да врати 4 јер ће, почевши од експеримента са индексом 0, Ники видети реакције током експеримената са индексима 0, 1, 3 и 4. Температура почиње од 0 степени и током експеримената температуре су 1, −2, −2, 1 и 3, редом.

Пример грејдера

Формат улаза је следећи:

- линија 1: један цео број – вредност N.
- линија 2: N целих бројева – $D_0, D_1, \ldots, D_{N-1}$.
- линија 3: N целих бројева – $T_0, T_1, \ldots, T_{N-1}$.

Формат излаза је следећи:

- линија 1: један цео број – повратна вредност позива.

Задатак Vacation

⏳ 1.2 sec. 💾 512 MB

Антон и његови пријатељи планирају одмор заједно. Они су већ одабрали локацију; међутим, још увек се нису договорили око датума.

Свих N пријатеља унапред знају датуме када могу да одсуствују са посла. Пријатељ i првобитно је заказао своје слободне дане од дана L_i до дана R_i, укључујући оба. Да би максимизовали време које могу да проведу заједно, сваки пријатељ може да прилагоди своје време померањем раније или касније. Специјално, i-ти пријатељ може да изабере цео број d_i и промени своје слободне дане у интервал $[L_i + d_i, R_i + d_i]$. Позитиван d_i значи да слободни дани почињу касније него што је планирано, негативно d_i значи да почињу раније, а $d_i = 0$ значи да слободни дани почињу по првобитном распореду.

Пријатељи су схватили да се њиховим шефовима неће свидети њихове промене дана. Због тога, помераће само дане тако да важи $|d_0| + |d_1| + \cdots + |d_{N-1}| \leq K$, за неки цео број K.

Помози пријатељима да пронађу максималан број дана **тако да сви** могу да буду заједно на одмору ако оптимално промене слободне дане.

Детаљи имплементације

Треба имплементирати следећу функцију `plan_vacation`:

```
int plan_vacation(int N, std::vector<int> L, std::vector<int> R,
    long long K)
```

- N: број пријатеља
- L: вектор N позитивних целих бројева, од којих сваки означава први слободан дан по првобитном распореду за тог пријатеља;
- R: вектор N позитивних целих бројева, од којих сваки означава последњи слободан дан по првобитном распореду за тог пријатеља;
- K: максимална дозвољена вредност за $|d_0| + |d_1| + \cdots + |d_{N-1}|$.

Ова функција ће бити позвана једном за сваки тест. Треба да врати максималан број дана који пријатељи могу да проведу заједно или 0 ако то уопште није могуће.

Ограничења

- $1 \leq N \leq 500\,000$
- $1 \leq L_i \leq R_i \leq 10^9$
- $0 \leq K \leq 10^{18}$

Подзадаци

Подзадатак	Поени	Захтевани подзадаци	Додатна ограничења
0	0	–	Пример.
1	7	–	$K = 0$
2	11	1	$K \leq 1$
3	6	–	$K = 10^{18}$
4	13	0	$N \leq 10^4$, $L_i \leq 10$, $R_i \leq 10$
5	18	0	$N \leq 10^3$
6	29	$0, 4, 5$	$N \leq 10^5$
7	16	$0 - 6$	–

Пример

Размотримо следећи позив:

```
plan_vacation(3, {1, 5, 2}, {3, 9, 5}, 3)
```

Пријатељи су захтевали следеће интервале слободних дана: $[1, 3]$, $[5, 9]$, $[2, 5]$. Можемо померити слободне дане пријатеља 0 за 2 дана касније и дане пријатеља 1 за 1 дан раније. Онда су интервали слободних дана $[3, 5]$, $[4, 8]$, $[2, 5]$. Тада сви пријатељи имају слободне дане 4 и 5, што као резултат даје 2 заједничка слободна дана. Може се доказати да не можемо добити више за $K = 3$. Функција треба да врати 2.

Пример грејдера

Формат улаза је следећи:

- линија 1: два цела броја – вредности N и K.
- линије од 2 до $N+1$: два цела броја – L_i и R_i.

Формат излаза је следећи:

- линија 1: један цео број – повратна вредност позива.