Лов за съкровища

Ограничение по време: 8 s *Ограничение по памет: 256 MiB*

Търсите отдавна изгубени съкровища в поле с размер $N \times N$, използвайки магически компас. Общо $K \leq 3$ съкровищни сандъка са скрити в различни клетки на полето. Вашата цел е проста: да откриете всички!

Когато поставите компаса в дадена клетка, той ще намери най-кратките пътища до съкровищен сандък, използвайки само движения наляво, нагоре, надясно и надолу (т.е. най-близък по Манхатъново разстояние сандък). След това ще покаже посоката на първата стъпка от пътя. Ако има няколко валидни първи движения, водещи до най-близък сандък (или до няколко от тях), компасът ще върне всички от тях. Ако клетката съдържа съкровищен сандък, компасът ще индикира това.

Всеки път, когато намерите съкровищен сандък, изпразвате съдържанието му, но не премахвате самия сандък – той е твърде тежък. Вашият компас не знае дали даден сандък е пълен или празен. Той ще сочи към най-близък сандък, независимо дали празен или пълен.

Опитайте се да намерите всички съкровищни сандъци с възможно най-малко заявки към компаса.

Задача

Това е интерактивна задача. При всеки тест (т.е. при всяко изпълнение на програмата Ви) тя трябва да реши няколко отделни лова за съкровища. Ще комуникирате с проверяващата система чрез библиотека, предоставена от организаторите. Тази библиотека съдържа следните декларации:

- **void** *NextHunt*(**int** &N, **int** &K) — извикайте тази функция, за да започнете следващия лов за съкровища. Тя ще запише размера на полето в променливата N и броя на съкровищата в K. Ако няма повече ловове за решаване в текущото изпълнение, функцията ще запише -1 в N и K; в такъв случай програмата Ви трябва да терминира с изходен код 0. Обърнете внимание, че можете да извикате тази функция, преди да сте намерили всички съкровища в текущия лов, например ако се стремите към частични точки.

- **enum** { *TREASURE* = 0, *DIR_RIGHT* = 1, *DIR_UP* = 2, *DIR_LEFT* = 4, *DIR_DOWN* = 8 }; — това са константи, използвани в стойностите, които функцията *Query* връща (вижте по-долу).

- **int** *Query*(**int** x, **int** y) — ако клетката с координати (x, y) съдържа съкровищен сандък, функцията връща *TREASURE*. В противен случай, връща сумата на една или повече от константите *DIR_RIGHT*, *DIR_UP*, *DIR_LEFT* и *DIR_DOWN*, за да обозначи кои движения връща компасът, когато е поставен в клетката (x, y). Координатите x и y трябва да бъдат цели числа от 0 до $N-1$. (Забележка: в тази задача координатата y нараства отгоре надолу.)

След като *NextHunt* запише $N = K = -1$, програмата Ви не трябва повече да извиква нито *NextHunt*, нито *Query*. Тя също така не трябва да извиква *Query* преди

първото извикване на *NextHunt*. Ако програмата Ви не спази тези ограничения, или ако направи повече от 1000 заявки в рамките на един лов на съкровища, или ако подаде стойности за x и/или y извън допустимите граници при извикване на *Query*, библиотеката ще прекрати програмата и ще даде резултат run-time-error (RTE) за съответния тест.

Съкровище се счита за намерено, ако сте направили поне една заявка към клетката, която го съдържа. Ако програмата Ви извика *NextHunt* преди да е намерила всички съкровища от текущия лов, това не се счита за грешка, но ще повлияе на резултата Ви (вижте оценяването по-долу).

За да използва библиотеката, програмата Ви трябва да включи хедър файла `treasurehuntlib.h`:

```
#include "treasurehuntlib.h"
```

Можете да изтеглите този хедър файл от тук: `treasurehuntlib.h`.

За Ваша помощ, е предоставена проста реализация на библиотеката тук: `treasurehuntlib-public.cpp`. За да я компилирате заедно с програмата си, просто добавете името ѝ като параметър на компилатора, например:

```
g++ foo.cpp treasurehuntlib-public.cpp
```

ако `foo.cpp` е името на файла, съдържащ Вашето решение.

На оценяващата система ще бъде използвана различна реализация на библиотеката, така че не бива да правите никакви предположения за начина, по който тя работи. Все пак може да приемете, че тя не замърсява глобалното пространство от имена с други декларации освен изброените по-горе (*NextHunt*, *Query* и петте константи).

Вашата програма не трябва да чете от стандартния вход и не трябва да пише в стандартния изход, защото те ще бъдат използвани от нашата реализация на библиотеката на сървъра за оценяване за комуникация с останалата част от средата за оценяване.

Вход

Ограничения

- $1 \leq N \leq 10^6$
- $1 \leq K \leq 3$
- В рамките на едно изпълнение на програмата ще има най-много 100 000 лова на съкровища.
- В рамките на един лов на съкровища можете да направите най-много 1000 заявки.
- Оценяващата система не е адаптивна.

Подзадачи

- Подзадача 1 (10 точки) $K = 1$
- Подзадача 2 (30 точки) $K = 2$
- Подзадача 3 (60 точки) $K = 3$

Оценяване

Една подзадача може да се състои от няколко теста (няколко изпълнения на програмата), а всеки тест може да съдържа няколко лова на съкровища. За целите на оценяването всички ловове на съкровища от дадена подзадача се разглеждат заедно, независимо как са били разпределени между тестовете. За i-тия лов на съкровища ще означим размера на полето с $N_i \times N_i$, броя на съкровищата с K_i, броя на заявките, направени от програмата Ви, с Q_i, а броя на намерените съкровища с F_i. Освен това нека S е общият брой точки, предназначени за тази подзадача. Тогава броят точки, които програмата Ви получава за тази подзадача, е:

- Ако програмата Ви винаги е намерила всички съкровища (т.е. ако $F_i = K_i$ за всички i), точките ѝ зависят от $t_i = \frac{Q_i}{\lceil \log_2 N_i \rceil}$:

$$\frac{S}{2} + \frac{S}{2} \cdot \min_i f(t_i) \text{ точки, където } f(t_i) = \begin{cases} 1, & t_i \leq 11 \\ 1 - (t_i - 11)/9, & 11 \leq t_i \leq 20 \\ 0, & t \geq 20. \end{cases}$$

- Ако програмата Ви не винаги е намерила всички съкровища (т.е. ако съществува i, такова че $F_i < K_i$), тя получава:

$$\frac{S}{2} \cdot \min_i \frac{F_i}{K_i} \text{ точки.}$$

С други думи, получавате половината от точките за намирането на всички съкровища и другата половина — за това да ги намирате с възможно най-малко заявки. За максимален резултат решението Ви трябва да намира всички съкровища с не повече от $11\lceil \log_2 N_i \rceil$ заявки. Между $11\lceil \log_2 N_i \rceil$ и $20\lceil \log_2 N_i \rceil$ заявки резултатът намалява линейно; а ако програмата прави повече от $20\lceil \log_2 N_i \rceil$ заявки, ще получи само първата половина от точките за намирането на всички съкровища.

Ако горните формули доведат до нецелочислен брой точки за подзадачата, резултатът се закръгля до най-близкото цяло число.

Ако програмата Ви получи грешка по време на изпълнение или не спазва описания по-горе протокол за използване на библиотеката, тя ще получи 0 точки за цялата подзадача. За да получите частични точки за намиране само на част от съкровищата, все пак трябва вярно да приключите търсенето чрез извикване на *NextHunt*.

Пример

Извикване	Върната стойност
NextHunt(N, K)	$N = 4$, $K = 1$
Query(2, 0)	$DIR\_DOWN + DIR\_RIGHT = 9$
Query(3, 1)	DIR_DOWN
Query(3, 2)	$TREASURE$
NextHunt(N, K)	$N = -1$, $K = -1$