**Jan Kanty Milczek, Jakub Pachocki**
Treść zadania, Opracowanie

**Jan Kanty Milczek**
Program

**Dostępna pamięć: 24 MB.**

*OI, etap III, dzień pierwszy, 2.04.2014*

# Dookoła świata

*Bajtazar, po wielu latach starań, otrzymał wreszcie upragnioną licencję pilota. Z radości postanowił kupić sobie samolot i oblecieć dookoła całą planetę 3-SATurn (jest to planeta, na której znajduje się Bajtocja). Bajtazar planuje lecieć wzdłuż równika, tak by przelecieć nad każdym jego punktem. Niestety, całą sprawę utrudnia fakt, że samoloty trzeba tankować. Dla każdego samolotu wiadomo, ile co najwyżej kilometrów może on przelecieć, jeśli zaczynał z pełnym bakiem. Paliwo można uzupełnić na dowolnym lotnisku położonym na równiku, co niestety wiąże się z lądowaniem na tym lotnisku.*

*Ponieważ kupno samolotu nie jest łatwą sprawą, Bajtazar prosi Cię o pomoc. Przedstawił Ci propozycje różnych modeli samolotów – mogą one różnić się pojemnością baku. Dla każdego z tych modeli należy wyznaczyć minimalną liczbę lądowań (włącznie z finalnym lądowaniem), które będzie musiał wykonać Bajtazar w celu ukończenia podróży dookoła świata. Dla każdego modelu podróż można zaczynać od dowolnego lotniska.*

## Wejście

*Pierwszy wiersz standardowego wejścia zawiera dwie liczby całkowite $n$ i $s$ ($2 \leqslant n \leqslant 1\,000\,000$, $1 \leqslant s \leqslant 100$) oddzielone pojedynczym odstępem, oznaczające liczbę lotnisk położonych na równiku i liczbę samolotów, jakich kupno rozważa Bajtazar.*

*Drugi wiersz zawiera $n$ dodatnich liczb całkowitych $l_1, l_2, \ldots, l_n$ ($l_1 + l_2 + \ldots + l_n \leqslant 10^9$) pooddzielanych pojedynczymi odstępami, opisujących odległości między kolejnymi lotniskami na równiku. Liczba $l_i$ oznacza, że odległość między $i$-tym a $(i+1)$-szym (lub $n$-tym i pierwszym, jeśli $i = n$) lotniskiem wynosi $l_i$ kilometrów.*

*Trzeci wiersz zawiera $s$ liczb całkowitych $d_1, d_2, \ldots, d_s$ ($1 \leqslant d_i \leqslant l_1 + l_2 + \ldots + l_n$) pooddzielanych pojedynczymi odstępami. Liczba $d_i$ oznacza, że $i$-ty samolot jest w stanie przelecieć $d_i$ kilometrów, zanim będzie musiał lądować i zatankować.*

*W testach wartych 50% punktów zachodzi warunek $n \leqslant 100\,000$, a w podzbiorze tych testów wartym 20% punktów zachodzi dodatkowy warunek $n \leqslant 1000$.*

*W innych testach (nie wymienionych powyżej) wartych 18% punktów zachodzi warunek: $s \leqslant 5$.*

## Wyjście

*Na standardowe wyjście należy wypisać $s$ wierszy: $i$-ty z nich powinien zawierać jedną liczbę całkowitą oznaczającą minimalną liczbę lotów, które trzeba wykonać $i$-tym samolotem, aby oblecieć planetę 3-SATurn dookoła wzdłuż równika, zaczynając z dowolnego lotniska, lub słowo* `NIE`*, jeśli nie da się tym samolotem odbyć podróży.*

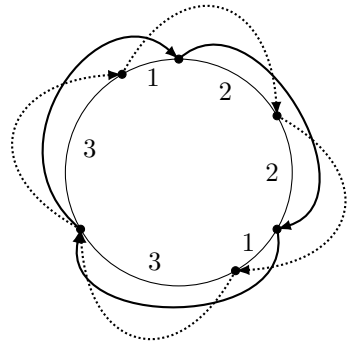
## Przykład

*Dla danych wejściowych:*

```
6 4
2 2 1 3 3 1
3 2 4 11
```

*poprawnym wynikiem jest:*

```
4
NIE
3
2
```




**Wyjaśnienie do przykładu:** *Na rysunku pogrubiona ciągła linia reprezentuje optymalną podróż samolotu o pojemności baku 4, zaś linią przerywaną oznaczono podróż samolotu o pojemności baku 3.*

**Testy „ocen":**

`1ocen`: $n = 8$, $s = 3$, *mały test sprawdzający kilka przypadków brzegowych;*

`2ocen`: $n = 24$, $s = 8$, *test, w którym każde kolejne dwa lotniska są odległe o 1;*

`3ocen`: $n = 1\,000\,000$, $s = 100$, *test maksymalnego rozmiaru.*

# Rozwiązanie

## Co widzimy

Widząc ograniczenia z treści zadania, dochodzimy do wniosku, że oczekuje się od nas obliczenia odpowiedzi dla każdego samolotu w czasie liniowym – jest to słuszny wniosek.

## Pierwsza oczywista obserwacja

Zawsze opłaca się latać w tym samym kierunku. Jeśli jakaś sekwencja lotów w prawo (na wschód) rozwiązuje zadanie, to można ją odwrócić, otrzymując rozwiązanie składające się z lotów w lewo (na zachód). Dla zupełnej jasności dodajmy jeszcze, że nie ma żadnego sensu zawracać. Od tej chwili zakładamy, że wykonujemy loty tylko w prawo.

## Druga oczywista obserwacja

Nie ulega wątpliwości, że istnieje optymalna podróż, w której każdy lot – być może oprócz ostatniego – jest tak długi, jak tylko się da. Rozpatrywanie danego samolotu

warto zacząć od obliczenia dla każdego lotniska, jakie jest najdalsze lotnisko po prawej, na które jesteśmy w stanie dolecieć na jednym baku. Możemy to prosto wykonać tzw. *funkcją gąsienicową* – korzystając z obserwacji, że jeśli z lotniska $a$ najdalej można dolecieć na lotnisko $b$ ($b \neq a$), to z lotniska $a+1$ na pewno można dolecieć na lotnisko $b$, a może i do lotnisk o większych numerach (modulo $n$). Zwiększając $a$ od 1 do $n$, indeks $b$ również tylko zwiększamy i wykonamy nim łącznie najwyżej $2n$ zwiększeń, bo przecież z ostatniego lotniska nie możemy lecieć dalej niż o $n$ lotnisk.

## Rozwiązanie $O(sn \log n)$ w pamięci $O(n \log n)$

Wiedząc, jak daleko można dolecieć jednym lotem, możemy obliczyć w czasie i pamięci $O(n \log n)$, jak daleko można z każdego lotniska dolecieć $2^i$ lotami dla dowolnego $i = 0, 1, \ldots, \lfloor \log n \rfloor$. W tych obliczeniach korzystamy z prostej obserwacji, że aby przemieścić się z danego lotniska o $2^i$ lotów, wystarczy dwukrotnie przemieścić się o $2^{i-1}$ lotów.

Znając dla każdego lotniska „zasięg" $2^0$ lotami, $2^1$ lotami, $2^2$ lotami... możemy dla danego lotniska obliczyć (w czasie $O(\log n)$), ilu lotów potrzeba, aby oblecieć świat. Aby to zrobić, wybieramy największe takie $i$, że $2^i$ lotów *nie* wystarczy do oblecenia świata, i wykonujemy te loty. Następnie sprawdzamy w podobny sposób, ile jeszcze lotów trzeba, aby dokończyć podróż. Zauważmy, że kolejne wybierane $i$ będą coraz mniejsze (zamiast wybrać $i$ i $i$ mogliśmy wybrać $i+1$).

### Problem

Moglibyśmy tak rozwiązać zadanie, gdyby nie bezlitosne ograniczenie pamięci – 24 MB. Powyższe rozwiązanie, zaimplementowane w pliku `doos3.cpp`, zgodnie z sugestią w treści zadania uzyskiwało na zawodach połowę punktów.

## Nieoczywista obserwacja

Obliczmy (w czasie liniowym) wynik dla jakiegoś lotniska (np. lotniska nr 1) – oznaczmy ten wynik $x$. Zauważmy, że jeśli zaczynając z jakiegoś lotniska, potrzebujemy $x$ lotów, to z każdego innego potrzebujemy $x$, $x-1$ albo $x+1$ lotów. Jest tak dlatego, że z dowolnego lotniska możemy wlecieć na jakieś lotnisko należące do (jakiegoś) rozwiązania optymalnego, później wykonać wszystkie loty wchodzące w skład rozwiązania optymalnego poza ostatnim, a na koniec wrócić na lotnisko startowe.

## Rozwiązanie $O(sn \log n)$ w pamięci liniowej

Oznacza to, że musimy jedynie sprawdzić, czy da się z jakiegoś lotniska oblecieć świat w $x-1$ lotów. Możemy w takim razie obliczyć dla każdego lotniska, jak daleko dolecimy z niego $x-1$ lotami. W tym celu przedstawimy $x-1$ jako sumę potęg dwójki i znów zastosujemy pomysł z obliczaniem zasięgów dla lotnisk w $2^0$ lotów, $2^1$ lotów itd. Tym razem jednak wystarczy nam pamięć $O(n)$. Będziemy przemieszczać się o $2^i$ lotów ze wszystkich lotnisk naraz, dla tych potęg dwójki, które występują w przedstawieniu

binarnym $x - 1$. Natomiast przy obliczaniu zasięgów w $2^{i+1}$ lotów wystarczy nam pamiętać zasięgi w $2^i$ lotów.

Rozwiązanie takie jest zaimplementowane w pliku `doos1.cpp`. Treść zadania sugeruje, że takie rozwiązanie zdobywa ok. 70% punktów.

## Rozwiązanie wzorcowe $O(sn)$

Zastanówmy się, kiedy punkt startowy potrzebuje o jeden więcej lot niż rozwiązanie optymalne. Dzieje się tak w dwóch przypadkach:

- w pewnym momencie podczas okrążania świata „wlatujemy" na jakieś lotnisko zawarte w rozwiązaniu optymalnym;
- cały czas latamy pomiędzy lotniskami z rozwiązania optymalnego, przenosząc się do o jeden dalszego „odcinka" (tj. fragmentu pomiędzy „optymalnymi" lotniskami), aż w końcu wracamy do pierwszego odcinka, ale w jego wcześniejszej części.

Zauważmy, że jeśli jakieś lotnisko należy do jakiegokolwiek rozwiązania optymalnego, to po wykonaniu z niego lotu o maksymalnej długości znajdziemy się w innym lotnisku należącym do jakiegoś rozwiązania optymalnego. To ważne spostrzeżenie oznacza, że jeśli nasze startowe lotnisko jest lotniskiem o wyniku nieoptymalnym pierwszego typu, to po dokładnie $n$ lotach znajdziemy się na pewno w optymalnym punkcie startowym. Jeśli za to startowe lotnisko jest lotniskiem nieoptymalnym typu drugiego, to po powrocie do startowego odcinka, możemy rozpocząć loty od lotniska, na jakie trafiliśmy. Jeśli jest ono nieoptymalne typu pierwszego, to wiemy już, co zrobić (co więcej, do chwili znalezienia się w jakimś optymalnym punkcie startowym nie odwiedzimy żadnego z już odwiedzonych lotnisk, więc wystarczy nam $n$ lotów minus liczba już wykonanych lotów). Jeśli natomiast jest typu drugiego, to znowu wykonamy jakieś loty i wrócimy na ten sam odcinek – w jeszcze wcześniejszej części. Kontynuujmy latanie. W pewnym momencie powtórzy się jakiś wierzchołek (wtedy mamy pewność, że jest on lotniskiem optymalnym – przecież za każdym razem zmienialiśmy odcinek pomiędzy takimi). Wierzchołek powtórzy się po najwyżej $n$ lotach.

Podsumowując – po wykonaniu $n$ lotów z dowolnego miejsca zawsze znajdujemy w optymalnym punkcie startowym. Obliczenie wyniku jest już tylko formalnością. Rozwiązanie takie jest zaimplementowane w pliku `doo.cpp`.

## Testy

Większość testów ma z góry ustalony wynik dla pewnej długości lotu, składa się z lotnisk parami oddalonych o mniej więcej tyle, a oprócz tego zawiera lotniska „wabiki", które mają różne, atrakcyjne dla rozwiązań heurystycznych, cechy. Testy z grup $a$ mają najwięcej wabików, testy $b$ średnio wiele, a testy $c$ najmniej (gdzie najmniej to około $\frac{9}{10}$ lotnisk, a najwięcej to ok. $\frac{29999}{30000}$). Testy $d$ to testy całkowicie losowe.