Paweł Parys
Treść zadania, Opracowanie

Marcin Pilipczuk
Program

OI, Etap II, dzień próbny, 8–02–2005

Lot na marsa

Bajtazar postanowił polecieć na Marsa, aby zwiedzić istniejące tam stacje badawcze. Wszystkie stacje na Marsie leżą na okręgu. Bajtazar ląduje w jednej z nich, a następnie porusza się za pomocą specjalnego pojazdu, który jest napędzany odpowiednim paliwem. Litr paliwa starcza na metr jazdy. Zapasy paliwa są rozmieszczone w różnych stacjach. Bajtazar może tankować paliwo na stacji, na której w danym momencie się znajduje (zawsze może zatankować cały zapas z danej stacji — pojemność baku jego pojazdu jest nieograniczona). Musi mu to wystarczyć na dojazd do następnej stacji. Bajtazar musi zdecydować, gdzie powinien wylądować, tak żeby mógł zwiedzić wszystkie stacje. Na koniec Bajtazar musi wrócić do stacji, w której wylądował. W czasie podróży Bajtazar musi poruszać się po okręgu, stale w wybranym jednym z dwóch kierunków.

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia liczbę stacji na Marsie, odległości między nimi i ilości paliwa dostępne w każdej z nich,*
- *dla każdej stacji sprawdzi, czy Bajtazar może tam wylądować, czyli czy zaczynając tam i jadąc w wybranym przez siebie kierunku, może objechać wszystkie stacje i wrócić do swojej rakiety,*
- *wypisze wynik na standardowe wyjście.*

Wejście

W pierwszym wierszu standardowego wejścia zapisana jest jedna liczba całkowita n ($3 \leqslant n \leqslant 1\,000\,000$). Jest to liczba stacji na Marsie. Stacje są ponumerowane od 1 do n. W kolejnych n wierszach znajdują się opisy poszczególnych stacji i odległości między nimi. W $(i+1)$-szym wierszu znajdują się dwie liczby całkowite: p_i oraz d_i ($p_i \geqslant 0$, $d_i > 0$). Pierwsza z nich to ilość paliwa w litrach dostępna na i-tej stacji. Druga z nich to odległość w metrach pomiędzy stacją i a $i+1$ (oczywiście d_n to odległość między stacją n a 1). Łączna ilość dostępnego paliwa, a także suma wszystkich odległości między stacjami nie przekracza $2\,000\,000\,000$.

Wyjście

Na standardowe wyjście należy wypisać n wierszy. W i-tym wierszu powinno znajdować się słowo `TAK`*, jeśli Bajtazar może wylądować w stacji numer i, lub* `NIE` *w przeciwnym wypadku.*

Przykład

Dla danych wejściowych:

```
5
3 1
1 2
5 2
0 1
5 4
```

poprawnym wynikiem jest:

```
TAK
NIE
TAK
NIE
TAK
```

Rozwiązanie

Wstępne spostrzeżenia

Pierwsze spostrzeżenie wiąże się z rozmiarem danych wejściowych. Liczba stacji rzędu miliona sugeruje, że powinniśmy poszukiwać algorytmu działającego w czasie liniowym $O(n)$, w najgorszym razie w czasie rzędu $O(n \log n)$, gdzie n oznacza liczbę stacji.

Następnie zauważmy, że zadanie można rozwiązać oddzielnie dla dwóch przypadków — przy założeniu, że Bajtazar porusza się zawsze zgodnie z rosnącym porządkiem numerów stacji (z wyjątkiem przejazdu ze stacji n-tej do pierwszej), lub przy założeniu, że kierunek jego podróży jest przeciwny. Poniżej rozważymy pierwszy z tych przypadków i przedstawimy algorytm pozwalający odnaleźć stacje, z których Bajtazar może wówczas rozpocząć podróż. Aby uzyskać końcowe rozwiązanie, wystarczy opisany algorytm lekko zmodyfikować i zastosować ponownie do przypadku, gdy Bajtazar porusza się w przeciwnym kierunku.

Rozwiązanie zadania rozpoczniemy od oczywistego spostrzeżenia, iż jeśli sumaryczna ilość paliwa na wszystkich stacjach jest mniejsza niż sumaryczna długość drogi, jaką ma przebyć Bajtazar, to nie istnieje stacja, z której Bajtazar może rozpocząć podróż. Co ważniejsze, i już nie tak oczywiste, jeżeli sumaryczna ilość paliwa wystarcza na przebycie całej drogi, to taka stacja istnieje. Pokażemy jak ją znaleźć, a następnie na jej podstawie odnajdziemy pozostałe stacje, w których Bajtazar może wylądować, by odbyć swą podróż po Marsie.

Uważny Czytelnik pilnie śledzący zadania olimpijskie zauważy, że w rozwiązaniu zadania wykorzystuje się ten sam fakt, co w zadaniu „Zwiedzanie” *z finału VIII OI [8].*

Rozwiązanie wzorcowe

Oznaczenia

Przyjmujemy dalej następujące oznaczenia:

- ilość paliwa na stacji i-tej oznaczymy przez p_i, a odległość w metrach między stacją i oraz stacją $i+1$ przez d_i (d_n będzie oznaczać odległość pomiędzy stacją n i stacją 1);
- *stan paliwa na stacji i* to ilość paliwa w baku pojazdu Bajtazara w momencie wjeżdżania na stację i (przed zatankowaniem na tej stacji);
- $S[a,b]$ oznacza zmianę stanu baku pojazdu Bajtazara na odcinku pomiędzy stacją a i stacją b; oczywiście zakładamy, że Bajtazar na każdej stacji tankuje całe znajdujące się tam paliwo, stąd

  $$S[a,b] = p_a - d_a + p_{a+1} - d_{a+1} + \ldots + p_{b-1} - d_{b-1},$$

  przy czym sumowanie we wzorze przeprowadzamy cyklicznie, tzn. $S[a,b] = S[a,n] + p_n - d_n + S[1,a]$, dla $b < a$; wartość $S[a,a]$ można interpretować jako zero (zmianę stanu baku, gdy Bajtazar nie rusza się ze stacji a) lub zmianę stanu baku po objechaniu wszystkich stacji i powrocie do stacji a — od pewnego momentu nie powinno to powodować niejednoznaczności (patrz uwaga 4);
- stację i nazwiemy *stacją wypadową*, jeśli lądując w niej Bajtazar może objechać wszystkie stacje (w rozważanym przypadku zgodnie z rosnącym porządkiem numerów);

Ponadto we wszystkich dalszych rozważaniach numery stacji będziemy traktować cyklicznie, tzn. numery: $-2,-1,0,1,2,\ldots$ są równoważne odpowiednio numerom: $n-2,n-1,n,n+1,n+2,\ldots$.

Znajdowanie pierwszej stacji wypadowej

Na początek pokażemy dwa proste fakty.

Fakt 1 *Jeśli sumaryczna ilość paliwa na Marsie jest mniejsza niż sumaryczna droga do przebycia, tzn. $\sum_{i=1}^{n} p_i < \sum_{i=1}^{n} d_i$, to na Marsie nie istnieje stacja wypadowa.*

Fakt 2 *Stacja a jest stacją wypadową wtedy i tylko wtedy, jeśli dla każdego b zachodzi nierówność $S[a,b] \geqslant 0$.*

Pierwszy fakt jest oczywisty. Podobnie pierwsza część dowodu drugiego faktu. Wystarczy zauważyć, że jeśli a jest stacją, z której Bajtazar rozpoczyna podróż, to $S[a,b]$ oznacza stan baku na stacji b, gdyż bak w chwili początkowej (w momencie wylądowania w stacji a) jest pusty. Stąd jeśli a jest stacją wypadową, to dla każdej stacji b stan baku po dojechaniu do niej z a musi być nieujemny.

Pozostaje wykazać, że jeśli dla każdego b zachodzi $S[a,b] \geqslant 0$, to a jest stacją wypadową. W tym celu rozważmy kolejno $i = a+1, a+2, \ldots, a-1, a$. Nierówność $S[a,a+1] \geqslant 0$ oznacza, że da się dojechać ze stacji a do następnej startując z pustym bakiem. Następnie

wiedząc, że da się dojechać ze stacji a do stacji i, zauważamy, że zapas w baku w momencie wyjazdu ze stacji i, czyli $S[a,i]+p_i$, jest dodatni i wystarcza do pokonania drogi d_i, gdyż $S[a,i]+p_i-d_i = S[a,i+1] \geqslant 0$.

W kolejnym fakcie pokażemy, jak stwierdzić, czy na Marsie istnieje stacja wypadowa, i jak ją znaleźć.

Fakt 3 *Jeśli* $\sum_{i=1}^{n} p_i \geqslant \sum_{i=1}^{n} d_i$, *to istnieje stacja wypadowa.*

Dowód Najpierw, aby uprościć dowód, zmienimy nieco dane wejściowe dbając jednak, by nie wpłynęło to na rozwiązanie zadania. Zauważmy, że możemy wydłużyć drogę na Marsie tak, by dostępna ilość paliwa dokładnie wystarczała na jej pokonanie. Na przykład możemy ustalić $d_n = d_n + \sum_{i=1}^{n} p_i - \sum_{i=1}^{n} d_i$. Jeżeli przy tak zmienionych danych istnieje stacja a, która jest stacją wypadową, to ta sama stacja jest stacją wypadową dla danych oryginalnych. Również w sposób oczywisty stacja, która przed zmianą nie mogła być stacją wypadową — nadal nią nie będzie.

Teraz rozważymy drogę, którą nazwiemy *fikcyjną* — założymy, że pojazd Bajtazara ma na początku w baku zapas paliwa $Z = \sum_{i=1}^{n} d_i$. To oczywiście wystarczy, by odbyć drogę rozpoczynając z dowolnej stacji. W takim razie rozpoczniemy od stacji 1 i będziemy kontynuować podróż przez kolejne stacje zawsze tankując całe znajdujące się tam paliwo. Przez x_i oznaczymy stan baku na stacji i. Niech x_k będzie minimalną spośród wartości $\{x_i \mid 1 \leqslant i \leqslant n\}$.

Możemy teraz rozważyć drugą drogę, którą nazwiemy *rzeczywistą*. Rozpoczynamy ją od stacji k z pustym bakiem. Niech y_i oznacza stan baku na stacji i w trakcie drogi rzeczywistej (mamy więc $y_k = 0$).

Zauważmy, że dla każdego i zarówno $x_{i+1} - x_i = p_i - d_i$, jak i $y_{i+1} - y_i = p_i - d_i$. Stąd dla każdego i zachodzi równość $x_i - y_i = x_{i+1} - y_{i+1}$, a więc różnica pomiędzy stanem baku w trakcie podróży fikcyjnej i rzeczywistej na poszczególnych stacjach jest stała.

Na koniec wystarczy przypomnieć, że x_k była minimalną spośród wartości $\{x_i \mid 1 \leqslant i \leqslant n\}$, więc $y_k = 0$ musi być minimalną spośród wartości $\{y_i \mid 1 \leqslant i \leqslant n\}$. Z tego wnioskujemy, że stan baku w trakcie podróży rzeczywistej nigdy nie spada poniżej zera, więc k jest stacją wypadową. ■

Na podstawie dowodu faktu 3 możemy opisać algorytm znajdowania stacji wypadowej. Jeśli

$$\sum_{i=1}^{n} p_i \quad \geqslant \quad \sum_{i=1}^{n} d_i, \tag{1}$$

to obliczamy wartości $x_i = S[1,i]$ dla $1 \leqslant i \leqslant n$ i wybieramy $x_k = \min\{x_i \mid 1 \leqslant i \leqslant n\}$ znajdując stację wypadową k. Jeśli nierówność (1) nie zachodzi, to odpowiadamy, że stacja wypadowa nie istnieje.

Uwaga 4 *Jeżeli podane wartości* $p_1, p_2, \ldots, p_n$ *i* $d_1, d_2, \ldots, d_n$ *spełniają nierówność (1) i dla pewnej stacji* a *da się z niej dojechać do wszystkich stacji* $b \neq a$, *to da się także powrócić do rakiety zaparkowanej w stacji* a.

To proste spostrzeżenie powoduje, że w dalszych rozważaniach wykorzystujących fakt 2 nie musimy brać pod uwagę wartości $S[a,a]$ *oznaczających zmianę stanu baku po objechaniu wszystkich baz. Od tej chwili przyjmiemy więc* $S[a,a] = 0$ *dla wszystkich stacji* a.

Poszukiwanie pozostałych stacji wypadowych

Na początek pokażemy prosty fakt.

Fakt 5 *Jeśli stacja b jest stacją wypadową i dla pewnej stacji a zachodzą nierówności*

$$\begin{aligned} S[a,b] &\geqslant 0 \quad \text{oraz} \\ S[i,b] &< 0 \quad \text{dla } i = a+1, a+2, \ldots, b-1, \end{aligned}$$

to stacja a jest stacją wypadową, a stacje $a+1, a+2, \ldots, b-1$ *nie są stacjami wypadowymi.*

Dowód Oznaczmy $A = \{a+1, a+2, \ldots, b-1\}$. Ponieważ dla każdej stacji $x \in A$ zachodzi $S[x,b] < 0$, więc z faktu 2 wynika, iż żadna z tych stacji nie jest stacją wypadową.

Pozostaje wykazać, że stacja a jest stacją wypadową. Zauważmy, że dla każdej stacji $x \in A$ zachodzi równość $S[a,x] + S[x,b] = S[a,b]$. Skoro także $S[x,b] < 0$ i $S[a,b] \geqslant 0$, to otrzymujemy, iż $S[a,x] \geqslant 0$. Następnie rozważmy stacje $x \notin A \cup \{b\}$. Dla każdej z nich zachodzi równość $S[a,x] = S[a,b] + S[b,x]$. Ponieważ $S[a,b] \geqslant 0$ oraz b jest stacją wypadową, czyli dla każdej stacji y zachodzi $S[b,y] \geqslant 0$, stąd otrzymujemy $S[a,x] \geqslant 0$. Pozostaje przypomnieć, że również $S[a,b] \geqslant 0$, i na mocy faktu 2 kończymy dowód. ■

Na podstawie dowodu faktu 5 możemy opisać algorytm znajdowania wszystkich stacji wypadowych, znając jedną stację wypadową k. Wystarczy rozważać kolejno (cyklicznie) stacje w malejącym porządku numerów $i = k-1, k-2, \ldots, k+1$ obliczając wartości $S[i,k]$ do czasu, aż natrafimy na pierwszą stację x, dla której $S[x,k] \geqslant 0$. Wówczas z faktu 5 wnioskujemy, że stacja x jest wypadowa, a stacje o numerach $x+1, x+2, \ldots, k-1$ nie są stacjami wypadowymi. Jeśli $x \neq k$, to powtarzamy rozumowanie rozpoczynając tym razem od stacji wypadowej x. W ten sposób rozpoznajemy wszystkie stacje wypadowe dla podróży w pierwszym kierunku.

Algorytm wzorcowy

Wykorzystując pokazane fakty skonstruujemy procedurę znajdowania wszystkich stacji wypadowych dla Bajtazara. Musimy przy tym pamiętać, by uruchomić ją dwukrotnie, aby wykryć stacje wypadowe dla podróży w obu dopuszczalnych kierunkach.

1: Jeżeli $\sum_{i=1}^{n} p_i < \sum_{i=1}^{n} d_i$, to odpowiadamy n razy „NIE” i kończymy.

2: Poszukujemy stacji wypadowych przy założeniu, że Bajtazar będzie podróżował w kierunku rosnących numerów stacji.

   a: Obliczamy wartości $S[1,i]$ dla $1 \leqslant i \leqslant n$ i znajdujemy stację k_0, dla której $S[1,k_0] = \min\{S[1,i] \mid 1 \leqslant i \leqslant n\}$.

   b: Szukamy pozostałych stacji wypadowych k_i, dla $i = 1, 2, \ldots$

      i: Sprawdzamy kolejno stacje $k = k_{i-1} - 1, k_{i-1} - 2, \ldots$, aż natrafimy na pierwszą stację k, dla której $S[k, k_{i-1}] \geqslant 0$ (natomiast $S[j, k_{i-1}] < 0$ dla $j = k+1, k+2, \ldots, k_{i-1} - 1$). Oznaczamy znalezioną stację jako k_i.

      ii: Jeśli $k_i = k_0$, to kończymy pętlę (b).

c: Stacje $k_0, k_1, \ldots, k_{i-1}$ oznaczamy jako stacje wypadowe.

3: Poszukujemy stacji wypadowych przy założeniu, że Bajtazar będzie podróżował w kierunku malejących numerów stacji. W tym celu wykonujemy krok 2 odwracając wszędzie kierunek przeglądania stacji.

4: Dla każdej stacji wypadowej wypisujemy odpowiedź „TAK”, dla pozostałych wypisujemy odpowiedź „NIE”.

W przedstawionym algorytmie kilkakrotnie przeglądamy tablice $p[1..n]$ i $d[1..n]$ oraz wyliczamy kolejne wartości $S[1,i]$ dla $i = 1,2,\ldots$ oraz $S[i,k]$ dla $i = k-1, k-2, \ldots$. Wszystkie te obliczenia wymagają wykonania $O(n)$ operacji. Zauważmy także, że wartości występujące w obliczeniach nie wykraczają poza zakres 32-bitowych liczb całkowitych ze znakiem. Ostatecznie algorytm ma więc złożoność czasową i pamięciową $O(n)$.

Inne rozwiązanie

Przedstawimy także nieco wolniejsze rozwiązanie o złożoności $O(n \log n)$. Podobnie, jak w rozwiązaniu wzorcowym oddzielnie rozważymy dwa dopuszczalne kierunki podróży.

Zakładając, że będziemy poruszać się zgodnie z rosnącym porządkiem numerów stacji, definiujemy tak samo jak poprzednio wartości $S[a,b]$ i sprawdzamy warunek ze wzoru (1). Następnie dla każdej stacji będziemy testować, czy może być stacją wypadową, opierając się, jak poprzednio, na warunku z faktu 2. W związku z tym, dla każdej stacji $a = 1,2,\ldots,n$ musimy znać wartości $S[a,i]$ dla $i = a+1, a+2, \ldots, a$. Jeżeli minimalna spośród nich jest nieujemna, to będziemy wiedzieli, że stacja x może być stacją wypadową.

Dla stacji a obliczane wartości $S[a,i]$ podzielimy na dwie grupy.

$$\begin{aligned} \overline{X}_a &= \{S[a,i] \mid 1 < i \leqslant a\}; \\ \overline{Y}_a &= \{S[a,i] \mid a < i \leqslant 1\}. \end{aligned}$$

Aby stacja a była stacją wypadową musi zachodzić nierówność

$$\min\{\min(\overline{X}_a), \min(\overline{Y}_a)\} \quad \geqslant \quad 0 \tag{2}$$

Jej sprawdzenie wymaga śledzenia zawartości zbiorów $\overline{X}_a$ i $\overline{Y}_a$, co jest o tyle kłopotliwe, że zbiory te zmieniają się istotnie dla kolejnych stacji a. Zastąpimy je więc zbiorami

$$\begin{aligned} X_a &= \{S[1,i] \mid 1 < i \leqslant a\}, \\ Y_a &= \{S[1,i] \mid a < i \leqslant 1\}, \end{aligned}$$

które są związane z poprzednimi następującymi zależnościami

$$\begin{aligned} \overline{X}_a &= \{S[a,i] \mid 1 < i \leqslant a\} = \{S[a,1] + S[1,i] \mid 1 < i \leqslant a\} = S[a,1] + X_a \\ Y_a &= \{S[1,i] \mid a < i \leqslant 1\} = \{S[1,a] + S[a,i] \mid a < i \leqslant 1\} = S[1,a] + \overline{Y}_a, \end{aligned}$$

gdzie dla zbioru $A = \{a_1, a_2, \ldots, a_m\}$ i liczby d przez $d + A$ oznaczymy zbiór $\{d + a_1, d + a_2, \ldots, d + a_m\}$. Wartości zbiorów X_a i Y_a można łatwo aktualizować dla

kolejnych stacji a, ponieważ

$$X_{a+1} = X_a \cup S[1, a+1]\}, \quad (3)$$
$$Y_{a+1} = Y_a \setminus \{S[1, a+1]\}. \quad (4)$$

Ostatecznie warunek (2) pozwalający sprawdzić, czy stacja a jest stacją wypadową, możemy zastąpić warunkiem

$$\min\{\min(X_a + S[a,1]), \min(Y_a - S[1,a])\} \geqslant 0.$$

Cała procedura wyznaczania stacji wypadowych ma następującą postać.

1: Utwórz zbiory $X = \{0\}$ oraz $Y = \{S[1,i] \mid 1 < i \leqslant n\}$;

2: Rozważaj kolejno stacje $a = 1, 2, 3, \ldots, n$:

   a: oblicz $M = \min\{\min(X + S[a,1]), \min(Y - S[1,a])\}$;

   b: jeśli $M < 0$, to odpowiedz, że a nie jest stacją wypadową i przejdź do sprawdzania kolejnej stacji;

   c: oblicz zbiory X i Y dla kolejnej stacji zgodnie ze wzorami (3) i (4).

Zauważmy, że dla kolejnych wartości zbioru X minima możemy wyznaczać w czasie stałym, gdyż do zbioru tego w każdej iteracji pętli 2 jest dorzucany tylko jeden element. W przypadku zbioru Y, z którego w kolejnych iteracjach ujmujemy po jednym elemencie, należy zastosować strukturę, która umożliwi wyznaczanie minimów w czasie logarytmicznym (na przykład kopiec), lub wstępnie posortować zbiór.

Ostatecznie wyliczenie wszystkich minimów może być wykonane w czasie $O(n \log n)$ i w pamięci $O(n)$.

Rozwiązania mniej efektywne i niepoprawne

Kolejne poprawne, aczkolwiek tym razem znacznie wolniejsze od wzorcowego, rozwiązanie polega na „naiwnym" sprawdzeniu każdej stacji. W tym celu symulujemy podróż Bajtazara z tej stacji w obu kierunkach i sprawdzamy, czy da się obejść wszystkie stacje i powrócić do wyjściowej. Algorytm ten działa w czasie $O(n^2)$ i wymaga pamięci $O(n)$.

Wśród rozwiązań zawodników mogą również pojawić się różne błędne heurystyki. Na przykład decydowanie o tym, czy stacja jest wypadową, na podstawie możliwości dojścia do kilku stacji pośrednich. Programy takie nie powinny przejść przez zaproponowane testy.

Testy

Rozwiązania zawodników były sprawdzane na 11 testach. W każdym, poza *lot1b.in*, opisana jest sytuacja, gdy istnieje stacja wypadowa, tzn. na Marsie jest wystarczająco wiele paliwa, by objechać wszystkie stacje. Testy *lot1a.in* i *lot1b.in* zostały połączone w jeden zestaw.
W przedstawionej poniżej tabeli n oznacza liczbę stacji na Marsie, natomiast #TAK liczbę stacji wypadowych.

Nazwa	n	#TAK	Opis
lot1a.in	20	5	prosty test poprawnościowy
lot1b.in	100 000	0	spory test, w którym w stacjach nie ma dość paliwa, by obejść wszystkie stacje
lot2.in	1 001	9	test, w którym występują małe wartości p_i oraz d_i, a w czasie podróży Bajtazar ma zawsze w baku niewiele paliwa — algorytmy działające w czasie kwadratowym powinny zaliczyć ten test
lot3.in	15 000	5 919	test, w którym w stacjach jest bardzo dużo paliwa; oprócz dwóch najszybszych algorytmów, także zoptymalizowany algorytm kwadratowy ma szansę go zaliczyć
lot4.in	50 053	50 001	test, w którym prawie każda stacja jest dopuszczalna
lot5.in	306 000	67 416	test, w którym droga jest podzielona na małe odcinki po 100 wierzchołków i w stacjach pomiędzy odcinkami jest dużo paliwa — wystarczy wyjść poza odcinek, by objechać wszystkie stacje
lot6.in	901 800	71 498	test analogiczny do poprzedniego, ale złożony z 900 odcinków długości 1000
lot7.in	100 000	1	test, w którym z każdej stacji da sie objechać prawie wszystkie, ale istnieje tylko jedna wypadowa
lot8.in	1 000 000	492 891	test, w którym zapasy paliwa są duże
lot9.in	1 000 000	18	test, w którym zapasy paliwa ledwo wystarczają, by objechać wszystkie stacje
lot10.in	1 000 000	19 484	test, w którym zapasy paliwa są nieco większe niż w poprzednim, ale nadal jest go niewiele