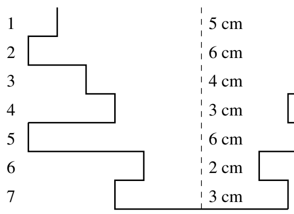
Jakub Radoszewski **Michał Adamaszek**
Treść zadania, Opracowanie Program

OI, Etap I

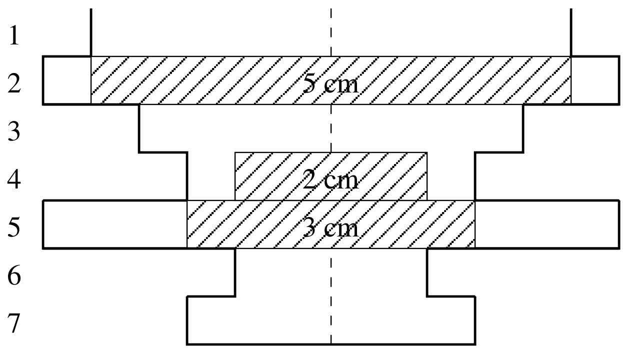
Krążki

Mały Jaś dostał od rodziców na urodziny nową zabawkę, w której skład wchodzą rurka i krążki. Rurka ma nietypowy kształt — mianowicie jest to połączenie pewnej liczby walców (o takiej samej grubości) z wyciętymi w środku (współosiowo) okrągłymi otworami różnej średnicy. Rurka jest zamknięta od dołu, a otwarta od góry. Na poniższym rysunku przedstawiono przykładową taką rurkę, złożoną z walców, w których wycięto otwory o średnicach kolejno: 5 cm, 6 cm, 4 cm, 3 cm, 6 cm, 2 cm i 3 cm.



Krążki w zabawce Jasia są walcami o różnych średnicach i takiej samej grubości, co walce tworzące rurkę.

Jaś wymyślił sobie następującą zabawę. Mając do dyspozycji pewien zestaw krążków zastanawia się, na jakiej głębokości zatrzymałby się ostatni z nich, gdyby wrzucał je kolejno do rurki centralnie (czyli dokładnie w jej środek). Dla przykładu, gdyby wrzucić do powyższej rurki krążki o średnicach kolejno 3 cm, 2 cm i 5 cm, to otrzymalibyśmy następującą sytuację:



Jak widać, każdy kolejny krążek po wrzuceniu spada dopóki się nie zaklinuje (czyli nie oprze się o walec, w którym wycięty jest otwór o mniejszej średnicy niż średnica krążka) albo nie natrafi na przeszkodę w postaci innego krążka lub dna rurki.

Ponieważ zabawa ta jest trudna dla małego Jasia, to ciągle prosi swoich rodziców o pomoc. A jako że rodzice Jasia nie lubią takich zabaw intelektualnych, to poprosili Ciebie — znajomego programistę — o napisanie programu, który zamiast nich będzie udzielał odpowiedzi Jasiowi.

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia schemat rurki i opis krążków, jakie Jaś będzie wrzucał do rurki,*
- *wyznaczy głębokość, na jakiej zatrzyma się ostatni wrzucony przez Jasia krążek,*
- *wypisze wynik na standardowe wyjście.*

Wejście

Pierwszy wiersz wejścia zawiera dwie liczby całkowite n i m ($1 \leqslant n,m \leqslant 300\,000$), oddzielone pojedynczym odstępem i oznaczające wysokość rurki Jasia (liczbę walców wchodzących w jej skład) i liczbę krążków, które zamierza wrzucić do rurki. Drugi wiersz wejścia zawiera n liczb całkowitych $r_1, r_2, \ldots, r_n$ ($1 \leqslant r_i \leqslant 1\,000\,000\,000$ dla $1 \leqslant i \leqslant n$) oddzielonych pojedynczymi odstępami i oznaczających średnice otworów wyciętych w kolejnych (od góry) walcach tworzących rurkę. Trzeci wiersz wejścia zawiera m liczb całkowitych $k_1, k_2, \ldots, k_m$ ($1 \leqslant k_j \leqslant 1\,000\,000\,000$ dla $1 \leqslant j \leqslant m$) oddzielonych pojedynczymi odstępami i oznaczających średnice kolejnych krążków, które Jaś zamierza wrzucić do rurki.

Wyjście

Pierwszy i jedyny wiersz wyjścia powinien zawierać jedną liczbę całkowitą, oznaczającą głębokość zatrzymania się ostatniego krążka. Jeżeli krążek ten w ogóle nie wpadnie do rurki, to odpowiedzią powinna być liczba 0.

Przykład

Dla danych wejściowych:

```
7 3
5 6 4 3 6 2 3
3 2 5
```

poprawnym wynikiem jest:

```
2
```

Rozwiązanie

Rozwiązanie o złożoności czasowej $O(n + m\log n)$

Najprostszym rozwiązaniem jest symulacja sytuacji opisanej w zadaniu — każdy krążek spada, dopóki nie zaklinuje się bądź nie natrafi na przeszkodę w postaci innego krążka lub dna rurki. Symulację spadania jednego krążka możemy przeprowadzić w czasie $O(n)$ — po prostu przeglądamy rurkę od góry, poszukując miejsca, w którym zatrzyma się dany krążek. Całkowita złożoność takiego rozwiązania to $O(nm)$. Jest ona zdecydowanie za duża jak na ograniczenia z zadania.

Kluczem do rozwiązania zadania jest przyspieszenie powyższej symulacji — musimy znaleźć poziom, na którym zatrzyma się dany krążek w czasie krótszym niż $O(n)$. Oczywiście bierzemy pod uwagę jedynie poziomy powyżej pozycji ostatnio wrzuconego krążka. Następny krążek może dolecieć do określonego poziomu, jeżeli wcześniej nie zaklinuje się, czyli jeżeli wszystkie walce na wyższych poziomach mają średnice niemniejsze od średnicy krążka. To sformułowanie jest równoważne temu, że *minimum* ze średnic wszystkich walców położonych powyżej danego poziomu musi być niemniejsze od średnicy krążka. Jeżeli na początku policzymy minima $m_1, m_2, \ldots, m_n$ dla wszystkich poziomów (można to zrobić w czasie $O(n)$), to w czasie stałym uzyskujemy odpowiedź na pytanie, czy krążek może dolecieć na dany poziom. Co ważniejsze, obliczony ciąg wartości $m_1, m_2, \ldots, m_n$ jest niemalejący i w celu znalezienia najniższego poziomu i, dla którego m_i jest niemniejsze od średnicy wrzucanego krążka, można zastosować wyszukiwanie binarne. To oznacza, że pozycję końcową jednego krążka umiemy znaleźć w czasie $O(\log n)$, a zatem całe rozwiązanie ma złożoność $O(n + m \log n)$.

Poniżej zamieszczamy pseudokod tego rozwiązania. Pełny kod można znaleźć na dysku dołączonym do książeczki.

```
program Rozwiązanie pierwsze;
{ Liczymy ciąg minimów m_i }
m_1 := r_1;
for i := 2 to n do
  m_i := min(m_{i-1}, r_i);
{ Poszukiwanie pozycji krążków }
poprzedni := n+1; { pierwszy krążek może się zatrzymać }
                  { najdalej na dnie rurki }
for j := 1 to m do
begin  { Wyszukiwanie binarne }
  a := 0, b := poprzedni-1;
  while a < b do
  begin { Uwaga! Wyszukujemy binarnie wśród liczb całkowitych! }
    c := ⌊(a+b)/2⌋ + 1;
    if m_c < k_j then
      { Tak daleko krążek nie doleci }
      b := c-1;
    else
      { Tutaj krążek może dolecieć }
      a := c;
  end
  poprzedni := a;
  if poprzedni = 0 then
    return 0;
end
return poprzedni;
```

Rozwiązanie o złożoności czasowej $O(n+m)$

Co prawda powyższe rozwiązanie przechodzi wszystkie testy, jednak możemy je jeszcze usprawnić. Zamiast wyszukiwać pozycję krążka binarnie wykorzystamy proste wyszukiwanie liniowe, z tym, że rozpoczniemy je od pozycji poprzedniego krążka (dla pierwszego krążka — od dna rurki). Począwszy od tej pozycji będziemy badać kolejne, coraz płytsze, do momentu aż znajdziemy poziom i o wystarczająco dużej wartości m_i — będzie to poziom, na którym zatrzyma się dany krążek.

Mimo, że jeden krok takiego wyszukiwania może w najgorszym przypadku wymagać przejrzenia nawet $O(n)$ poziomów, to łatwo zauważyć, że każdy poziom analizujemy *co najwyżej* raz. Jeśli bowiem w trakcie poszukiwania pozycji krążka rozważamy pewien poziom, to oznacza, że dany krążek zatrzyma się na nim lub ponad nim i poszukując pozycji pozostałych krążków będziemy już rozważać tylko poziomy leżące powyżej. Stąd wynika, że sumaryczna złożoność czasowa algorytmu wynosi $O(n+m)$ (stały czas na rozważanie każdego poziomu oraz stały czas na rozważenie każdego krążka).

Z analizy tego prostego algorytmu można wysnuć ciekawy wniosek: algorytm, którego dowolny krok może mieć pesymistycznie dużą złożoność, nie musi być wolnym algorytmem. Jego *całkowita* złożoność może okazać się istotnie lepsza niż iloczyn pesymistycznej złożoności jednego jego kroku i liczby wszystkich kroków (choć oczywiście nie jest to regułą).

Oto pseudokod tego rozwiązania (pełny kod można znaleźć na dysku dołączonym do książeczki):

```
program Rozwiązanie drugie
{ Liczymy ciąg minimów m_i }
m_1 := r_1;
for i := 2 to n do
  m_i := min(m_{i-1}, r_i);
{ Poszukiwanie pozycji krążków }
poprzedni := n+1; { pierwszy krążek może się zatrzymać }
                  { najdalej na dnie rurki }
for j := 1 to m do
begin
  { Wyszukiwanie liniowe od pozycji poprzedniego krążka }
  a := poprzedni - 1;
  while (a > 0) and (m_a < k_j) do
    a := a - 1;
  poprzedni := a;
  if poprzedni = 0 then
    return 0;
end
return poprzedni;
```

Testy

Zadanie testowane było na zestawie 11 danych. Dwa z nich stanowiły grupy po dwa testy — jeden miał odpowiedź równą 0 (czyli nie wszystkie krążki mieściły się w rurce), a drugi dodatnią.

Nazwa	n	m	Opis
kra1a.in	50	51	za dużo krążków, odpowiedź 0
kra1b.in	100	60	rurka zwęża się
kra2.in	353	71	losowe nieregularności w rurce
kra3a.in	100	25	odpowiedź 0
kra3b.in	1000	600	rurka zwęża się ku górze
kra4.in	7100	1457	długie sekwencje jednakowych średnic
kra5.in	50001	20043	rurka na przemian rozszerza się i zwęża
kra6.in	95000	63000	miejscowe przewężenia w rurce
kra7.in	120001	97003	równomierna szerokość rurki z dwiema przeszkodami
kra8.in	250002	90001	rurka rozszerza się losowo ku górze
kra9.in	300000	90000	rurka najpierw rozszerza się, a potem zwęża
kra10.in	300000	240301	rurka na przemian rozszerza się i zwęża
kra11.in	300000	280001	rurka rozszerza się losowo ku górze