Usuwanka (rozwiązanie)

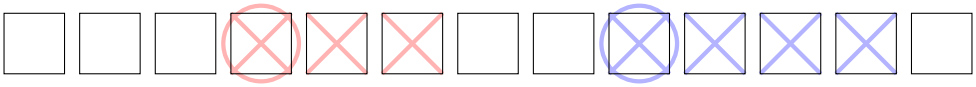
Autor zadania: **Michał Bryjak**
Opracowanie: **Michał Górniak, Michał Niedziółka, Tymoteusz Wiśniewski**
Opis rozwiązania: **Bartosz Kostka, Tymoteusz Wiśniewski**

Olimpiada Informatyczna Juniorów

Mamy dany ciąg klocków (a tak naprawdę liczb) T. W każdym ruchu możemy wybrać dowolny element T_i z tego ciągu i usunąć go oraz T_i kolejnych liczb z jego prawej strony (o ile istnieją). Naszym celem jest usunięcie wszystkich liczb z ciągu w minimalnej liczbie ruchów.

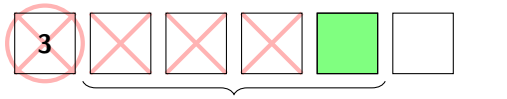
Aby pojęcia nam się nie myliły, będziemy dalej mówili, że jeżeli w danym ruchu wybieramy klocek do usunięcia to go *klikamy*. Ma to na celu rozróżnienie usuwania klocka jako wybrania go i usuwania klocków po prawej stronie wybranego klocka jako następstwo kliknięcia.

Załóżmy że wybraliśmy już jakie klocki będziemy klikać i teraz decydujemy tylko w jakiej kolejności będziemy to robić. Okazuje się, że możemy zawsze klikać wybrane klocki od prawej do lewej. Aby to pokazać, zastanówmy się co by się stało jakby w optymalnym rozwiązaniu istniała jakaś para klocków, taka że klocek z lewej strony potrzebowalibyśmy kliknąć przed jakimś klockiem z prawej strony. Naturalnie nie może być tak, że lewy klocek usunie prawy klocek (ponieważ wtedy nie moglibyśmy później kliknąć prawego klocka), zatem te klocki usuwają rozłączne (nie pokrywające się) zbiory klocków.



Jeżeli tak jest, to nic nie stoi na przeszkodzie by najpierw kliknąć jednak prawy klocek, a później lewy.

Wiemy już w jakiej kolejności klikać klocki. Zastanówmy się teraz jakie klocki wybrać. Na początku zauważmy, że na pewno musimy kliknąć pierwszy klocek, gdyż nie ma innej mozliwości usunięcia go z ciągu. Zauważmy także, że zgodnie z powyższą obserwacją, będzie to ostatni kliknięty klocek. Klocek ten ma jakąś wartość T_1. Teraz zauważmy, że używając podobnej argumentacji, możemy powiedzieć, że któryś z klocków $2, 3, 4, \ldots, T_1 + 1$ także musi być kliknięty, ponieważ tylko to spowoduje usunięcie klocka $T_1 + 1$.



musimy kliknąć któryś z tych klocków, aby usunąć zielony klocek

Który z tych klocków klikniemy? Jako że chcemy minimalizować liczbę ruchów, opłaca nam się pewnie wybrać klocek o największej wartości, bo usuniemy najwięcej innych klocków. Okazuje się, że to zachłanne podejście jest poprawne. Co więcej, to rozumowanie możemy kontynuować. Załóżmy, że kliknęliśmy już pewne klocki o indeksach $(i_1, i_2, i_3, \ldots, i_k)$ i zastanawiamy się jaki kolejny klocek powinniśmy kliknąć. Ponieważ wybraliśmy już te klocki, wiemy że usuniemy w sumie $(T_{i_1}+1)+(T_{i_2}+1)+\ldots+(T_{i_k}+1)$ klocków z lewej strony, ale kolejnego już nie. Powinniśmy zatem wybrać dowolny, niewybrany jeszcze, klocek z pierwszych $(T_{i_1} + 1) + (T_{i_2} + 1) + \ldots + (T_{i_k} + 1) + \mathbf{1}$ klocków z lewej strony. Jak już wspomnieliśmy, opłaca nam się zawsze wybrać klocek o największej wartości. Łatwo zauważyć, że używając naszej metody klikania klocków od prawej do lewej, jesteśmy w stanie kliknąć we wszystkie nasze wybrane klocki i można udowodnić, że jest to optymalne rozwiązanie.

Zastanówmy się teraz, jak zaimplementować takie rozwiązanie. Będziemy przechodzili ciąg klocków od lewej do prawej i utrzymywali ciąg klocków, które możemy kliknąć (będziemy takie klocki nazywali kandydatami). Początkowo jedynym kandydatem jest pierwszy klocek. W każdym momencie, kiedy musimy podjąć jakiś wybór i wybrać jakiś klocek do kliknięcia, wybieramy wśród zbioru kandydatów klocek o największej wartości, usuwamy go ze zbioru kandydatów i dodajemy kolejne $x + 1$ elementów do naszego zbioru kandydatów, gdzie x jest wartością wybranego klocka. Kontynuujemy cały proces póki nie usuniemy wszystkim elementów ciągu.

Możemy oczywiście trzymać listę kandydatów jako listę lub tablicę, którą będziemy musieli sortować za każdym razem, ale okaże się to za wolne. Powinniśmy tutaj użyć jakiejś struktury danych, która umożliwia nam następujące operacje:

- dodaj element do zbioru,
- wybierz i usuń maksimum ze zbioru.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

W standardowych bibliotekach mamy kilka struktur danych, które umożliwiają wykonywanie tych operacji w czasie $O(logN)$, gdzie N to liczba elementów w zbiorze.

W języku C++ możemy użyć:

- kontenera `priority_queue` z biblioteki `queue`[a],
- funkcji `push_heap`, `pop_heap` z biblioteki `algorithm`[b],
- kontenera `set` z biblioteki o tej samej nazwie[c].

[a] https://en.cppreference.com/w/cpp/container/priority_queue
[b] https://en.cppreference.com/w/cpp/algorithm/push_heap, https://en.cppreference.com/w/cpp/algorithm/pop_heap
[c] https://en.cppreference.com/w/cpp/container/set

Z kolei w Pythonie możemy użyć:

- kontenera `PriorityQueue` z biblioteki `queue`[a],
- kontenera `heapq`[b].

[a] https://docs.python.org/3.7/library/queue.html#queue.PriorityQueue
[b] https://docs.python.org/3.7/library/heapq.html

Finalnie otrzymujemy rozwiązanie działające w czasie $O(N \log N)$ i pamięci $O(N)$.

usu.cpp

```cpp
#include "bits/stdc++.h"

using namespace std;

int main() {
  // Wczytujemy dane z wejścia.
  int N;
  cin >> N;
  vector <int> T(N);
  for (int i=0; i<N; i++) cin >> T[i];

  // Deklarujemy zmienne pomocnicze:
  // id - gdzie obecnie się znajdujemy w ciągu
  // granica - gdzie jest ostatni element, który mozemy kliknac
  // klikniecia - ile ruchów w grze wykonaliśmy (to jest wynik)
  int id = 0, granica = 0, klikniecia = 0;

  // Deklarujemy strukturę, która pozwala nam dynamicznie dodawać elementy
  // i wyciągać minimum - kolejkę priorytetową.  Trzymamy w niej elementy,
  // które możemy kliknac.
  priority_queue <int> kandydaci;

  // Dopóki nie usunęliśmy wszystkich elementów z ciągu.
  while (granica < N) {
    // Dodaj do struktury kandydatow wszystkie elementy do granicy.
    while (id <= granica) {
      kandydaci.push(T[id]);
      id++;
    }
    // Wybierz element, który będziemy chcieli kliknąć - maksimum wśród kandydatów.
    int element_klikany = kandydaci.top();
    kandydaci.pop();

    // Kliknij ten element, to jest przestaw granicę o wartość tego elementu + 1.
    granica += element_klikany + 1;
    // Wykonaliśmy jeden ruch, zatem zwiększ licznik kliknięc o 1.
    klikniecia++;
  }

  // Na końcu wypisujemy ile kliknięć wykonaliśmy.
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

```
  cout << klikniecia << "\n";
}
```

usu.py

```
from queue import PriorityQueue


def main():
  # Wczytujemy dane z wejścia.
  N = int(input())
  T = list(map(int, input().split()))

  # Deklarujemy zmienne pomocnicze:
  # id - gdzie obecnie się znajdujemy w ciągu
  # granica - gdzie jest ostatni element, który możemy kliknac
  # klikniecia - ile ruchów w grze wykonaliśmy (to jest wynik)
  id = 0
  granica = 0
  klikniecia = 0

  # Deklarujemy strukturę, która pozwala nam dynamicznie dodawać elementy
  # i wyciągać maksimum - kolejkę priorytetową.  Trzymamy w niej elementy,
  # które możemy usunąć.  Należy zwrócić uwagę, że kolejka priorytetowa w Pythonie
  # zwraca minimum, zatem będziemy w niej trzymać wartości pomnożone przez -1.
  # Na przykład zamiast 5, 3, 4, będziemy trzymać -5, -3, -4.
  kandydaci = PriorityQueue()

  # Dopóki nie usunęliśmy wszystkich elementów z ciągu.
  while granica < N:
    # Dodaj do struktury kandydatow wszystkie elementy do granicy.
    while id <= granica:
      # Pamiętaj że wstawiamy wartość przeciwną do T[id]!
      kandydaci.put(-T[id])
      id += 1

    # Wybierz element, który będziemy chcieli kliknąć - maksimum wśród kandydatów,
    # pamiętając aby wziąć wartość przeciwną.
    element_usuwany = -kandydaci.get()

    # Kliknij ten element, to jest przestaw granicę o wartość tego elementu + 1.
    granica += element_usuwany + 1
    # Wykonaliśmy jeden ruch, zatem zwiększ licznik klikniecia o 1.
    klikniecia += 1


  # Na końcu wypisujemy ile kliknięć wykonaliśmy.
  print(klikniecia)


main()
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".