Notowania akcji (rozwiązanie)

XIV OIJ, próbne zawody II stopnia
29 lutego 2020

Kod zadania: **not**
Limit czasu: **10 s**
Limit pamięci: **256 MB**

Olimpiada Informatyczna Juniorów

Zadanie pochodzi z Wrocławskich Sparingów Informatycznych (`https://solve.edu.pl/~sparingi/`). Omówienie video znajduje się pod adresem: `https://www.youtube.com/watch?v=uyIO97zHk44`. Gorąco zachęcamy do udziału w sparingach, które stanowią świetne przygotowanie do Olimpiady Informatycznej Juniorów.

W zadaniu na wejściu dostajemy ciąg liczb naturalnych oraz zapytania o liczbę ściśle rosnących spójnych fragmentów tego ciągu określonej długości.

Rozwiązanie na 17% punktów

W pierwszym podzadaniu na wejściu było tylko jedno zapytanie i tylko o fragmenty rosnące długości 2. Wystarczy więc policzyć ile jest par sąsiednich elementów, z których pierwszy jest mniejszy niż drugi.

not1.py

```python
n = int(input())
ciag = list(map(int, input().split()))

q = int(input())
for _ in range(q):
  k = int(input())  # zakladamy, ze k = 2

  wynik = 0
  for i in range(n-1):
    if ciag[i] < ciag[i+1]:
      wynik += 1

  print(wynik)
```

not1.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAX_N = 500005;

int n;
int ciag[MAX_N];

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);

  cin >> n;
  for (int i = 0; i < n; i++)
    cin >> ciag[i];

  int q;
  cin >> q;
  for (int i = 0; i < q; i++) {
    int k;
    cin >> k;  // zakladamy, ze k = 2

    int wynik = 0;
    for (int i = 0; i < n-1; i++)
```


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".

```
25        if (ciag[i] < ciag[i+1])
26          wynik++;
27
28      cout << wynik << "\n";
29    }
30
31    return 0;
32  }
```

Rozwiązanie na 47% punktów

W drugim podzadaniu na wejściu było tylko jedno zapytanie.

Porównując kolejne dwa sąsiednie elementy od lewej strony, możemy podzielić ciąg na możliwie długie (nieprzedłużalne) rozłączne fragmenty rosnące. Jeśli fragment rosnący składa się z p liczb to jest w nim $\max(0, p-q)$ spójnych fragmentów rosnących składających się z q liczb. Dla ustalonego zapytania wystarczy więc posumować tę liczbę dla każdego wcześniej znalezionego fragmentu rosnącego.

not2.py

```
1  def oblicz_dlug_nieprzedluzalne(n, ciag):
2    dlug_nieprzedluzalne = []
3    aktualna_dlugosc = 1
4    for i in range(n):  # na koncu porownamy sie z ciag[n+1], ktory jest zerem
5      if ciag[i] < ciag[i+1]:
6        aktualna_dlugosc += 1
7      else:
8        dlug_nieprzedluzalne.append(aktualna_dlugosc)
9        aktualna_dlugosc = 1
10   return dlug_nieprzedluzalne
11
12
13 def ile_fragmentow(dlug_nieprzedluzalne, k):
14   wynik = 0
15   for x in dlug_nieprzedluzalne:
16     wynik += max(0, x-k+1)
17   return wynik
18
19
20 n = int(input())
21 ciag = list(map(int, input().split()))
22 ciag.append(0)  # dodajemy zero na koncu ciagu dla latwiejszej implementacji
23
24 dlug_nieprzedluzalne = oblicz_dlug_nieprzedluzalne(n, ciag)
25
26 q = int(input())
27 for _ in range(q):
28   k = int(input())
29   print(ile_fragmentow(dlug_nieprzedluzalne, k))
```

not2.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int MAX_N = 500005;
5
6  int n;
7  int ciag[MAX_N];
8
```


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".

```cpp
vector<int> oblicz_dlug_nieprzedluzalne() {
  vector<int> dlug_nieprzedluzalne;
  int aktualna_dlugosc = 1;
  for (int i = 0; i < n; i++) {  // na koncu porownamy sie z ciag[n+1], ktory jest zerem
    if (ciag[i] < ciag[i+1])
      aktualna_dlugosc++;
    else {
      dlug_nieprzedluzalne.push_back(aktualna_dlugosc);
      aktualna_dlugosc = 1;
    }
  }
  return dlug_nieprzedluzalne;
}

int ile_fragmentow(const vector<int>& dlug_nieprzedluzalne, int k) {
  int wynik = 0;
  for (int x : dlug_nieprzedluzalne)
    wynik += max(0, x-k+1);
  return wynik;
}

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);

  cin >> n;
  for (int i = 0; i < n; i++)
    cin >> ciag[i];

  vector<int> dlug_nieprzedluzalne = oblicz_dlug_nieprzedluzalne();

  int q;
  cin >> q;
  for (int i = 0; i < q; i++) {
    int k;
    cin >> k;
    cout << ile_fragmentow(dlug_nieprzedluzalne, k) << "\n";
  }

  return 0;
}
```

Rozwiązanie wzorcowe

Chcąc rozwiązać zadanie na 100% punktów, nie możemy sobie pozwolić na uruchomienie powyższego algorytmu z osobna dla każdego zapytania. Plan na rozwiązanie jest następujący: po podzieleniu ciągu na nieprzedłużalne fragmenty rosnące (obliczeniu tablicy C, w której $C[i]$ oznacza liczbę nieprzedłużalnych fragmentów rosnących o długości i) chcemy szybko ustalić odpowiedzi dla wszystkich możliwych zapytań, które mogą nadejść. Następnie odpowiadać na każde wczytane zapytanie od razu, już bez żadnych dalszych obliczeń, zgodnie z wcześniej obliczonymi wynikami.

Jak już zauważyliśmy wcześniej, we fragmencie rosnącym o długości p znajdują się jeszcze dwa fragmenty rosnące o długości $p-1$, trzy fragmenty rosnące o długości $p-2$, ... oraz p fragmentów o długości 1. A zatem fragmentów rosnących długości p jest $R[p] = C[p]+2\cdot C[p+1]+3\cdot C[p+2]+\cdots+(n-p+1)\cdot C[n]$. Ustalmy więc odpowiedzi (elementy tablicy R) od końca, od $p = n$ do $p = 1$. Być może jeśli znamy już $R[p+1]$, to pomoże nam to w ustaleniu $R[p]$? Otóż tak, bo $R[p] = R[p+1] + C[p] + C[p+1] + C[p+2] + \cdots + C[n]$ A skoro obliczamy wyniki od końca, od największych indeksów w tablicy R, podczas obliczania $R[p]$ możemy dodatkowo utrzymywać sumę napotkanych dotychczas wartości

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".

tablicy C o indeksach większych lub równych p (sumę $C[p] + C[p+1] + C[p+2] + \cdots + C[n]$).

not3.py

```python
def oblicz_ile_nieprzedluzalne(n, ciag):
  ile_nieprzedluzalne = [0] * (n+1)
  aktualna_dlugosc = 1
  for i in range(n):  # na koncu porownamy sie z ciag[n+1], ktory jest zerem
    if ciag[i] < ciag[i+1]:
      aktualna_dlugosc += 1
    else:
      ile_nieprzedluzalne[aktualna_dlugosc] += 1
      aktualna_dlugosc = 1
  return ile_nieprzedluzalne


def sumuj_wyniki(n, ile_nieprzedluzalne):
  wyniki = [0] * (n+2)
  suma = 0
  for i in range(n, 0, -1):  # na poczatku wyniki[n+1] jest 0
    suma += ile_nieprzedluzalne[i]
    wyniki[i] = wyniki[i+1] + suma
  return wyniki


n = int(input())
ciag = list(map(int, input().split()))
ciag.append(0)  # dodajemy zero na koncu ciagu dla latwiejszej implementacji

ile_nieprzedluzalne = oblicz_ile_nieprzedluzalne(n, ciag)
wyniki = sumuj_wyniki(n, ile_nieprzedluzalne)

q = int(input())
for _ in range(q):
  k = int(input())
  print(wyniki[k])
```

not3.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAX_N = 500005;

int n;
int ciag[MAX_N];

vector<int> oblicz_ile_nieprzedluzalne() {
  vector<int> ile_nieprzedluzalne(n+1, 0);
  int aktualna_dlugosc = 1;
  for (int i = 0; i < n; i++) {  // na koncu porownamy sie z ciag[n+1], ktory jest zerem
    if (ciag[i] < ciag[i+1])
      aktualna_dlugosc++;
    else {
      ile_nieprzedluzalne[aktualna_dlugosc]++;
      aktualna_dlugosc = 1;
    }
  }
  return ile_nieprzedluzalne;
}

```


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".

```cpp
vector<int> sumuj_wyniki(const vector<int>& ile_nieprzedluzalne) {
  vector<int> wyniki(n+2, 0);
  int suma = 0;
  for (int i = n; i >= 1; i--) {  // na poczatku wyniki[n+1] jest 0
    suma += ile_nieprzedluzalne[i];
    wyniki[i] = wyniki[i+1] + suma;
  }
  return wyniki;
}

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);

  cin >> n;
  for (int i = 0; i < n; i++)
    cin >> ciag[i];

  vector<int> ile_nieprzedluzalne = oblicz_ile_nieprzedluzalne();
  vector<int> wyniki = sumuj_wyniki(ile_nieprzedluzalne);

  int q;
  cin >> q;
  for (int i = 0; i < q; i++) {
    int k;
    cin >> k;
    cout << wyniki[k] << "\n";
  }

  return 0;
}
```


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".