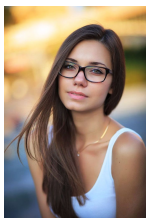

Halinka i książki

Wskazówka

Wartości wysokości książek w zadaniu pochodzą z bardzo ograniczonego (i dyskretnego) zbioru liczb:

$$\{50, 51, 52, \ldots, 499, 500\}$$

Dzięki temu łatwo zliczyć ilości wystąpień poszczególnych liczb. Do tego posłuży nam pomocnicza tablica liczników. Zadeklarujemy ją na wyrost (rozmiar = 501) i po prostu nie będziemy wykorzystywać jej początkowego fragmentu. W tablicy liczników wyszukujemy największą występującą w niej wartość, a jeśli będziemy przeglądać ją od lewej do prawej (nie odwrotnie), wtedy mamy zagwarantowane spełnienie dodatkowego warunku podanego w zadaniu: jeśli kilka wartości wysokości występuje tyle samo razy (mówimy o najczęściej występujących wartościach), wtedy należy wybrać największą z nich (dlaczego taka metoda działa?).

Kod w języku C++

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

int main()
{
  int L[501] = {0};
  int n; cin >> n;
  for(int i = 0; i < n; i++)
  {
    int h; cin >> h;
    L[h]++;
  }
  int idom = 0;
  for(int i = 50; i <= 500; i++)
    if(L[i] >= L[idom]) idom = i;
  cout << idom << endl;
}
```

Proszę zwrócić uwagę na nieostrą nierówność w instrukcji warunkowej w drugiej pętli `for`.

Kod w języku Python

```
L = [0] * 501
n = int(input())
```

```
h = [int(x) for x in input().split()]
for k in h:
  L[k] += 1
m = max(L)
A = [z for z in range(501) if L[z] == m]
print(max(A))
```

Uwagi

Pytanie brzmi, czy poradzilibyśmy sobie równie łatwo, gdyby wczytywane liczby pochodziły ze znacznie większego zakresu (na przykład do miliarda) lub posiadały część ułamkową (typ `double`). Wówczas należałoby się posłużyć nieco inną strukturą danych niż zwykła tablica (w przypadku Pythona: lista), a mianowicie *mapą* (w przypadku Pythona: słownikiem), gdzie rolę klucza pełniłaby dana wartość wysokości, zaś drugą składową elementu mapy byłby licznik wystąpień (liczba naturalna). Należy wszelako pamiętać, że takie struktury działają nieco wolniej od zwykłych tablic, ale – co gorsza – mają bardzo duży narzut pamięciowy. Jednak ich funkcjonalność bywa czasami – po prostu – niezastąpiona.