Informatyczny kącik olimpijski (97): Flagi

W styczniu br. nakładem Wydawnictwa Naukowego PWN ukazała się książka Jacka Tomasiewicza *Zaprzyjaźnij się z algorytmami. Przewodnik dla początkujących i średnio zaawansowanych.*

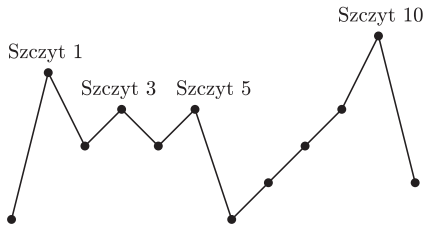
Książka ta zawiera opis podstawowych i najważniejszych technik algorytmicznych i struktur danych, które zostały uporządkowane w osiemnastu rozdziałach.

Do każdego tematu wybrano zadania o zróżnicowanym poziomie trudności, odpowiednie zarówno dla początkujących, jak i bardziej zaawansowanych czytelników. Zadania pochodzą z konkursów, takich jak Olimpiada Informatyczna, oraz obozów informatycznych ILOCAMP.

Implementacje swoich rozwiązań można testować pod kątem poprawności oraz wydajności w serwisie `main2.edu.pl`

W tym miesiącu, chcąc polecić Czytelnikom kącika książkę *Zaprzyjaźnij się z algorytmami*, przedstawimy jedno zadanie z tej pozycji. Wybieramy się na wyprawę w góry. Na mapie trasy zaznaczyliśmy n kolejnych miejsc o wysokościach $a_0, a_1, \ldots, a_{n-1}$. Podczas wędrówki chcemy rozmieścić jak największą liczbę flag na szczytach, czyli takich miejscach, dla których dwa sąsiednie miejsca są położone niżej (zakładamy, że pierwsze i ostatnie miejsce nie są szczytami). Jest jedno ograniczenie: jeśli rozmieszczamy k flag, to odległość pomiędzy dwiema dowolnymi flagami powinna być równa co najmniej k.

Przykładowo dla trasy opisanej tablicą $a = [1, 5, 3, 4, 3, 4, 1, 2, 3, 4, 6, 2]$ mamy dokładnie cztery szczyty (zaznaczone na poniższym rysunku):



Na tej trasie możemy rozmieścić 3 flagi (np. na szczytach 1, 5 i 10). Nie możemy jednak rozmieścić 4 flag, tak aby odległość pomiędzy nimi była równa co najmniej 4.

Wynik może być znaleziony przy użyciu wyszukiwania binarnego. Jeśli wiemy, że x flag może zostać rozmieszczonych na szczytach, to również wiemy, że każda ich mniejsza liczba może też być rozmieszczona. Z drugiej strony, jeśli x flag nie może zostać rozmieszczonych, to każda większa liczba flag również nie może być rozmieszczona. W ten sposób, używając wyszukiwania binarnego, redukujemy problem do sprawdzenia, czy można rozmieścić na szczytach ustaloną liczbę x flag. Taki problem możemy rozwiązać już zachłannie: idziemy trasą i zawsze ustawiamy flagę na najbliższym dozwolonym szczycie. Całe rozwiązanie działa w czasie $O(n \log n)$ ze względu na czas wyszukiwania binarnego.

Zadanie da się rozwiązać szybciej, w czasie liniowym. Na początku, dla każdego miejsca, jeśli nie jest szczytem, znajdujemy najbliższy szczyt położony na dalszym odcinku trasy. W tym celu możemy najpierw oznaczyć wszystkie szczyty, a następnie przeglądać trasę w odwróconej kolejności, pamiętając pozycję ostatnio spotkanego szczytu. W ten sposób wypełnimy tablicę $nast$, gdzie $nast[i]$ będzie najbliższym szczytem od miejsca numer i na prawo (lub wartość ∞, jeżeli dalej nie występuje już żaden szczyt). Dla powyższego rysunku mamy $nast = [1, 1, 3, 3, 5, 5, 10, 10, 10, 10, 10, \infty]$.

Flagi powinny znajdować się w odległości równej co najmniej k, więc biorąc k flag możemy rozmieścić ich co najwyżej $\frac{n-1}{k} + 1$. A więc liczba flag, jaką da się rozmieścić nie przekroczy $\lfloor \sqrt{n} + 1 \rfloor$. Przykładowo, dla tablicy $a = [1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1]$ o rozmiarze $n = 15$ da się rozmieścić $k = 4$ flagi.

Zauważmy, że możemy rozmieścić co najwyżej $\lfloor \sqrt{n} + 1 \rfloor$ flag. Ta obserwacja doprowadzi nas do rozwiązania optymalnego. Jeśli ustawiamy jakąś flagę na pozycji p, to wiemy, że następna flaga powinna być ustawiona na pozycji dalszej lub równej $p + k$. Takie miejsce możemy znaleźć w czasie stałym, korzystając z tablicy $nast$. Pseudokod takiego rozwiązania może wyglądać następująco:

```
k := 1;
wynik := 0;
while (k − 1) · k ⩽ n do
    p := 0;
    flagi := 0;
    while p < n and flagi < k do
        p := nast[p];
        if p = ∞ then
            break;
        p := p + k;
        flagi := flagi + 1;
    wynik := max(wynik, flagi);
    k := k + 1;
```

Całkowita liczba operacji nie przekroczy $O(\sqrt{n}^2) = O(n)$, gdyż zarówno zewnętrzna jak i wewnętrzna pętla zostaną wykonane co najwyżej $O(\sqrt{n})$ razy.

Jacek TOMASIEWICZ