

Algorytmy i Struktury Danych

Egzaminy 2016/2017

Termin 1, grupa B

[2pkt.] **Zadanie 1.** Dana jest struktura opisująca listę jednokierunkową przechowującą liczby typu `double`:

```
struct Node { Node* next; double value; };
```

Proszę zaimplementować funkcję `Node* sortList(Node* L)`, która sortuje listę wejściową w kolejności nierosnącej i zwraca wskaźnik na element, który po posortowaniu został głową tej listy (wejściowa lista nie ma wartownika). Funkcja powinna być możliwie jak najszybsza (wiadomo, że liczby w wejściowej liście zostały wygenerowane zgodnie z rozkładem jednostajnym na przedziale $(0,4)$).

[2pkt.] **Zadanie 2.** Dana jest `SkipLista` przechowująca liczby z przedziału $(0,1)$. Proszę opisać (bez implementacji) algorytm, który z takiej `SkipListy` usuwa wszystkie liczby z zadanego przedziału (x, y) (gdzie $0 < x < y < 1$). Proszę oszacować złożoność obliczeniową zaproponowanego algorytmu jako funkcję liczby n elementów w `SkipLiście` (przed operacją usuwania) oraz liczby d elementów, które zostaną usunięte. Proszę uzasadnić przedstawione oszacowanie złożoności. Algorytm powinien być możliwie jak najszybszy. Ocenie podlega poprawność i efektywność algorytmu (1pkt) oraz poprawność oszacowania złożoności czasowej (1pkt).

[2pkt.] **Zadanie 3.** Niecierpliwy Bob ma wykonać k spośród prac J_1, \dots, J_n , gdzie każda praca jest opisana przez czas rozpoczęcia oraz czas zakończenia:

```
struct Job {  
    int start, end; // czas rozpoczęcia i zakończenia (wyrażone w minutach)  
};
```

Bob może wybrać dowolne k prac, byle w jednej chwili nie musiał zajmować się więcej niż jedną. Bob jest niecierpliwy i chce zminimalizować sumę czasu, jaki czeka między wybranymi pracami. Proszę zaimplementować funkcję:

```
int impatientBob( Job J[], int n, int k );
```

która na wejście otrzymuje tablicę n prac (posortowanych rosnąco ze względu na czas zakończenia) i liczbę k , a zwraca minimalną sumę minut, które musi czekać Bob między pracami (lub -1 jeśli nie da się wybrać k niepokrywających się zadań). Proszę skrótkowo opisać wykorzystany algorytm.

Podpowiedź: Można skorzystać z programowania dynamicznego opartego na funkcji $f(i, j) =$ minimalna suma minut, które Bob musi czekać jeśli wybierze j prac spośród J_1, \dots, J_i tak, żeby się nie pokrywały i żeby praca J_i była ostatnią wybraną.

[2pkt.] **Zadanie 4.** Dany jest graf $G = (V, E)$, którego wierzchołki reprezentują punkty nawigacyjne nad Bajtocią, a krawędzie reprezentują korytarze powietrzne między tymi punktami. Każdy korytarz powietrzny $e_i \in E$ powiązany jest z optymalnym pułapem przelotu $p_i \in \mathbb{N}$ (wyrażonym w metrach). Przepisy dopuszczają przelot danym korytarzem jeśli pułap samolotu różni się od optymalnego najwyżej o t metrów. Proszę zaproponować algorytm (bez implementacji), który sprawdza czy istnieje możliwość przelotu z zadanego punktu $x \in V$ do zadanego punktu $y \in V$ w taki sposób, żeby samolot nigdy nie zmieniał pułapu. Algorytm powinien być poprawny i możliwie jak najszybszy. Proszę oszacować jego złożoność czasową.