

Algorytmy i Struktury Danych
Egzamin I (26. VI 2015)

B

Imię i nazwisko: _____

Zadanie 1. (zadania krótkie) Proszę rozwiązać następujące zadania (dla podpunktów 1. i 2. proszę podać sam kod, dla ostatniego podpunktu 2-3 zadania wyjaśnienia):

1. [4pkt.] Proszę zaimplementować funkcję usuwającą z listy wszystkie elementy o podanej wartości x . Lista (bez wartownika) jest opisana przez struktury typu `struct List { List* next; int value };`
2. [4pkt.] Drzewo BST jest opisane przez strukturę `struct BST{ BST *left, *right; int value; };`. Proszę zaimplementować funkcję `double average(BST* T)`, która oblicza średnią wartość elementów w drzewie T .
3. [2pkt.] Pewien algorytm dla danych rozmiaru n najpierw zeruje dwuwymiarową tablicę o rozmiarze $n \times n$ a potem wykonuje obliczenia w czasie $O(n)$ wykorzystujące niektóre z pól tej tablicy (ale trudno przewidzieć które). Łączna złożoność algorytmu to $O(n^2)$. Proszę opisać jak przerobić algorytm tak, by działał w czasie $O(n \log n)$ (możliwe, ale nie wymagane, jest osiągnięcie lepszej złożoności).

[10pkt.] **Zadanie 2.** Dana jest struktura danych opisująca tablicę haszującą, która przechowuje liczby typu `int` indeksowane napisami:

```
struct HT{
    string* key;    // tablica na klucze danych
    int * data;     // tablica na dane
    bool * free;    // pole wolne czy zajete
    int size;       // rozmiar tablicy
};
```

Tablica wykorzystuje funkcję haszującą `int hash(string key, int size)`, która zwraca pozycję w tablicy, na której powinny się znaleźć dane o kluczu `key`. Stosowane jest liniowe rozwiązywanie kolizji. Proszę zaimplementować funkcję `double averageAccess(HT* ht)`, która oblicza jaka jest średnia ilość pól w tablicy, które musi sprawdzić standardowy algorytm wyszukiwujący gdy poszukuje losowo wybranego klucza znajdującego się w tablicy `ht`.

[10pkt.] **Zadanie 3.** Żeglarz Henryk mieszka na wysepce pewnego archipelagu. Wszystkie wyspy w tym archipelagu są tak małe, że można je reprezentować jako punkty w przestrzeni \mathbb{R}^2 . Pozycje wszystkich wysp dane są jako ciąg $W = ((x_1, y_1), \dots, (x_n, y_n))$. Henryk mieszka na wyspie (x_1, y_1) ale chce się przeprowadzić na wyspę (x_n, y_n) . Normalnie, każdego dnia może przepłynąć na wyspę znajdującą się w odległości najwyżej Z (w sensie standardowej odległości euklidesowej), ale może także danego dnia przepłynąć odległość do $2Z$, pod warunkiem, że cały następny dzień będzie odpoczywał. Henryk musi zawsze nocować na jakiejś wyspie. Proszę zaproponować (bez implementacji) wielomianowy algorytm, który oblicza ile minimalnie dni Henryk potrzebuje, żeby dostać się na swoją docelową wyspę (lub stwierdza, że to niemożliwe).

Zadanie 1. (zadania krótkie) Proszę rozwiązać następujące zadania (dla podpunktów 1. i 2. proszę podać sam kod, dla ostatniego podpunktu 2-3 zdania wyjaśnienia):

1. [4pkt.] Proszę zaimplementować funkcję scalającą dwie posortowane listy (obie bez wartownika). Listy są opisane przez struktury typu `struct List { List* next; int value };`
2. [4pkt.] Drzewo BST jest opisane przez strukturę `struct BST{ BST *left, *right; int value; };`. Proszę zaimplementować funkcję `int countInterval(BST* T, int a, int b)`, która oblicza ile liczb z zadanego przedziału domkniętego $[a, b]$ znajduje się w drzewie T .
3. [2pkt.] Pewien algorytm dla danych rozmiaru n działa w czasie $O(n^2)$, a jego złożoność pamięciowa wynosi $O(n^3)$; korzysta z kilku zmiennych lokalnych oraz jednej tablicy rozmiaru $O(n^3)$. W trakcie obliczeń algorytm korzysta z niektórych pól tej tablicy, ale trudno przewidzieć których. Proszę opisać jak przerobić ten algorytm tak, żeby jego złożoność pamięciowa wynosiła $O(n^2)$.

[10pkt.] **Zadanie 2.** Dana jest struktura danych opisująca tablicę haszującą, która przechowuje liczby typu `int` indeksowane napisami:

```
struct HT{
    string* key;    // tablica na klucze danych
    int * data;     // tablica na dane
    bool * free;    // pole wolne czy zajete
    int size;       // rozmiar tablicy
};
```

Tablica wykorzystuje funkcję haszującą `int hash(string key, int size)`, która zwraca pozycję w tablicy, na której powinny się znaleźć dane o kluczu `key`. Stosowane jest liniowe rozwiązywanie kolizji. Niestety możliwe, że tablica zawiera błędne dane. Proszę zaimplementować funkcję `bool checkHT(HT* ht)`, która sprawdza czy dla każdego klucza umieszczonego w tablicy faktycznie możliwe jest jego odszukanie standardowym algorytmem używanym w tablicach haszujących z liniowym rozwiązywaniem konfliktów.

[10pkt.] **Zadanie 3.** Żaba Monika żyje na mokradłach modelowanych jako płaszczyzna \mathbb{R}^2 . Na mokradłach są kamienie umieszczone na pozycjach $(x_1, y_1), \dots, (x_n, y_n)$. Początkowo Żaba Monika siedzi na pierwszym kamieniu (czyli na pozycji (x_1, y_1)) i chce się dostać na ostatni kamień (czyli (x_n, y_n)). W tym celu musi skakać po kamieniach. Maksymalna długość zwykłego skoku Żaby Moniki to L (czyli zwykłym skokiem Żaba Monika może dostać się na dowolny kamień, którego odległość euklidesowa od obecnej pozycji żaby nie przekracza L). Oprócz zwykłych skoków, może ona także wykonywać długie skoki na odległość do $2L$, ale po każdym długim skoku jej kolejny skok musi być na odległość najwyżej $\frac{1}{2}L$. Proszę opisać (bez implementacji) wielomianowy algorytm obliczający minimalną liczbę skoków (nie ważne jakiej długości), które Żaba Monika musi wykonać, żeby dostać się na kamień (x_n, y_n) (lub stwierdza, że to niemożliwe).

ASD – egzamin – II termin – VII.2015

Zadanie 1. (zadania krótkie) Proszę rozwiązać następujące zadania:

1. [5pkt.] Proszę zaimplementować możliwie jak najszybszą funkcję sortującą tablicę n liczb.

2. [5pkt.] Dany jest typ `struct List {List * next; int val; }` realizujący listę jednokierunkową. Proszę zaimplementować funkcję `List * reverse { List *L; }`, która odwraca kierunek listy L (bez wartownika) i zwraca wskaźnik na jej głowę.

[10 pkt.] **Zadanie 2.** Dany jest ważony, nieskierowany graf pełny G zawierający n wierzchołków, reprezentowany przez macierz wag krawędzi. Proszę zaimplementować funkcję

`int MST(int **G, int n)`

która otrzymuje na wejściu ten graf i zwraca sumę wag krawędzi minimalnego drzewa rozpinającego dla G . Funkcja powinna być możliwie jak najszybsza. Można założyć, że dostępna jest struktura:

`struct List { int u, v; int w; };`

oraz funkcja `List *sort { List *L }`, która sortuje niemalejąco podaną listę L według pól w (lista L zawiera wartownika i funkcja `sort` zwraca na niego wskaźnik). Wszystkie pozostałe funkcje i struktury należy zaimplementować.

[10 pkt.] **Zadanie 3.** Dana jest następująca struktura opisująca drzewo:

```
struct Tree{
    Tree *parent;      //rodzic, lub NULL jeśli to korzeń
    Tree *left, *right; //lewe i prawe dziecko
    int w_left, w_right //wagi krawędzi do lewego i do prawego dziecka (dodatnie)
}
```

Proszę opisać (bez implementacji) możliwie jak najszybszy algorytm, który mając na wejściu drzewo opisane przez strukturą `Tree` znajduje długość najdłuższej ścieżki między dwoma węzłami drzewa (prosta ścieżka to taka, która nie odwiedza żadnego węzła więcej niż raz). (Na potrzeby algorytmu mogą państwo uzupełnić strukturę `Tree` o dalsze pola.)