

# Algorytmy i struktury danych

Kolokwia (1)  
algorytmy sortujące

2016/17

A

1. Dana jest struktura opisująca listę jednokierunkową dla liczb rzeczywistych:

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

Proszę zaimplementować funkcję `void Sort( Node* list )`, która otrzymuje na wejściu listę liczb rzeczywistych (z wartownikiem), wygenerowaną zgodnie z rozkładem jednostajnym na przedziale  $[0,10]$  i sortuje jej zawartość w kolejności niemalejącej. Funkcja powinna być możliwie jak najszybsza (biorąc pod uwagę warunki zadania). Proszę oszacować złożoność zaimplementowanej funkcji.

2. Proszę zaimplementować funkcję:

```
int SumBetween(int T[], int from, int to, int n);
```

Zadaniem tej funkcji jest obliczyć sumę liczb z  $n$  elementowej tablicy  $T$ , które w posortowanej tablicy znajdowałyby się na pozycjach o indeksach od `from` do `to` (włącznie). Można przyjąć, że liczby w tablicy  $T$  są parami różne (ale nie można przyjmować żadnego innego rozkładu danych). Zaimplementowana funkcja powinna być możliwie jak najszybsza. Proszę oszacować jej złożoność czasową (oraz bardzo krótko uzasadnić to oszacowanie).

3. Proszę opisać (bez implementacji!) jak najszybszy algorytm, który otrzymuje na wejściu pewien ciąg  $n$  liter oraz liczbę  $k$  i wypisuje najczęściej powtarzający się podciąg długości  $k$  (jeśli ciągów mogących stanowić rozwiązanie jest kilka, algorytm zwraca dowolny z nich). Można założyć, że ciąg składa się wyłącznie z liter `a` i `b`.

Na przykład dla ciągu `ababaaaabb` oraz  $k = 3$  rozwiązaniem jest zarówno ciąg `aba`, który powtarza się dwa razy (to, że te wystąpienia na siebie nachodzą nie jest istotne). Zaproponowany algorytm opisać, uzasadnić jego poprawność oraz oszacować jego złożoność.

2015/16

A

1. Proszę zaimplementować funkcję `void SumSort(int A[], int B[], int n)`. Funkcja ta przyjmuje na wejściu dwie  $n^2$ -elementowe tablice ( $A$  i  $B$ ) i zapisuje w tablicy  $B$  taką permutację elementów z  $A$ , że:

$$\sum_{i=0}^{n-1} B[i] \leq \sum_{i=n}^{2n-1} B[i] \leq \dots \leq \sum_{i=n^2-n}^{n^2-1} B[i].$$

Proszę zaimplementować funkcję `SumSort` tak, by działała możliwie jak najszybciej. Proszę oszacować i podać jej złożoność czasową.

2. Dana jest  $n$  elementowa tablica  $A$  zawierająca liczby naturalne (potencjalnie bardzo duże). Wiadomo, że tablica  $A$  powstała w dwóch krokach. Najpierw wygenerowano losowo (z nieznanym rozkładem)  $n$  różnych liczn nieparzystych i posortowano je rosnąco. Następnie wybrano losowo  $\lceil \log n \rceil$  elementów powstałej tablicy i zamieniono je na losowo wybrane liczby parzyste. Proszę zaproponować (bez implementacji!) algorytm sortowania tak powstałych danych. Algorytm powinien być możliwie jak najszybszy. Proszę oszacować i podać jego złożoność czasową.

3. Dana jest struktura `Node` opisująca listę jednokierunkową:

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

Proszę zaimplementować funkcję `Node* fixSortedList( Node* L )`, która otrzymuje na wejściu listę jednokierunkową bez wartownika. Lista ta jest prawie posortowana w tym sensie, że powstała z listy posortowanej przez zmianę jednego losowo wybranego elementu na losową wartość. Funkcja powinna przeplątać elementy listy tak, by lista stała się posortowana i zwrócić wskaźnik do głowy tej listy. Można założyć, że wszystkie liczby na liście są różne i że lista ma co najmniej dwa elementy. Funkcja powinna działać w czasie liniowym względem długości listy wejściowej.

2014/15

1. Napisać funkcję: `void sortString(string A[])`; Funkcja sortuje tablicę  $n$  stringów różnej długości. Można założyć, że stringi składają się wyłącznie z małych liter alfabetu łacińskiego.
2. Dane są następujące struktury:

```
struct Node { Node* next; int val; };  
struct TwoLists { Node* even; Node* odd; };
```

Napisać funkcję: `TwoLists split(Node* list)`;

Funkcja rozdziela listę na dwie: jedną zawierającą liczby parzyste i drugą zawierającą liczby nieparzyste. Listy nie zawierają wartowników.

3. Jak posortować  $n$ -elementową tablicę liczb rzeczywistych, które przyjmują tylko  $\log n$  różnych wartości? Uzasadnić poprawność algorytmu i oszacować złożoność. (Nie trzeba implementować).

2012/13

1. Proszę zaimplementować funkcję sortującą (rosnąco) listę jednokierunkową metodą QUICKERSORT. Algorytm QUICKERSORT to odmiana algorytmu QUICKSORT, w której funkcja podziału dzieli sortowane dane według przyjętej wartości  $x$  na trzy grupy: mniejsze od  $x$ , równe  $x$ , oraz większe od  $x$ . Następnie rekurencyjnie sortowane są grupy pierwsza i trzecia. Państwa funkcja powinna mieć następujący prototyp:

```
struct Node { Node* next; int val; };  
Node* QuickerSort ( Node* head )
```

Argumentem funkcji jest wskaźnik na głowę listy do posortowania a wynikiem powinien być wskaźnik na głowę listy posortowanej. Sortowanie powinno polegać na porównywaniu wartości `val` list oraz przepinaniu wskaźników `next`.

2. Proszę zaprojektować strukturę danych przechowującą liczby i pozwalającą na następujące operacje (zakładamy, że wszystkie liczby umieszczane w strukturze są różne):

**Init( $n$ ).** Tworzy zadaną strukturę danych zdolną pomieścić maksymalnie  $n$  liczb.

**Insert( $x$ ).** Dodaje do struktury liczbę  $x$ .

**RemoveMin()** Znajduje najmniejszą liczbę w strukturze, usuwa ją i zwraca jej wartość.

**RemoveMax()** Znajduje największą liczbę w strukturze, usuwa ją i zwraca jej wartość.

Każda z operacji powinna mieć złożoność  $O(\log n)$ , gdzie  $n$  to ilość liczb znajdujących się obecnie w strukturze. W tym zadaniu nie trzeba implementować podanych operacji, a jedynie przekonująco opisać jak powinny być zrealizowane i dlaczego mają wymaganą złożoność.

3. Proszę napisać funkcję `bool possible( char* u, char* v, char* w )`, która zwraca prawdę jeśli z liter słów  $u$  i  $v$  da się ułożyć słowo  $w$  (nie jest konieczne wykorzystanie wszystkich liter) oraz fałsz w przeciwnym wypadku. Można założyć, że  $w$  i  $v$  składają się wyłącznie z małych liter alfabetu łacińskiego. Proszę krótko uzasadnić wybór zaimplementowanego algorytmu.