ASD - ćwiczenia IV

Zadanie o „uproszczonym“ sortowaniu

Dana jest tablica $A[1], A[2], \ldots, A[n]$ n parami różnych liczb całkowitych. Zaproponuj możliwie efektywną funkcję

`int SORT(int A[], int n)`,

częściowego sortowania tej tablicy, tak aby dla pewnego indeksu $k \in \{1, 2, \ldots, n\}$ zachodziło:

$$\forall i \in \{1, 2, \ldots, k\} \ : \ A[i] < 0 \ \wedge \ \forall j \in \{k+1, k+2, \ldots, n\} \ : \ A[j] \geq 0.$$

Rezultatem funkcji `SORT()` powinien być indeks k, spełniający powyższe warunki. Zakładamy, że $\exists i \in \{1, 2, \ldots, n\} \ : \ A[i] < 0$.

Zadanie o piłkarzach jednakowego wzrostu

W turnieju piłkarskim występuje n drużyn $T[1], T[2], \ldots, T[n]$, z których w każdej gra dokładnie n zawodników $T[i] = \{P[i][1], P[i][2], \ldots, P[i][n]\}$, gdzie $n = 2^k$ dla $k \in \mathbb{N} \setminus \{0, 1, 2, 3\}$. Każdego zawodnika biorącego udział w turnieju zmierzono z dokładnością do jednego milimetra. Zaprojektuj możliwie efektywną funkcję

`int EQUAL(Player P[][], int n)`,

która wyznaczy maksymalną liczbę zawodników o tym samym wzroście. Zakładamy, że macierz T zmiennych typu `Player` składa się z n wierszy (odpowiadających drużynom) oraz n kolumn (odpowiadających zawodnikom), gdzie:

```
typedef str_Player Player;

struct str_Player {
  real heigth
};
```

Rozważ dwa warianty danych wejściowych:

- zawodnicy w każdej drużynie $T[i]$, dla $i = 1, 2, \ldots, n$ nie są uporządkowani ze względu na wzrost,
- zawodnicy w każdej drużynie $T[i]$, dla $i = 1, 2, \ldots, n$ są uporządkowani rosnąco ze względu na wzrost, tj. $\forall k = 1, 2, \ldots, n-1 \ : \ P[i][k].height \leq P[i][k+1].height$.

Który z wariantów problemu można rozwiązać asymptotycznie szybciej?

Zadania dodatkowe

1. Napisz kod procedury

   `TOURNAMENT_SORT(int A[])`,

   która sortuje w kolejności niemalejącej zadaną n-elementową tablicę, parami równych liczb naturalnych, $A[]$ zgodnie z poniższym schematem:

   (a) za pomocą algorytmu „Turniej“ wyszukuję w tablicy $A[]$ element największy i drugi co do wielkości,

   (b) ustawiam te dwa elementy na końcu tablicy,

(c) powtarzam czynność dla pozostałego fragmentu tablicy.

Oszacuj pesymistyczny i średni koszt zaproponowanego rozwiązania, przyjmując, że operacją dominującą jest czynność porównywania elementów tablicy. Jaka jest jego złożoność pamięciowa?

2. Załóżmy, że pewien algorytm $\mathcal{A}$, dla danego ciągu wejściowego S rozmiaru n, składa się z czterech części $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ oraz $\mathcal{A}_4$:

   - $\mathcal{A}_1$ - sortowania nierosnącego ciągu S za pomocą algorytmu sortowania przez scalanie,
   - $\mathcal{A}_2$ - sortowania niemalejącego ciągu S za pomocą algorytmu sortowania przez wybór,
   - $\mathcal{A}_3$ - sortowania nierosnącego ciągu S za pomocą algorytmu sortowania szybkiego,
   - $\mathcal{A}_4$ - sortowania niemalejącego ciągu S za pomocą algorytmu sortowania przez wstawianie.

   Podaj dokładne oszacowanie złożoności całego algorytmu w przypadku oczekiwanym (średnim) oraz pesymistycznym jeżeli:

   (a) operacją dominującą jest porównanie elementów ciągu S,

   (b) operacją dominującą jest transpozycja elementów ciągu S.

3. Podaj przykład n-elementowego ciągu wejściowego, dla którego złożoność algorytmu sortowania przez wstawianie jest istotnie mniejsza niż złożoność algorytmu sortowania przez selekcję (wybór). Jak zachowa się dla wybranych danych wejściowych algorytm sortowania szybkiego?