

## ALGORYTMY I STRUKTURY DANYCH

IHUWr. II rok informatyki.

1. (1pkt) Niech  $\sigma$  będzie ciągiem instrukcji *Union* i *Find*, w którym wszystkie instrukcje *Union* występują przed instrukcjami *Find*. Udowodnij, że algorytm oparty na strukturach drzewiastych wykonuje  $\sigma$  w czasie proporcjonalnym do długości  $\sigma$ .
2. (2pkt) Rozważamy ciągi operacji *Insert*( $i$ ), *DeleteMin* oraz *Min*( $i$ ) wykonywanych na  $S$  - podzbiórze zbioru  $\{1, \dots, n\}$ . Obliczenia rozpoczynamy z  $S = \emptyset$ . Instrukcja *Insert*( $i$ ) wstawia liczbę  $i$  do  $S$ . Instrukcja *DeleteMin* wyznacza najmniejszy element w  $S$  i usuwa go z  $S$ . Natomiast wykonanie *Min*( $i$ ) polega na usunięciu z  $S$  wszystkich liczb mniejszych od  $i$ .

Niech  $\sigma$  będzie ciągiem instrukcji *Insert*( $i$ ), *DeleteMin* oraz *Min*( $i$ ) takim, że dla każdego  $i$ ,  $1 \leq i \leq n$ , instrukcja *Insert*( $i$ ) występuje co najwyżej jeden raz. Mając dany ciąg  $\sigma$  naszym zadaniem jest znaleźć ciąg liczb usuwanych kolejno przez instrukcje *DeleteMin*. Podaj algorytm rozwiązujący to zadanie.

UWAGA: Zakładamy, że cały ciąg  $\sigma$  jest znany na początku, czyli interesuje nas wykonanie go *off-line*.

WSKAZÓWKA: Rozdział 4.8 z książki Aho, ... .

3. (2pkt) Rozważamy ciągi instrukcji: *Link*( $r, v$ ) oraz *Depth*( $v$ ) wykonywanych na lesie rozłącznych drzew o wierzchołkach z etykietami ze zbioru  $\{0, \dots, n-1\}$  (różne wierzchołki mają różne etykiety). Operacja *Link*( $r, v$ ) czyni  $r$ , korzeń jednego z drzew, synem  $v$ , wierzchołka innego drzewa. *Depth*( $v$ ) oblicza głębokość wierzchołka  $v$ .

Naszym celem jest napisanie algorytmu, który dla danego ciągu  $\sigma$  wypisze w sposób on-line wyniki instrukcji *Depth* (tzn. wynik każdej instrukcji *Depth* ma być obliczony przed wczytaniem kolejnej instrukcji z ciągu  $\sigma$ ). Pokaż jak zastosować drzewiastą strukturę danych dla problemu *Union* – *Find* do rozwiązania tego problemu.

WSKAZÓWKA: Rozdział 4.8 z książki Aho, ... .

4. (1pkt) Rozważ taką wersję wykonywania kompresji ścieżek, w której wierzchołki wizytowane podczas wykonywania operacji *Find* podwieszane są pod własnego dziadka. Czy analiza złożoności przeprowadzona na wykładzie da się zastosować w tym przypadku?
5. (2pkt) Załóżmy, że wzorec  $P$  może zawierać znak  $\diamond$  (tzw. gap character). Znak ten jest zgodny z dowolnym podsłowem (także z podsłowem pustym). Na przykład, wzorec  $ab\diamond ba\diamond c$  występuje w słowie  $cabccbacbacab$  jako

$$c \overbrace{ab}^{ab} \underbrace{cc}_{\diamond} \overbrace{ba}^{ba} \underbrace{cba}_{\diamond} \underbrace{c}_{c} \overbrace{ab}^{ab}$$

a także jako

$$c \overbrace{ab}^{ab} \underbrace{ccbac}_{\diamond} \overbrace{ba}^{ba} \underbrace{\quad}_{\diamond} \underbrace{c}_{c} \overbrace{ab}^{ab}.$$

Podaj algorytm znajdujący wystąpienie takiego wzorca w danym tekście  $T$  (oczywiście zakładamy, że  $\diamond$  nie występuje w  $T$ ).

6. (1pkt) Podaj algorytm, który w czasie liniowym określa, czy tekst  $T$  powstał przez przesunięcie cykliczne tekstu  $T'$ .