

## ALGORYTMY I STRUKTURY DANYCH

IIUWr. 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) Instrukcja *Delete(i)* usuwa element  $i$  ze zbioru, do którego on aktualnie należy i tworzy jednoelementowy zbiór  $\{i\}$ . Podaj implementację instrukcji *Delete(i)*, która pozwoli na wykonywanie ciągów złożonych z  $n$  instrukcji *Union*, *Find* i *Delete* w czasie  $O(n \log^* n)$ .
3. (2pkt) Rozważamy ciągi operacji *Insert(i)*, *DeleteMin* oraz *Min(i)* wykonywanych na  $S$  - podzbiorze 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, ... .

4. (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, ... .

5. (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? Czy widzisz jakąś zaletę takiej kompresji ścieżek w stosunku do oryginalnej metody?
6. (1pkt) Opisz, jak skonstruować automat, który dla dwóch danych wzorców  $P$  i  $P'$  znajduje wszystkie ich wystąpienia w tekście. Spróbuj zminimalizować liczbę stanów w Twoim automacie.
7. (2pkt) Załóżmy, że wzorec  $P$  może zawierać znak  $\diamond$  (tzw. gap character). Znak ten jest zgodny z dowolnym podśłowem (także z podśłowem pustym). Na przykład, wzorec  $ab\diamond ba\diamond c$  występuje w słowie *cabccbacbacab* jako

$$\begin{array}{ccccccc} c & ab & cc & ba & cba & c & ab \\ & \underbrace{\hspace{1cm}} & & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \\ & ab & \diamond & ba & \diamond & c & \end{array}$$

a także jako

$$\begin{array}{ccccccc} c & ab & ccba & ba & & c & ab \\ & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \\ & ab & \diamond & ba & \diamond & c & \end{array}$$

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

8. (1pkt) Podaj algorytm, który w czasie liniowym określa, czy tekst  $T$  powstał przez przesunięcie cykliczne tekstu  $T'$ .