Algorytmy i Struktury Danych, 10. ćwiczenia

2021-12-08

Zadanie 8.1

Oto ulepszony algorytm BubbleSort sortujący niemalejąco n-elementową tablicę $a[1, \ldots, n]$:

```
i := n+1;
repeat
    i := i - 1;
    posortowany := true;
    for j := 2 to i do
    if a[j] < a[j-1] then begin
        a[j-1] :=: a[j]; { zamiana elementów }
        posortowany := false
    end
until posortowany;
```

Zaprojektuj efektywny algorytm, który dla tablicy a zawierającej permutację liczb $1, 2, \ldots, n$ obliczy ile razy algorytm BubbleSort wykona pętlę **repeat**.

Rozwiązanie: Zauważmy, że element będzie brał udział w zamianie (jako prawy element) jeśli po jego lewej stronie będzie istniał jakiś większy co do wartości element.

$$\Phi = 1 + \max_i |\{j : j < i \text{ and } A[j] > A[i]\}|$$

Gdyby tablica A zawierała dowolne wartości to Φ można obliczyć w czasie $O(n \log n)$ wstawiając kolejne elementy A do drzewa AVL.

Jednak gdy A to permutacja liczb $1..n$, to formuła się upraszcza:

$$\Phi = 1 + \max_{i \in 1,\ldots,n} (i - A[i])$$

(krytyczna wartość i ma na prawo tylko elementy większe – inaczej nie była by optymalna).

Zadanie 8.2

a) Do początkowo pustego AVL-drzewa wstawiamy kolejno klucze $1, 2, \ldots, n$, gdzie n jest dodatnią liczbą całkowitą. Podaj klucze, które kiedykolwiek znajdą się w korzeniach kolejno powstających drzew.

b) Jaka jest minimalna, a jaka maksymalna liczba węzłów (wewnętrznych) w AVL-drzewie o wysokości h?

c) Załóżmy, że kluczami w AVL drzewie są kolejne liczby naturalne $1, 2, \ldots$. Jaki najmniejszy, a jaki największy klucz może znaleźć się w korzeniu AVL-drzewa o wysokości h?

Rozwiązanie:

a) 2^i dla $i = 0, \ldots, \lfloor \log n \rfloor$. Dowód przez indukcję na prawym poddrzewie.

b) $T_{min}(h)$ drzewo AVL o wysokości h i minimalnej liczbie węzłów, $T_{min}(0) = 0$, $T_{min}(1) = 1$, $T_{min}(2) = 2$, $T_{min}(3) = 4$, $T_{min}(4) = 7$, $T_{min}(5) = 12$, $T_{min}(6) = 20$, $T_{min}(h) = T_{min}(h-1) + T_{min}(h-2) + 1 = F_n - 1$, $T_{max}(h)$ drzewo AVL o wysokości h i maksymalnej liczbie węzłów, $T_{max}(h) = 2^h - 1$,

T_{min} to ciąg `https://oeis.org/A000071`

c) w korzeniu mogą być wartości z zakresu $[T_{min}(h-1)+1, 2^{h-1}]$.

Zadanie 8.3

Zaproponuj implementacje operacji Join i Split dla drzew typu "splay".

Rozwiązanie:

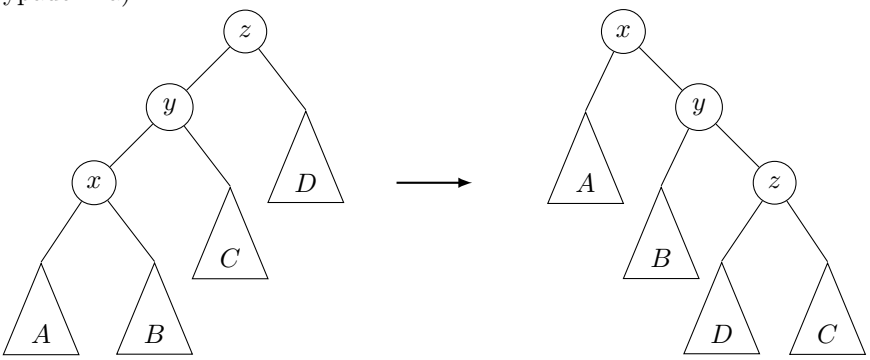
Split(x):: wykonaj operację Splay(x) i usuń korzeń, otrzymując lewe i prawe poddrzewo,

Join(T_x, T_y):: wykonaj operację Splay(max(T_x)) a następnie podłącz T_y jako prawe poddrzewo max(T_x).

Zadanie 8.4

Dokonaj analizy kosztu zamortyzowanego operacji LocalSplay w przypadku 2a (patrz wykład).

Rozwiązanie:
Przypadek 2a)



$$\Delta\Phi = m'(z) + m'(y) + m'(x) - m(x) - m(y) - m(z)$$

Ponieważ $m'(x) = m(z)$:

$$\Delta\Phi = m'(z) + m'(y) - m(x) - m(y)$$

Analogicznie do rozwiązania wykładu przypadku 2b. Definiujemy tak samo a i b ($a = |A| + |B| + 1$ i $b = |C| + |D| + 1$), tylko używamy $\log a = m_x$, $\log b = m'_z$ i $\log(a + b + 1) = m_z = m'_x$.

Zadanie 8.5

Zaproponuj implementacje operacji Join i Split dla 2-3-4 drzew.

Rozwiązanie: Join i Split na 2-3-4 drzewach: `http://courses.csail.mit.edu/6.046/spring04/handouts/ps5-sol.pdf`
Ogólnie o 2–3–4 drzewach: `https://www.cs.purdue.edu/homes/ayg/CS251/slides/chap13b.pdf`

Zadanie 8.6

Zaprojektuj algorytm, który sprawdza, czy wierzchołki danego drzewa binarnego można pokolorować w taki sposób, żeby otrzymać drzewo czerwono-czarne i jeśli odpowiedź jest tak, to znajduje takie pokolorowanie

Rozwiązanie:
Zdefiniujmy następujące wartości:

$h(v, B)$ oznacza zbiór możliwych czarnych wysokości drzewa czerwono-czarnego o korzeniu v przy założeniu, że v jest czarny,

$h(v, R)$ oznacza zbiór możliwych czarnych wysokości drzewa czerwono-czarnego o korzeniu v przy założeniu, że v jest czerwony.

Z własności drzew czerwono-czarnych wiemy, że $|h(v, R/B)| = O(\log n)$.
Wzory rekurencyjne na obliczanie h:

- $h(NULL, B) = \{1\}$, $h(NULL, R) = \emptyset$,
- $h(v, R) = h(left(v), B) \cap h(right(v), B)$
- dla $h(v, B)$:
  - $L = \{x + 1 : x \in h(left(v), R) \cap h(left(v), B)\}$
  - $R = \{x + 1 : x \in h(right(v), R) \cap h(right(v), B)\}$
  - $h(v, B) = L \cap R$.

Co daje rozwiązanie $O(n \log n)$. Jeśli jednak zauważmy, że $h(v, B)$ i $h(v, R)$ stanowią spójny przedział lub zbiór pusty, to możemy obliczyć wszystkie wartości w $O(n)$.