Autor: Krystian Jasionek

6. (2pkt) Rozważmy problem wyznaczenia za pomocą porównań elementów największego i drugiego z kolei w zbiorze n-elementowym. Udowodnij, że $n + \lceil \log n \rceil - 2$ porównań potrzeba i wystarcza do wyznaczenia tych elementów.

Chcemy pokazać, że do wyznaczenia tych dwóch elementów musimy wykonać $n + \lceil \log n \rceil - 2$ porównań, oraz że jeśli wykonamy tyle porównań, to mamy pewność, że je wyznaczymy.

Dowód, że wystarcza $n + \lceil \log n \rceil - 2$ porównań

Zauważmy, że wyszukanie maksimum można zrealizować w $n-1$ porównaniach. Wystarczy rozpatrzyć turniej, w którym porównujemy ze sobą pary elementów, a do kolejnej rundy przechodzą zwycięzcy porównań. Łatwo zauważyć, że do kolejnej rundy przechodzi zawsze $\lceil \frac{n}{2} \rceil$ elementów (dla nieparzystej liczby elementów jeden czeka i przechodzi do kolejnej), a turniej skończy się, gdy pozostanie nam 1 element - maksimum. Widzimy stąd, że turniej musi składać się z $\lceil \log_2 n \rceil$ rund.

Zauważmy, że w wyniku turnieju otrzymujemy pełne drzewo binarne, ponieważ każdy wierzchołek (porównanie) ma zero lub dwoje dzieci (elementów porównywanych) - albo porównujemy dwa elementy, albo przenosimy jeden do kolejnej rundy bez porównań. n elementów porównywanych w pierwszej rundzie stanowi liście drzewa.

W takim razie wiemy, że liczba wierzchołków drzewa to $2n-1$, a zatem mamy $n-1$ porównań.

Lemat 1

Dla dowolnego drzewa n-arnego, którego wszystkie wierzchołki mają 0 lub n dzieci zachodzi:

$$L = (n-1) \cdot J + 1, \text{ gdzie } L \text{ - liczba liści}$$
$$J \text{ - liczba wierzchołków wewnętrznych}$$

Dowód

Weźmy drzewo n-arne. Załóżmy, że jest ono zawiera T wierzchołków. Zauważmy, że $T = L + J$. Takie drzewo ma oczywiście $T-1$ krawędzi (z własności drzew). W takim razie każdy wierzchołek wewnętrzny ma n krawędzi, czyli wszystkie wierzchołki wewnętrzne mają $n \cdot J$ wierzchołków. Zachodzą wtedy następujące równości:

1) $n \cdot J = T - 1$ - każda krawędź wchodzi jest krawędzią wewnętrzną
2) $L + J = T$ - suma liści i wierzchołków wewnętrznych to suma wierzchołków w drzewie

Otrzymujemy stąd, że:

Uszczególniając dla $n=2$

2) $L + J - 1$

Otrzymujemy stąd, że:

$$\begin{cases} n \cdot J = T - 1 \\ L + J = T \end{cases} \Rightarrow n \cdot J = L + J - 1$$

$$\underline{L = (n-1)J + 1} \quad \blacksquare$$

Uszczególnienie dla $n=2$

$L = J + 1$
$J = L - 1$
$T = 2L - 1$

Skoro mamy $n-1$ porównań, to zauważmy, że zwycięzca turnieju – korzeń drzewa – jest elementem największym, czyli wystarczy $n-1$ porównań, by wyznaczyć maksimum.

Zauważmy też, że chcąc wyznaczyć drugi co do wielkości element, nie musimy wykonać $n-2$ porównań dla pozostałych elementów – przeprowadzić kolejnego turnieju. Wystarczy obserwacja, że drugi co do wielkości element mógł zostać wybrany tylko z tych elementów, które przegrały porównanie z maksimum i wygrały wszystkie pozostałe. W każdej rundzie maksimum mogło być porównane tylko raz, zatem takich elementów jest $\lceil \log n \rceil - 1$, bo nie musimy porównywać elementu z nim samym.

Łącznie wystarczy nam więc $n + \lceil \log n \rceil - 2$ porównań, by wyznaczyć te dwa elementy. Czy nie możemy jednak zrobić tego, wykonując mniej porównań? Pokażemy, że nie.

Dowód, że potrzeba $n + \lceil \log n \rceil - 2$ porównań

Przeprowadzimy dowód poprzez argument o adwersarzu. Rozważmy adwersarza, który stara się tak kierować poczynaniami algorytmu, by ten zawsze wykonywał maksymalną liczbę operacji – $n + \lceil \log n \rceil - 2$ – z kolei algorytm dąży do wykonania minimalnej liczby porównań. Pokażemy, że strategia adwersarza jest zawsze wygrywająca, czyli algorytm potrzebuje wykonać $n + \lceil \log n \rceil - 2$ porównań.

Strategia adwersarza

Niech adwersarz przydziela wagi każdemu wierzchołkowi drzewa, są to wagi, które algorytm wykorzystuje do porównywania elementów. Zaczniemy od ustalenia wagi każdego liścia na 1. Zauważmy, że suma wag liści wynosi n. Adwersarz traktuje tę sumę jako niezmiennik i dąży do tego, by w każdej iteracji algorytmu suma wag wynosiła n.

Postępowanie adwersarza jest następujące:

Algorytm porównuje wierzchołki x i y (zadając pytanie adwersarzowi):

1) $\underline{W(x) \geq W(y)}$, wtedy adwersarz mówi, że $\underline{x > y}$ i ustawia $\underline{W'(x) = W(x) + W(y),\ W'(y) = 0}$,

2) w przeciwnym przypadku adwersarz mówi cokolwiek, co jest zgodne z obecnymi relacjami elementów i nie zmienia ich wag.

2) w przeciwnym przypadku adwersarz mówi cokolwiek, co jest zgodne z obecnymi relacjami elementów i nie zmienia ich wag.

Zauważmy, że mamy następujące konsekwencje:

1) $W(x)=0$, jeśli x przegrało porównanie,

2) $W(x)>0$, jeśli x jeszcze nie przegrało i ciągle jest kandydatem na maksimum,

3) suma wag zawsze wynosi n;

4) algorytm kończy działanie, gdy znajdzie się taki element x, że $W(x)=n$, wtedy wszystkie inne elementy przegrały z nim porównanie.

Zauważmy, że skoro waga tego maksymalnego x zmieniła się od 1 do n, a w każdym porównaniu mogła co najwyżej się podwoić $(W(x) \geq W(y) \to W'(x)=W(x)+W(y))$, to musiało zajść co najmniej $\lceil \log n \rceil$ porównań elementu maksymalnego. Od razu dostajemy też liczbę kandydatów na drugi co do wielkości element — $\lceil \log n \rceil$. Wśród nich w podobny sposób szukamy maksimum.

Algorytm musi wyzerować $n-1$ wag, zatem musi wykonać $n-1$ porównań. Musi też wykonać $\lceil \log n \rceil - 1$ porównań, by wyzerować wagi $\lceil \log n \rceil - 1$ kandydatów na drugi co do wielkości element.

Dostajemy, że algorytm musi wykonać co najmniej $n+\lceil \log n \rceil - 2$ porównań, by znaleźć dwa największe elementy.