1 Lista 4, Zadanie 6

Mając już funckję $DualPivotPartition(A, p, r)$ zwracającą parę k_1, k_2 będącą, odpowiednio, pozycją pierwszego i drugiego pivota, wybranych losowo, należy napisać algorytm $DualPivotRandomSelect(A, p, r, i)$ znajdujący i-tą statystykę pozycyjną. Będę bazował na algorytmie $RANDOMIZED\!-\!SELECT$ z książki "Wprowadzenie do Algorytmów" Cormen'a. (rozdz. 9.2)

Algorithm 1: DualPivotRandomSelect(A, p, r, i)

Data: A - tablica liczb, p - indeks pierwszego elementu, r - indeks ostatniego elementu, i - szukana statystyka pozycyjna
Result: A[i] - wartość szukanej statystyki pozycyjnej

```
if p = r then
    return A[p]

k1, k2 = DualPivotPartition(A, p, r)
firstsize = k1 − p + 1
secondsize = k2 − p + 1

if i = firstsize then
    return A[k1]

if i = secondsize then
    return A[k2]

if i < firstsize then
    return DualPivotRandomSelect(A, p, k1 − 1, i)
else if i > firstsize & i < secondsize then
    return DualPivotRandomSelect(A, k1 + 1, k2 − 1, i − firstsize)
else
    return DualPivotRandomSelect(A, k2 + 1, r, i − secondsize)
```

W porównaniu do klasycznego $RANDOMIZED - SELECT$, tutaj dzielimy tablicę na 3 pod-tablice i zajmujemy się wyłącznie jedną z nich, w naturalny sposób jest to szybsze od zajmowania się jedną z dwóch.

Sam $DualPivotRandomSelect$ dokonuje porównań wyłącznie na indekscach tablicy i i. Natomiast porównania dokonywane są w procedurze $DualPivotPartition$. Przypadek przesymistyczny podziału, zachodziłby gdy procedura tworzyłaby jeden obasz złożony z $n - 2$ elementów:

$$T(n) = T(n-2) + T(0) + T(0) + \Theta(n) \tag{1}$$

Przypadek optymistyczny polegałby na ciągłym podziale na równe części, to jest:

$$T(n) = 3T(\frac{n}{3}) + \Theta(n) \tag{2}$$

Przypadek oczekiwany (życiowy) polegałby na podziale na zrównoważone części:

$$T(n) = T(\frac{a_1 n}{\alpha}) + T(\frac{a_2 n}{\alpha}) + T(\frac{a_3 n}{\alpha}) + \Theta(n) \tag{3}$$

przy czym:

$$a_1 + a_2 + a_3 = \alpha$$

Analiza tego przypadku pokazuje, że dla niepatologicznych przypadków procedura *Partition* (i jej wariacje, oparte o ten sam kręgosłup) są $O(n \log n)$ ("Wprowadzenie do Algorytmów" - rozdz. 9.2).

1.1 Liczba porównań

Jeżeli chodzi natomiast o liczbę porównań pomiędzy elementami tablicy, wszystko zależy od implementacji procedury *DualPivotPartition*, oraz, przede wszystkim, od losowo wybranych pivotów. Natomiast załóżmy, że szukana statystyka znajduje się zawsze w najdłuższej pod-tablicy. W *DualPivotPartition* wybieramy dwa pivoty na $\binom{n}{2}$ sposobów, czyli szansa wybrania jednego to $\frac{1}{\binom{n}{2}} = \frac{2}{n(n-1)}$, ponadto liczba w pojedynczym przejściu po tablicy wynosi $O(n)$. Załóżmy, że $k_1 < k_2$. Najdłuższa z tablic to $max(k_1, k_2 - k_1 + 1, n - k_2 + 1)$. Zatem:

$$T(n) \leq \frac{2}{n(n-1)} \left(\sum_{k_1=1}^{n-1} \sum_{k_2=k_1+1}^{n} T(max(k_1, k_2 - k_1 + 1, n - k_2 + 1)) \right) + O(n) \tag{4}$$