ASD—4

1 Wybór elementu k-tego co do wielkości

Wybór elementu k-tego co do wielkości jest zadaniem pokrewnym sortowaniu. Załóżmy, że dany ciąg ma n elementów. Można wyznaczyć k-ty co do wielkości element w tym ciągu w następujący spsosób: najpierw sortujemy ciąg w porządku niemalejącym, a następnie z otrzymanego ciągu wybieramy k-ty element. Wiemy już, że takie postępowanie daje czas $n \log n$. Stosując w specjalny sposób strategię "dziel i zwyciężaj", możemy jednak wyznaczyć k-ty co do wielkości element w ciągu za pomocą tylko $O(n)$ kroków. Zaczniemy od algorytmu, który w oczekiwanym czasie liniowym znajduje element k-ty co do wielkości.

Aby znaleźć k-ty najmniejszy element w ciągu q (w zastosowaniach często $k = \lceil n/2 \rceil$ - przypadek mediany) możemy zastosować podobną metodę jak w algorytmie QuickSort. Wejściowy ciąg dzielimy na dwa podciągi względem pewnego elementu w ciągu. W zależności od liczebności podciągów, k-tego elementu szukamy albo w lewym albo w prawym podciągu. Algorytm ten nosi nazwę algorytmu Hoare'a.

```
long select(long *a, int l, int r, int k)
/* 1<=k<=r-l+1;  instancja  wyznacza
   k-ty  najmniejszy  element  w tablicy a[l..r];
   zewnetrzne wywolanie select(0,n-1,k) */
{
  int j;

  if (r > l){
    j = partition(a,l,r);
    if (k == (j-l+1))
      return a[j];
    else if (k <= (j-l))
      return select(a,l,j-1,k);
    else
      return select(a,j+1,r,k-(j-l+1));
  }else
     return a[l];
}
```

Ponieważ wywołanie `select(a,l,r,k)` pociąga za sobą co najwyżej jedno wywołanie rekurencyjne i to jako ostatnią akcję procedury, więc rekursję można w sposób naturalny zastąpić iteracją - zapisanie algorytmu iteracyjnego pozostawiamy jako ćwiczenie na laboratorium.

Równanie na pesymistyczną złożoność czasową jest następujące:

$W(1) = 0$;
$W(n) = (n+1) + W(n-1)$, dla $n > 1$.
Rozwiązaniem jest $W(n) = \frac{1}{2}n^2 + O(n)$.
Pesymistyczna wrażliwość algorytmu na dane wejściowe wynosi $\Delta(n) = O(n^2)$ (w przypadku optymistycznym wykonują się tylko $n+1$ porównania).

Równanie rekurencyjne na oczekiwaną złożoność czasową w modelu permutacyjnym jest następujące:
$A(1) = 0$;
$A(n) = n + 1 + \frac{1}{n}\Sigma_{j=k+1}^{n} A(j-1) + \frac{1}{n}\Sigma_{j=1}^{k-1} A(n-j)$, dla $n > 1$.
Przez indukcję względem wartości n można udowodnić, że $A(n) \leq 4n$. Iteracyjna wersja algorytmu Hoare'a działa w miejscu — $S(n) = O(1)$. Algorytm Hoare'a, podobnie jak QuickSort, jest szybkim algorytmem dla losowych danych i tak jak QuickSort ma pesymistyczną złożoność czasową $O(n^2)$.

Przedstawimy teraz szkic liniowego algorytmu na wyznaczanie k-tego co do wielkości elementu w ciągu (Algorytm Bluma-Floyda-Pratta-Rivesta-Tarjana). Jego zasada działania jest taka jak algorytmu Hoare'a. Różnica polega na specjalnym, wyważonym doborze elementu dzielącego tak, aby każda z części miała rozmiar nie większy od 3/4 rozmiaru ciągu ulegającego podziałowi.

```
long select(int k, ciag q)
/* q=[a_1,a_2,...,a_n] */
{
  if (n<50)
  {
    posortuj q;
    return k-ty element w ciagu q;
  }
  else{
    podziel q na [n/5] podciagow 5-elementowych i
    (ewentualnie) jeden ciag co najwyzej 4-elementowy;
    posortuj osobno kazdy podciag;
    niech M bedzie ciagiem median tych podciagow;
    m = select(sufit(|M|/2),M);
    niech q_1, q_2, i q_3 beda ciagami elementow z q,
    ktore sa odpowiednio <, = i > m;
    if (k <= |q_1|)
      return select(k,q_1);
    else if ((|q_1|+|q_2|) >= k)
      return m;
    else
      return select(k-|q_1|-|q_2|,q_3);
  }
}
```

Otrzymujemy nastepujące równanie rekurencyjne na pesymistyczną złożoność czasową powyższego algorytmu.
$W(n) \leq c_1 n$, dla $n < 50$, i
$W(n) \leq W(|M|) + \max(W(|q_1|), W(|q_3|)) + c_2 n$, dla $n \geq 50$,
gdzie c_1 i c_2 są pewnymi stałymi.

Przez indukcję dowodzi się, że $W(n) \leq 20cn$, gdzie $c = max(c_1, c_2)$. Dla liniowości rozwiązania równania rekurencyjnego jest istotne, że $\frac{1}{5} + \frac{3}{4} < 1$. Podział ciągu na 5-elementowe podciągi jest związany z tym warunkiem.