Rozwiązanie zadania 2. z listy 4.

21 kwietnia 2020

1 Treść zadania

Przyjmij, że dany jest algorytm w postaci "czarnej skrzynki", który wyznacza medianę w pesymistycznym przypadku w czasie liniowym. Zaprojektuj algorytm, który używając tej "czarnej skrzynki" wyznacza dowolną statystykę pozycyjną w czasie liniowym.

2 Rozwiązanie

Przyjmijmy, że $median([...])$ jest funkcją realizującą algorytm z treści zadania. Gdy dostanie ona jako argument pewną listę, zwraca wartość będącą medianą podanej listy. Przeanalizujmy poniższy pseudokod:

function KTH_ORDER_STATISTICS($[a_1, a_2, ..., a_n], k$)
  if $k > n$ **then return** ERROR
  end if

  $m \leftarrow median([a_1, ..., a_n])$
  $p \leftarrow partition([a_1, ..., a_n], m)$

  if $p = k$ **then return** a_p
  else if $p < k$ **then**
    $kth\_order\_statistics([a_1, ..., a_{p-1}], k)$
  else
    $kth\_order\_statistics([a_{p+1}, ..., a_n], k - p)$
  end if

end function

Najpierw sprawdzany jest warunek, który rozstrzyga czy szukana statystyka pozycyjna jest mniejsza lub równa rozmiarowi tablicy, a jeśli tak nie jest, zwracamy błąd. Następnie za pomocą funkcji $median()$ liczona jest mediana podanej tablicy. Koszt jej wykonania to $O(n)$. Zostaje ona dalej podana do funkcji $partition()$. Jej pseudokod został podany poniżej, jednak można powiedzieć, że jest to w zasadzie standardowy algorytm $partition()$ znany nam z Quicksorta, który jako pivot obiera podaną mu wartość:

function PARTITION($[a_1, a_2, ..., a_n], p$)
    $i \leftarrow 1$

    for $i; i < n; i \leftarrow i + 1$ **do**
        if $a_i = p$ **then**
            break
        end if
    end for

    $swap(a_i, a_n)$
    $i \leftarrow 1$

    for $j \leftarrow 1; j < n; j \leftarrow j + 1$ **do**
        if $a_j \leqslant p$ **then**
            $swap(a_i, a_j)$
            $i \leftarrow i + 1$
        end if
    end for

    $swap(a_i, a_n)$
    return i
end function

Najpierw w podanej tablicy szukany jest element o wartości p, aby następnie uznać go za pivot. Dalsza część funkcji jest już standardowym algorytmem $partition()$ znanym z Quicksorta. Łatwo zauważyć, iż przechodzi ona wejściową tablicę maksymalnie 2 razy: raz aby odnaleźć pivot i raz aby przestawić elementy na odpowiednie miejsca. Zatem jej złożoność wynosi w pesymistycznym przypadku $O(n)$.

Widzimy, że $kth\_order\_statistics()$ w trakcie pojedynczego wywołania wykonuje dokładnie jeden raz każdą z funkcji: $median()$ oraz $partition()$. Stąd koszt jej wywołania to $O(n) + O(n) = O(n)$. Na koniec funkcja podejmuje decyzję: jeśli obliczony przez $partition()$ punkt podziału odpowiada szukanej statystyce, zwraca ona jej wartość i kończy działanie; jeśli punkt ten jest mniejszy od k, znaczy to, że statystyki szukać należy w lewej części podziału, co jest następnie realizowane poprzez uruchomienie rekursji. Odpowiednio, jeśli punkt okaże się większy niż k, kontynuujemy poszukiwania w części prawej. Jako, że punktem podziału jest mediana zbioru, funkcja $partition()$ zapewni rozdzielenie zbioru na połowę (zakładając oczywiście, że w tablicy wejściowej brak jest powtórzeń elementów).

Koszt tego algorytmu da się więc opisać następującym równaniem:

$$T(n) = T(\frac{n}{2}) + O(n)$$

Korzystając z Master Theorem:

$$log_2(1) = 0 < 1 \Rightarrow T(n) = O(n)$$

Co pokazuje, że udało nam się osiągnąć warunek podany w treści zadania.