Lista 5

Victor Sena Molero - 8941317

March 25, 2016

Ex 8. Descreva um algoritmo que, dados n inteiros no intervalo de 1 a k, preprocesse sua entrada e então responda em $O(1)$ qualquer consulta sobre quantos dos n inteiros dados caem em um intervalo $[a..b]$. O preprocessammento efetuado pelo seu algoritmo deve consumir tempo $O(n+k)$.

Resposta. Para resolver o problema em tempo linear podemos, primeiro, inicializar um vetor c de contagem de tamanho $k+1$ (de 0 a k, inclusive) com todos os valores iguais a 0 em tempo $O(k)$. Depois, precisamos percorrer o vetor de entrada v e, para cada valor v_i, somar 1 a c_{v_i}, isso é feito em $O(n)$.
Agora, basta acumular o valor do vetor c nele mesmo, ou seja, percorrer o vetor c de 1 a k efetuando $c_i = c_{i-1} + c_i$, que também custa $O(k)$. Assim, nosso algortimo preprocessa o vetor de maneira conveniente em $O(k) + O(n) + O(k) = O(n+k)$.
Para responder a uma query (a, b) basta imprimir o valor de $c_b - c_{a-1}$. O fato do valor $a-1$ ser consultado justifica a posição 0 no vetor c. Além disso, se não houver garantia de que $1 \le a, b \le k$ basta executar, antes de calcular a resposta, $a = min(max(a, 1), k)$ e $b = min(max(b, 1), k)$. □

Algoritmo.

function PRE_PROCESSA
  $i \leftarrow 0$
  while $i \le k$ **do**
    $c[i] \leftarrow 0$
    $i \leftarrow i + 1$
  end while
  $i \leftarrow 1$
  while $i \le n$ **do**
    $c[v[i]] \leftarrow c[v[i]] + 1$
    $i \leftarrow i + 1$
  end while
  $i \leftarrow 1$
  while $i \le k$ **do**
    $c[i] \leftarrow c[i-1] + c[i]$
    $i \leftarrow i + 1$
  end while
end function

function $\textsc{Consulta}(a, b)$
  $a = max(min(a, k), 1)$
  $b = max(min(b, k), 1)$
  return $c[b] - c[a-1]$
end function

$\square$

Ex 13. Mostre como multiplicar dois números complexos $a+bi$ e $c+di$ usando apenas três multiplicações reais. O seu algoritmo deve receber como entrada os números a, b, c e d e devolver os números $ac-bc$ (componente real do produto) e $ad+bc$ (componente imaginária do produto).

Algoritmo.
  $r_1 \leftarrow (a+b)*(c-d)$
  $r_2 \leftarrow b*c$
  $r_3 \leftarrow a*d$
  return $r_1 - r_2 + r_3, r_2 + r_3$

$\square$

Ex 14. No $\textsc{Select}$-BFPRT, os elementos do vetor são divididos em grupos de 5. O algoritmo continua linear se dividirmos os elementos em grupos de 7? E em grupos de 3? Justifique sua resposta.

Reposta. As respostas são, respectivamente, sim e não.
Na análise do algoritmo $\textsc{Select}$-BFPRT padrão, chegamos em duas fórmulas importantes, $\lceil n/5 \rceil$, que representa a quantidade de intervalos de tamanho 5 ou menos nos quais o vetor original é dividido, e $\lceil 7n/10 \rceil + 3$.
A primeira é fácilmente adpatável se trocarmos a quantidade de elementos em cada intervalo, por exemplo, suponha que queiramos intervalos de k elementos, podemos seguramente substituir aquela fórmula por $\lceil n/k \rceil$.
Para generalizar a segunda, vamos pensar na quantidade mínima de elementos maiores que o pivô escolhido. Cada grupo que tem mediana maior que o pivô (pelo menos $\lfloor 1/2 \lceil n/k \rceil \rfloor$ grupos) contribui com $\lceil k/2 \rceil$ elementos, com exceção de um possível grupo com um só elemento que pode ser maior que o pivô, pelo qual são descontados $\lfloor k/2 \rfloor$ elementos da conta final. Além disso, temos mais $\lfloor k/2 \rfloor$ elementos maiores que o pivô no mesmo intervalo que ele. Assim, a segunda fórmula pode ser generalizada para:

$$n - 1 - (\lfloor 1/2 \lceil n/k \rceil \rfloor)(\lceil k/2 \rceil) \leq$$

$$\leq n - 1 - (1/2)(n/k + 1)((k+1)/2) =$$

$$= n - 1 - ((k+1)/2)(n/2k + 1/2)$$

Para resolver a recorrência para $k = 5$ e provar que $T(n) = O(n)$ chega um momento em que devemos mostrar que, com alguma constante α e outra β e um n suficientemente grande, temos

$$\alpha(\lceil n/5 \rceil) + \alpha(\lceil 7n/10 \rceil + 3) + \beta n < \alpha n$$

Generalizando isso, deveríamos mostrar que

$$\alpha(\lceil n/k \rceil) + \alpha(n - 1 - ((k+1)/2)(n/2k + 1/2)) + \beta n < \alpha n$$

Agora vamos separar os casos onde $k = 3$ e $k = 7$.
Se $k = 3$ temos que mostrar

$$\alpha(\lceil n/3 \rceil) + \alpha(n - 1 - 2(n/6 + 1/2)) + \beta n \leq$$

$$\leq \alpha(n/3 + 1 - 1 + 2n/3 + 1) + \beta n = \alpha(n + 1) + \beta n < \alpha n$$

O que é impossível, ou seja, pelos métodos apresentados pelo livro e com os bounds que conseguimos achar, não conseguimos mostrar que T é linear, isso não é uma prova de que T é não linear, mas eu não consegui provar isso, apenas consegui mostrar que, pelos métodos usados no livro, não conseguimos provar.
E, por outro lado, se $k = 7$ temos que mostrar

$$\alpha(\lceil n/7 \rceil) + \alpha(n - 1 - 4(n/14 + 1/2)) + \beta n \leq$$

$$\leq \alpha(n/7 + 1 - 1 + 10n/14 + 2) + \beta n =$$

$$= \alpha(12n/14) + \alpha(2) + \beta n < \alpha(n)$$

E para isso, basta escolher $\alpha = 14$ e $\beta = 1$ para obter

$$13n + 28 < 14n$$

O que é verdade quando $n > 28$, logo, T é linear. $\square$