MAC 6711 - Tópicos de Análise de Algoritmos

Departamento de Ciência da Computação

Primeiro semestre de 2012

Lista 1

1. Considere o seguinte algoritmo recursivo para calcular o máximo de um vetor $v[p\,..\,r]$.

   Algoritmo Máximo (v, p, r)

   1. **se** $p = r$
   2. **então devolva** $v[p]$
   3. **senão** $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
   4. $m1 \leftarrow$Máximo (v, p, q)
   5. $m2 \leftarrow$Máximo $(v, q+1, r)$
   6. **se** $m1 > m2$
   7. **então devolva** $m1$
   8. **senão devolva** $m2$

   Seja $C(n)$ o número de comparações executadas na linha 6 do algoritmo por uma chamada de Máximo(v, p, r), onde $n = r - p + 1$. Deduza do algoritmo uma recorrência que defina $C(n)$ e resolva-a, exibindo uma fórmula fechada para $C(n)$. (Ou seja, dê a solução exata da recorrência.)

2. Escreva um algoritmo que ordena uma lista de n itens dividindo-a em três sublistas de aproximadamente $n/3$ itens, ordenando cada sublista recursivamente e intercalando as três sublistas ordenadas. Analise seu algoritmo concluindo qual é o seu consumo de tempo.

3. Sejam $X[1\,..\,n]$ e $Y[1\,..\,n]$ dois vetores, cada um contendo n números ordenados. Dê um algoritmo $\mathrm{O}(\lg n)$ para encontrar a mediana de todos os $2n$ elementos nos vetores X e Y.

4. Altere a implementação do algoritmo de Shamos e Hoye vista na aula 2 para que, no Combine, sejam calculadas apenas distâncias entre pontos de lados distintos da coleção. (*Dica:* deixe para intercalar depois da chamada do Combine.)

5. Lembre-se do problema de determinar o número de inversões de um vetor. Naquele problema, dada uma seqüência de números $a_1, \ldots, a_n$ todos distintos, definimos uma inversão como um par de índices (i, j) tal que $i < j$ e $a_i > a_j$.

   A motivação para este problema é que o número de inversões é uma boa medida de quão diferentes duas ordens são. Entretanto, pode-se achar que essa medida é muito sensível. Assim, vamos chamar de uma *inversão significativa* um par de índices (i, j) tal que $i < j$ e $a_i > 2a_j$.

   Escreva um algoritmo $\mathrm{O}(n \lg n)$ que, dado uma seqüência de números $a_1, \ldots, a_n$ todos distintos, determina o número de inversões significativas da seqüência. Como sempre, mostre que seu algoritmo funciona e deduza o seu consumo de tempo.

6. Modifique o algoritmo visto em aula para calcular a inversa da transformada discreta de Fourier em tempo $\Theta(n \lg n)$.

7. A *silhueta de um prédio* é uma tripla (l, h, r) de números positivos com $l < r$, onde h representa a altura do prédio, l representa a posição inicial da sua base e r a posição final.

   Considere uma coleção $\mathcal{S} = \{(l_1, h_1, r_1), \ldots, (l_n, h_n, r_n)\}$ com a silhueta de n prédios. Para cada número positivo x, denote por $\mathcal{S}_x$ o conjunto $\{i : 1 \leq i \leq n \text{ e } l_i \leq x \leq r_i\}$. Denote ainda por $h(\mathcal{S}_x)$ o número $\max\{h_i : i \in \mathcal{S}_x\}$.

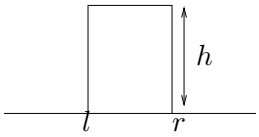

Figura 1: Silhueta (l, h, r) de um prédio.

O *skyline* de $\mathcal{S}$ é uma seqüência $(x_0, t_1, x_1, \ldots, t_k, x_k)$ tal que

1. $x_0 = 0$ e $x_k = \max\{r_i : 1 \leq i \leq n\}$;
2. $x_{j-1} < x_j$ para $j = 1, \ldots, n$;
3. para $j = 1, \ldots, n$, $t_j = h(\mathcal{S}_x)$ para todo x tal que $x_{j-1} < x < x_j$;
4. $t_j \neq t_{j+1}$ para $j = 1, \ldots, n-1$.

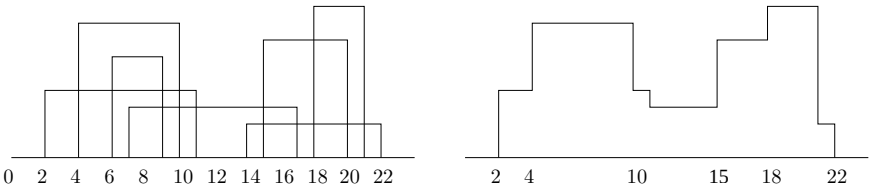

Figura 2: Coleção de silhuetas $\mathcal{S} = \{(15, 7, 20), (4, 8, 10), (18, 9, 21), (2, 4, 11), (7, 3, 17), (6, 6, 9), (14, 2, 22)\}$ e seu skyline $(0, 0, 2, 4, 4, 8, 10, 4, 11, 3, 15, 7, 18, 9, 21, 2, 22)$.

(a) Escreva um algoritmo que recebe o skyline de uma coleção $\mathcal{S}_1$ de silhuetas de prédios e o skyline de uma coleção $\mathcal{S}_2$ de silhuetas de prédios e devolve o skyline de $\mathcal{S}_1 \cup \mathcal{S}_2$. Seu algoritmo deve consumir tempo $\mathrm{O}(n)$, onde $n = |\mathcal{S}_1 \cup \mathcal{S}_2|$. Explique por que seu algoritmo faz o que promete e por que consome o tempo pedido.

(b) Escreva um algoritmo que recebe um inteiro n e uma coleção $\mathcal{S}$ de n silhuetas de prédios e devolve o skyline de $\mathcal{S}$. Seu algoritmo deve consumir tempo $\mathrm{O}(n \lg n)$. Explique por que seu algoritmo faz o que promete e por que consome o tempo pedido.

8. A remoção da superfície escondida é um problema em computação gráfica que raramente precisa de introdução: quando o João tá na frente da Maria, você pode ver o João, mas não a Maria; quando a Maria tá na frente do João, ... Você entendeu a idéia.

   A beleza desse problema é que você pode resolvê-lo mais rapidamente do que a intuição em geral sugere. Aqui está uma versão simplificada do problema onde já podemos apresentar um algoritmo mais eficiente do que a primeira solução em que se pode pensar. Imagine que são dadas n retas não verticais no plano, denotadas por $L_1, \ldots, L_n$. Digamos que L_i é dada pela equação $y = a_i x + b_i$, para $i = 1, \ldots, n$. Suponha que não há três retas entre as retas dadas que se interseptam mutuamente num mesmo ponto. Dizemos que a reta L_i é a *mais alta* numa dada coordenada $x = x_0$ se sua coordenada y em x_0 é maior que a coordenada y em x_0 de todas as outras retas dadas. Ou seja, se $a_i x_0 + b_i > a_j x_0 + b_j$ para todo $j \neq i$. Dizemos que L_i é *visível* se existe uma coordenada x na qual ela é a mais alta. Intuitivamente, isso corresponde a uma parte de L_i ser visível se você olhar para baixo a partir de $y = \infty$.

   Escreva um algoritmo $\mathrm{O}(n \lg n)$ que recebe uma seqüência de n retas, como descrito acima, e devolve a subseqüência delas que é visível.