

MAC 6711 - Análise de Algoritmos

Departamento de Ciência da Computação

Primeiro semestre de 2012

Lista 2

1. Problemas 3-1 a 3-5 e Problema 4-1 do CLRS.
2. A recorrência $T(n) = 7T(n/2) + n^2$ descreve o consumo de tempo de um algoritmo A . Um algoritmo A' , que compete com A , tem o seu consumo de tempo dado por uma recorrência da forma $T'(n) = aT'(n/4) + n^2$. Qual é o maior valor de a para o qual A' é assintoticamente melhor do que A ?
3. Se $T(0) = T(1) = 1$, cada uma das seguintes recorrências define uma função T nos inteiros não-negativos.
 - (a) $T(n) = 3T(\lfloor n/2 \rfloor) + n^2$;
 - (b) $T(n) = 2T(n-2) + 1$;
 - (c) $T(n) = T(n-1) + n^2$.

Qual delas não pode ser limitada por uma função polinomial? Justifique a sua resposta.

4. Diz-se que um vetor $A[1..n]$ tem um elemento majoritário se mais da metade de seus elementos são iguais. Dado um vetor, a sua tarefa é projetar um algoritmo eficiente que determine se tal vetor tem ou não um elemento majoritário. Os elementos do vetor não são necessariamente de um domínio ordenado como os inteiros, assim comparações do tipo “ $A[i] > A[j]$?” podem não fazer sentido. (Pense nas entradas do vetor como arquivos GIF por exemplo.) No entanto, você pode fazer comparações do tipo “ $A[i] = A[j]$?”. Projete um algoritmo linear para resolver este problema. Se não conseguir, tente projetar um algoritmo $O(n \lg n)$.
5. Descreva um algoritmo que, dados inteiros n e k , juntamente com k listas ordenadas que em conjunto tenham n registros, produza uma única lista ordenada contendo todos os registros dessas listas (isto é, faça uma *intercalação*). O seu algoritmo deve ter complexidade $O(n \lg k)$. Note que isto se transforma em $O(n \lg n)$ no caso de n listas de 1 elemento, e em $O(n)$ se só houver uma lista (de n elementos).
6. Considere a seqüência de vetores

$$A_k[1..2^k], A_{k-1}[1..2^{k-1}], \dots, A_1[1..2^1], A_0[1..2^0].$$

Suponha que cada um dos vetores é crescente. Queremos reunir, por meio de sucessivas operações de intercalação (= *merge*), o conteúdo dos vetores A_0, \dots, A_k em um único vetor crescente $B[1..n]$, onde $n = 2^{k+1} - 1$. Escreva um algoritmo que faça isso em $O(n)$ unidades de tempo. Você não precisa escrever o código da rotina INTERCALA, mas precisa dizer o que ela faz exatamente. Justifique.

7. Exercícios 28.2-4 e 28.2-5 do CLRS.
8. Exercícios 30.1-3 e 30.1-7, e problema 30-2 do CLRS.
9. Dado um algoritmo linear “caixa-preta” para encontrar uma mediana de um vetor, dê um algoritmo simples, linear, que resolve o problema do k -ésimo mínimo.
10. No 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.

11. Considere a seguinte variante do PARTICIONE-BFPRT, que chamaremos de PARTICIONE-D. Em vez de acionar o SELECT-BFPRT para calcular a mediana das medianas, ela aciona recursivamente o próprio PARTICIONE-D, para calcular uma “mediana aproximada” do vetor das medianas. Suponha que o PARTICIONE-D rearranja o vetor $A[p..d]$ e devolve um índice q tal que $A[p..q-1] \leq A[q] < A[q+1..d]$ e $\max\{k, n-k\} \leq 9n/10$, onde $n = d-p+1$ e $k = q-p+1$. Analise o consumo de tempo da variante do SELECT-BFPRT que chama o PARTICIONE-D em vez do PARTICIONE-BFPRT.
12. Tente provar por indução a suposição feita sobre o PARTICIONE-D no exercício acima. (Você pode assumir que para vetores pequenos, por exemplo, com até 5 elementos, o PARTICIONE-D devolve uma mediana do vetor.) Apresente a prova da suposição ou explique porque você não consegue prová-la. O que acontece com a sua prova/argumento caso você substitua a fração $9/10$ por uma outra fração α mais próxima de 1? Caso você não consiga provar a suposição, você é capaz de descrever um contra-exemplo para ela?
13. (Exercício 4 do KT) Você está trabalhando com alguns físicos que precisam estudar, como parte de seu projeto experimental, as interações entre um grande número de partículas carregadas eletricamente. Basicamente, o ambiente deles funciona da seguinte maneira. Eles têm uma estrutura de grade neutra, e eles usam esta estrutura para colocar as partículas carregadas numa linha reta, regularmente espaçadas. Assim, podemos modelar esta estrutura como os pontos $\{1, 2, \dots, n\}$ na reta real. Para cada ponto j em $\{1, 2, \dots, n\}$, eles têm uma partícula com carga q_j , que pode ser positiva ou negativa.

Eles querem estudar a força total em cada partícula, medindo-a e então comparando o resultado medido com uma previsão calculada computacionalmente. É nesta parte computacional que eles precisam da sua ajuda. A força total da rede sobre a partícula j , pela Lei de Coulomb, é igual a

$$F_j = \sum_{i < j} \frac{C q_i q_j}{(j-i)^2} - \sum_{i > j} \frac{C q_i q_j}{(j-i)^2},$$

onde C é uma constante. Eles escreveram o seguinte programa para calcular F_j para todo j :

Algoritmo CalculaF(q, n, j)

1. **para** $j \leftarrow 1$ **até** n **faça**
2. $F[j] \leftarrow 0$
3. **para** $i \leftarrow 1$ **até** n **faça**
4. **se** $i < j$
5. **então** $F[j] \leftarrow F[j] + C q[i]q[j]/(j-i)^2$
6. **se** $i > j$
7. **então** $F[j] \leftarrow F[j] - C q[i]q[j]/(j-i)^2$
8. **devolva** F

Não é difícil analisar o consumo de tempo do algoritmo acima: cada invocação do **para** interno leva tempo $\Theta(n)$ e ele é invocado n vezes. Assim o tempo consumido por este algoritmo é $\Theta(n^2)$.

O problema é que, para valores grandes de n , como os que os seus colegas físicos estão trabalhando, o programa leva vários minutos para executar. Por outro lado, o ambiente experimental deles está otimizado de modo que eles colocam n partículas, fazem a medição e estão prontos para manipular outras n partículas em poucos segundos. Assim eles realmente gostariam de ter um programa que calculasse as forças F_j muito mais rapidamente, numa velocidade comparável a de seus experimentos.

Ajude os seus colegas físicos projetando um algoritmo que compute o vetor de forças $F[1..n]$ em tempo $O(n \lg n)$.