

PUC-Rio
Departamento de Informática
Prof. Marcus Vinicius S. Poggi de Aragão
Horário: 2as-feiras de 13 às 16 horas - Sala 524L
15 de setembro de 2008
Período: 2008.2

PROJETO E ANÁLISE DE ALGORITMOS (INF 2926)

1ª Lista de Exercícios

1. Analise a eficiência do algoritmo abaixo, calculando o número de passos executados pelo mesmo em função de um número n fornecido como entrada. Considere que todas as operações básicas envolvidas (atribuição, operações lógicas, operações aritméticas, entrada/saída) possuem custo constante ($c = O(1)$).

```
Leia(n);  
x <- 0;  
Para i <- 1 até n faça  
  Para j <- i+1 até n faça  
    Para k <- 1 até j-i faça  
      x  x + 1;  
Imprima(x);
```

2. Perdido em uma terra muito distante, você se encontra em frente a um muro de comprimento infinito para os dois lados (esquerda e direita). Em meio a uma escuridão total, você carrega um lampião que lhe possibilita ver apenas a porção do muro que se encontra exatamente à sua frente (o campo de visão que o lampião lhe proporciona equivale exatamente ao tamanho de um passo seu). Existe uma porta no muro que você deseja atravessar. Supondo que a mesma esteja a n passos de sua posição inicial (não se sabe se à direita ou à esquerda), elabore um algoritmo para caminhar ao longo do muro que encontre a porta em $O(n)$ passos. Considere que n é um valor desconhecido (informação pertencente à instância). Considere que a ação composta por dar um passo e verificar a posição do muro correspondente custa $O(1)$.
3. “Pseudo ou não-pseudo? Eis a questão.”
 - (a) Defina os conceitos de algoritmo polinomial e algoritmo pseudopolinomial.
 - (b) Forneça um exemplo de um algoritmo polinomial e outro de um algoritmo pseudopolinomial.
 - (c) Prove ou refute: Todo algoritmo polinomial é pseudopolinomial.
 - (d) Prove ou refute: Todo algoritmo pseudopolinomial é polinomial.
4. Era uma vez um rei que não conhecia algoritmos. Durante um fatídico verão, seu reino fora invadido por um dragão, obrigando os conselheiros da ordem omega-theta a se reunirem e decidirem por contratar o cavaleiro mais corajoso (e ambicioso) do reino que fez uma proposta de recompensa associada ao peso do animal. Considerando que o peso do dragão seja n quilos, a cobrança será feita em moedas de ouro, seguindo a regra de $k \cdot 2^{(n-k)}$ moedas de ouro

adicionais para o k -ésimo quilo do dragão. Para exemplificar, um dragão com 4 quilos custaria ao reino $1 \cdot 23 + 2 \cdot 22 + 3 \cdot 21 + 4 \cdot 20 = 26$ moedas de ouro.

Elabore um algoritmo POLINOMIAL (LINEAR!!!) que, dado um inteiro positivo n , correspondente ao peso do dragão, calcule a quantidade de moedas a serem pagas ao cavaleiro. O algoritmo (projetado para ser executado pelos conselheiros da corte) deve conter apenas as operações aritméticas de soma, subtração, multiplicação e divisão de inteiros (considere que qualquer uma dessas operações é realizada em tempo constante).

5. No que se refere ao conceito de esquema de codificação de uma instância:

- Prove por indução que um número inteiro positivo n ao ser codificado na base $b \geq 2$ faz uso de $(\log_b n)$ caracteres.
- Prove por indução que um número inteiro positivo n ao ser codificado na base unária faz uso de (n) caracteres.
- Utilizando o resultados anteriores prove que, um algoritmo que recebe como entrada um número inteiro positivo n codificado em uma base $b \geq 2$ e que executa em n passos é um algoritmo de complexidade exponencial em função do tamanho da instância.

6. Considere o seguinte problema:

[ELE] Dados um conjunto de n inteiros distintos $X = \{x_1, \dots, x_n\}$ e pesos positivos $w(x_j)$, $j = 1, \dots, n$, associados aos elementos de X , e um inteiro positivo V tal que $0 \leq V \leq W = \sum_{j=1}^n w(x_j)$; encontrar o elemento x_k tal que $\sum_{x_j < x_k} w(x_j) < V$ e $w(x_k) + \sum_{x_j < x_k} w(x_j) \geq V$.

- Projete um algoritmo $O(n \log n)$ para resolver esse problema.
- Utilize divisão e conquista para projetar um algoritmo $O(n)$ para resolver esse problema.
- Apresente detalhadamente a análise da complexidade do item anterior.

7. Sejam u e v dois números de n bits (considere que n é potência de 2). A multiplicação tradicional de u por v utiliza $O(n^2)$ operações. Um algoritmo baseado em divisão e conquista divide os números em duas partes iguais, calculando o produto como: $uv = (a2^{n/2} + b)(c2^{n/2} + d) = ac2^n + (ad + bc)2^{n/2} + bd$. As multiplicações ac , ad , bc e bd são feitas usando este mesmo algoritmo recursivamente.

- Escreva este algoritmo
- Determine a complexidade
- Qual é a complexidade se $ad + bc$ é calculado como $(a + b)(c + d) - ac - bd$?

8. Verdadeiro ou Falso. Justifique.

- $(\log n)^{100} = O(n^\epsilon)$, $\forall \epsilon > 0$.
- $2^{n+1} = O(2^n)$.
- Se $g(n, m) = m \log_d n$ onde $d = \lceil m/n + 2 \rceil$ ($\lceil x \rceil$ é o menor inteiro maior que x), $m = O(n^2)$, então $g(n, m) = O(m^{1+\epsilon}) \forall \epsilon > 0$.

9. Determine uma forma fechada para cada uma das seguintes recorrências:

- (a) $T(1) = 1$;
 $T(2) = 6$;
 $T(n) = T(n - 2) + 3n + 4, \forall n \geq 3$.
- (b) $T(1) = 1$;
 $T(2) = 6$;
 $T(n) = 2.T(n - 2) + 3, \forall n \geq 3$.
- (c) $T(1) = 1$;
 $T(n) = \sum_{i=1}^{n-1} (T(i) + T(n - i)) + 1, \forall n \geq 2$.

10. Seja $S = \{x_1, x_2, \dots, x_n\}$ uma seqüência de números reais (não necessariamente positivos). Projete dois algoritmos de complexidade $O(n)$, um iterativo e um que utilize divisão e conquista (ambos podem ser recursivos!), que determine uma subseqüência $S' = \{x_i, x_{i+1}, \dots, x_j\}$ de elementos consecutivos da seqüência S tal que o produto dos números que compõem S' é máximo dentre todas as subseqüências consecutivas possíveis.
11. Dados um multiconjunto C contendo n números reais (não necessariamente distintos) e um número real x :
- (a) Projete um algoritmo de complexidade $O(n \log n)$ que determine se existem dois elementos em C cuja soma seja igual a x .
- (b) Considere agora que os elementos do conjunto C estão ordenados crescentemente. Projete um algoritmo de complexidade $O(n)$ para resolver o mesmo problema.
12. O departamento de transporte de uma cidade deseja averiguar se seus motoristas respeitam os conceitos pregados pela técnica de direção defensiva. De acordo com o departamento, todos os veículos deveriam manter entre si uma distância maior ou igual d , definida como *distância de segurança*. Com o objetivo de realizar uma pesquisa, várias câmeras foram instaladas pela cidade, captado as posições de inúmeros veículos. A posição de um veículo é definida por um par ordenado $(x, y) \in R^2$. Seu objetivo é projetar um algoritmo de complexidade $O(n \log n)$ que obtenha uma cena contendo as posições de n veículos captadas por uma câmera e a distância de segurança d atualmente estabelecida pelo departamento, informando ao final da execução se existem veículos que não respeitam a distância de segurança. Seu algoritmo deve responder apenas *sim* ou *não*. A distância entre dois veículos com posições (x_1, y_1) e (x_2, y_2) é dada por $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.
13. O problema da torre de Hanói generalizado consiste em mover n discos de diâmetros distintos, posicionados em um haste denominada *origem*, para uma haste denominada *destino* utilizando $m \geq 1$ hastes denominadas *de trabalho*. Inicialmente, todos os discos encontram-se empilhados na haste de origem em ordem decrescente de tamanho, de baixo para cima. As demais hastes de trabalho e destino encontram-se vazias. Durante o processo de transferência é permitida a movimentação de apenas um disco por vez. Considere ainda que nenhum disco pode ser posicionado acima de um disco com diâmetro menor que o seu. Projete um algoritmo que resolva o problema da torre de Hanói generalizado utilizando o menor número de movimentos possível. Seu algoritmo deve relatar a ordem e a quantidade de movimentos a serem realizados.

14. Considere a multiplicação de n matrizes A_1, \dots, A_n . Considere também que denota-se o produto das matrizes $A_k.A_{k+1} \dots A_q$ por $A_{k..q}$ e que as dimensões das matrizes são dadas por $d_k \times d_{k+1}$ para a matriz A_k . Assim, $A_{1..n}$ terá dimensão $d_1 \times d_{n+1}$.

Aqui a multiplicação de pares consecutivos de matrizes $A_k.A_{k+1}$ é feita calculando

$$a_{i,j}^{k..k+1} = \sum_{p=1}^{d_{k+1}} a_{i,p}^k \cdot a_{p,j}^{k+1}$$

para todo par (i, j) que é elemento de $A_k.A_{k+1}$ e onde $a_{i,j}^k$ representa o elemento (i, j) da matriz A_k .

Observe que a multiplicação de 3 matrizes A_1, A_2 e A_3 , pode ser feita de duas maneiras: $((A_1.A_2).A_3)$ e $(A_1.(A_2.A_3))$. (De quantas maneiras pode-se obter o produto de n matrizes ?)

Observe também que para cada maneira de se multiplicar n matrizes pode-se ter que realizar um número diferente de multiplicações. Quantas são ?

Apresente um algoritmo para determinar a maneira de se multiplicar as n matrizes que utiliza o menor número de multiplicações. Analise a complexidade do algoritmo proposto. A complexidade é polinomial ? Qual a complexidade menor possível que um algoritmo que resolve este problema pode ter ?

15. Um comerciante possui um armazém que utiliza para suprir seus clientes de um único produto. O seu armazém pode guardar até C unidades do produto. Para as próximas T semanas o comerciante TEM que atender às demandas dos seus clientes que somam d_t para a semana t , onde $t = 1, 2, \dots, T$. Além disso, ele possui $s_0 (\leq C)$ unidades em estoque antes do início da primeira semana, e já negociou com os fornecedores os preços unitários p_t ($t = 1, 2, \dots, T$). Ele deseja planejar o atendimento dos seus clientes de modo a gastar o mínimo possível com a compra do produto.

Ajude ao comerciante a definir a sua estratégia ótima de compra do produto nas semanas $t = 1, \dots, T$.

- Apresente o algoritmo que obtém a estratégia de compra de menor custo e atende às demandas dos seus clientes.
- Analise a complexidade do algoritmo proposto. A complexidade é polinomial ? Qual a complexidade menor possível que um algoritmo que resolve este problema pode ter ?
- Execute o seu algoritmo sobre a seguinte instância: $C = 12, T = 5, s_0 = 3, d_1 = 7, d_2 = 4, d_3 = 15, d_4 = 10, d_5 = 7$ e $p_1 = 3, p_2 = 4, p_3 = 7, p_4 = 6, p_5 = 8$. Informe quanto o comerciante deve comprar em cada semana e o seu custo total.

16. Considere um tabuleiro de xadrez e um rei que está inicialmente na posição $(1, 1)$ (as posições do tabuleiro são representadas por (i, j) onde $1 \leq i \leq 8$ e $1 \leq j \leq 8$). Para cada posição do tabuleiro estão associados um prêmio p_{ij} e um consumo q_{ij} (o prêmio pode ser em USD(!) e o consumo em litros de gasolina, por exemplo). Os prêmios e os consumos assumem somente valores positivos. O rei tem inicialmente Q unidades para consumir e pode passar quantas vezes quiser em cada posição do tabuleiro e a cada vez receber o prêmio e, naturalmente, consumir os seus recursos. Ao final (do passeio) **o rei tem que estar de volta na posição $(1, 1)$.**

- (a) Proponha um algoritmo para determinar o caminho que o rei deve fazer para obter o maior total possível em prêmios.
(Dica: Suponha que você conhece a solução que obtém o maior total em prêmios dado que o rei está em cada uma das posições do tabuleiro e para cada consumo possível. Escreva agora o teorema do passo indutivo reforçando a hipótese indutiva. A prova por indução matemática (simples) de que você sabe resolver o problema do rei leva ao algoritmo).
- (b) Analise a complexidade do algoritmo proposto. A complexidade é polinomial? Qual a complexidade menor possível que um algoritmo que resolve este problema pode ter? Qual a complexidade deste problema?
- (c) Suponha que $Q = 7$ e que $q_{ij} = 1$ e $p_{ij} = 2 * i + 3 * j$ para todo (i, j) . Determine o caminho em que o rei acumula o maior total possível de prêmios. Repita o cálculo modificando apenas $p_{22} = 100$ e mantendo os demais valores.

17. Sejam $T = \{t_1, \dots, t_n\}$ e $P = \{p_1, \dots, p_k\}$ duas sequências de caracteres on $k \leq n$.

- (a) Proponha um algoritmo linear para determinar se P é uma subsequência de T , isto é, se os elementos de P aparecem em T na mesma ordem que em P , mas não necessariamente consecutivos.
- (b) Suponha que a resposta do item anterior é negativa. Proponha um algoritmo para encontrar a maior subsequência de P que está em T .
- (c) Considere que para cada elemento de T é associado um custo positivo c_i , $i = 1, \dots, n$. Proponha um algoritmo para encontrar a subsequência de P que é subsequência de T onde a soma dos custos dos elementos de T é maximizada.
- (d) Analise a complexidade dos algoritmos propostos. Qual a complexidade menor possível de um algoritmo que resolve estes problemas?