PPGMCS - Projeto e Análise de Algoritmos
Lista de Exercícios
Análise de Complexidade de Algoritmos

Prof. Renê R. Veloso

1 de Julho de 2015

1 Resolva os exercícios a seguir:

Questão 1: Considerando um grafo $G = (V, E)$, qual é a complexidade de tempo de uma busca em profundidade?

Questão 2: Pesquise sobre: Uma estrutura Union-Find é ideal para representação de conjuntos disjuntos. Utilizando o mecanismo de compressão de caminho, o custo para cada operação *find* é muito próxima a constante. Para ser mais preciso, isso quer dizer que o custo para n operações *find* em n vértices é aproximadamente $O(n \log^* n)$, que cresce de forma muito lenta, proporcional ao inverso da função de Ackermann. Responda: O que tudo isso quer dizer?

Questão 3: Compare o desempenho dos algoritmos de ordenação Merge-Sort e Quick-Sort. Dê as complexidades de tempo no pior caso de cada um e demonstre-as usando uma implementação em linguagem de programação qualquer.

Questão 4: Considerando as complexidades de tempo de cada um, responda: Qual algoritmo para encontrar componentes fortemente conectados é o melhor, Tarjan ou Kosaraju ?

Questão 5: Compare as complexidades de tempo dos algoritmos para árvore geradora mínima: Kruskal e Prim.

Questão 6: Considerando um grafo muito grande (na ordem bilhões de vertices e arestas), há algoritmo melhor do que o algoritmo de Dijkstra para o problema de caminhos mais curtos de única origem? Qual?

Questão 7: Vários algoritmos como Prim e Dijkstra usam uma fila de prioridades para extrair vértices mínimos segundo algum critério. Uma das maneiras de implementar uma fila de prioridades é através de um

Heap-Binário. Qual é a complexidade de tempo de cada função do Heap-Binário visto em sala?

Questão 8: Para as expressões a seguir, faça: $4n^2$, $\log_3 n$, 3^n, $20n$, 2, $\log_2 n$, $n^{2/3}$.

1. Plote todas em um único gráfico.
2. Ordene as expressões pela sua taxa de crescimento assintótico (da menor para a maior).

Questão 9: Seja $A[1..n]$ um vetor de números inteiros ordenado de forma crescente, apresente um algoritmo que realize qualquer sequência das seguintes operações:

- $Soma(i, y)$: soma o valor y ao i-ésimo número do vetor.
- $Soma - Parcial(i)$: retorna a soma dos primeiros i números, i.e., $\Sigma_{i=1}^{n} A[i]$.

Cada operação deve ser realizada em $O(\log n)$ passos. (Você pode usar um vetor adicional de tamanho n como auxiliar.)

Questão 10: Considere a função abaixo:

```
int X(int a)
{
  if (a<=0) then
     return 0;
  else
     return (a + X(a-1));
}
```

1. O que faz essa função?
2. Calcule a sua ordem de complexidade de tempo. Mostre os cálculos.
3. Escreva uma função não-recursiva que resolve o mesmo problema.
4. Qual implementação é mais eficiente? Justifique.

Questão 11: Considere que a multiplicação de matrizes quadradas é $O(n^3)$. Se você tivesse a opção de utilizar um algoritmo exponencial $O(2^n)$ para isso, a partir de qual valor de n valeria a pena?

Questão 12: Indique se as afirmativas a seguir são verdadeiras e justifique sua resposta:

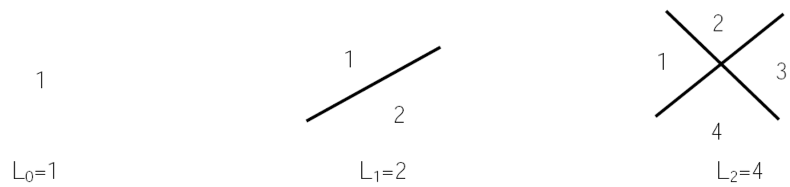
1. $2^{n+1} = O(2^n)$

2. $2^{2n} = O(2^n)$
3. $f(n) = O(u(n))$ e $g(n) = O(v(n)) \rightarrow f(n)+g(n) = O(u(n)+v(n))$

Questão 13: Qual é a complexidade de tempo do seguinte trecho de código?

```
for i = 1 to n do
   for j = 1 to i do
      for k = 1 to j do
         temp = temp + i + j+ k
```

Questão 14: Suponha que você comprou uma pizza e sabe que que pode fazer n cortes retos nela com uma faca. Quantas fatias você pode obter ao fazer esse n cortes? Isto é, qual é o número máximo de regiões L_n determinado por n retas no plano? Observe a figura abaixo e dê a equação de recorrência:



Questão 15: Prove por indução matemática que $\Sigma_{i=0}^{n} 3^i = O(3^n)$.

Questão 16: Prove por indução matemática que para $T(n) = 2T(n/2) + n$ e $T(1) = 1$, $T(n) = O(n \log n)$.

Questão 17: Qual é o limite inferior para a classe de algoritmos de ordenação por comparação?

Questão 18: Assuma que cada uma das expressões abaixo refere-se ao tempo $T(n)$ gasto por um algoritmo qualquer para resolver um problema de tamanho n. Selecione o(s) termo(s) dominante(s) considerando n e especifique a complexidade usando a notação O para cada algoritmo. (Faça o que conseguir.)

Expressão	Termo(s) dominante(s)	O(...)
$5 + 0.001n^3 + 0.025n$	?	?
$500n + 100n^{1.5} + 50n\log_{10} n$	?	?
$0.3n + 5n^{1.5} + 2.5n^{1.75}$	?	?
$n^2 \log_2 n + n(\log_2 n)^2$	?	?
$n\log_3 n + n\log_2 n$	?	?
$3\log_8 n + \log_2 \log_2 \log_2 n$	?	?
$100n + 0.01n^2$	?	?
$0.01n + 100n^2$	?	?
$2n + n^{0.5} + 0.5n^{1.25}$	?	?
$0.01n\log_2 n + n(\log_2 n)^2$	?	?
$100n\log_3 n + n^3 + 100n$	?	?
$0.003\log_4 n + \log_2 \log_2 n$	?	?

Questão 19: Considerando o problema de pesquisa por um elemento v em uma sequência A, observe que, se a sequência A estiver ordenada, poderemos comparar o ponto médio da sequência com v e eliminar metade da sequência da consideração posterior. Que algoritmo de pesquisa é esse? Escreve o seu pseudocódigo e demonstre a sua complexidade de tempo.

Questão 20: Implemente o algoritmo da Questão 19 em qualquer linguagem de programação e plote um gráfico com o tempo médio de pesquisa para diferentes tamanhos de A. Esse gráfico comprova a sua complexidade de tempo? Demonstre.

Questão 21: Descreva um algoritmo de tempo $\Theta(n \log n)$ que, dado um conjunto S de n inteiros e outro inteiro x, determine se existem ou não dois elementos em S cuja soma seja exatamente x.

Questão 22: Sejam $f(n)$ e $g(n)$ funções assintoticamente positivas. Prove ou conteste cada uma das seguintes conjecturas.

1. $f(n) = O(g(n))$ implica $g(n) = O(f(n))$
2. $f(n) + g(n) = \Theta(min(f(n), g(n)))$
3. $f(n) = O(g(n))$ implica $2^{f(n)} = O(2^{g(n)})$
4. $f(n) = O(g(n))$ implica $g(n) = \Omega(f(n))$

Questão 23: Utilize o teorema mestre para fornecer limites assintóticos restritos para as recorrências a seguir:

1. $T(n) = 4T(n/2) + n$
2. $T(n) = 4T(n/2) + n^2$
3. $T(n) = 4T(n/2) + n^3$

Questão 24: Dados n inteiros no intervalo fechado $[0, k]$. Dê um algoritmo que preprocesse a entrada em $O(n + k)$ de tal maneira que a pergunta a seguir possa ser respondida em $O(1)$. Quantos inteiros estão no intervalo $[a, b]$? Onde a e b são argumentos dados.