PUC GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
CMP1065 Projeto e Análise de Algoritmos I

PROVA 3

1. Considere o problema de ordenação. Seja dado o algoritmo de ordenação por seleção, a saber:

Algoritmo: ordenação por seleção.
% Entrada: vetor com n coordenadas, A.
% Saída: vetor ordenado em ordem não decrescente.
Para $i = 1, \ldots, (n-1)$
$min := i$.
Para $j = (i+1), \ldots, n$
Se $A(j) < A(min)$
então $min := j$.
Fim de se-então.
Fim de para.
$x := A(min)$.
$A(min) := A(i)$.
$A(i) := x$.
Fim de para.

Ainda, seja dado o algoritmo de ordenação por mergesort, a saber:

Algoritmo: ordenação por mergesort.
% Entrada: vetor com n coordenadas, A.
% Saída: uma versão ordenada desse vetor (não decrescente).
Se $n > 1$
então retorna

$merge(mergesort(A[1,\ldots,\lfloor n/2\rfloor]), mergesort(A[1,\ldots,\lceil n/2\rceil])).$
senão retorna A
Fim de se-então-senão.

No algoritmo mergesort a função *merge* é como segue: aqui o denota concatenação,

Função $merge(x[1,\ldots,k], y[1,\ldots,l])$.
Se $k = 0$, então retorna $y[1,\ldots,l]$. Fim de se-então.
Se $l = 0$, então retorna $x[1,\ldots,k]$. Fim de se-então.
Se $x[1] \leq y[1]$
então retorna $x[1]$ o $merge(x[2,\ldots,k], y[1,\ldots,l])$.
senão retorna $y[1]$ o $merge(x[1,\ldots,k], y[2,\ldots,l])$.
Fim de se-então-senão.

Pede-se: considerando o paradigma de projeto de algoritmo divisão e conquista, analisar a complexidade computacional de ambos os algoritmos e implementar os dois algoritmos para uma entrada de tamanho $n = 10$ e, em seguida, para $n = 100$. Observe o desempenho em relação ao tempo de execução.

2. Considere o problema de encontrar o $n+1$-ésimo número de Fibonacci. Seja dado o algoritmo de Fibonacci (recursivo), a saber:

Algoritmo: Fibonacci.
% Entrada: uma função F e um número natural n.
% Saída: número de Fibonacci $F(n)$.
Início de função.
Se $n < 2$
então $F(n) := n$.
senão $F(n) := F(n-1) + F(n-2)$.
Fim de se-então-senão.
Fim de função.

Ainda, seja dada a seguinte ideia de um algoritmo para o problema de encontrar o $n+1$-ésimo número de Fibonacci, a saber. O problema de se determinar $F(n)$ foi decomposto nos subproblemas de se determinar $F(n-1)$ e $F(n-2)$, respectivamente. Para se calcular $F(n-1)$, com

$n-1>1$, necessita-se dos resultados dos subproblemas $F(n-2)$ e $F(n-3)$. A ideia então é calcular cada um destes, exatamente uma vez, armazenando o resultado em uma tabela. Esta seria consultada cada vez que o mesmo subproblema fosse reconsiderado. Nesse caso, a tabela pode ser constituída simplesmente de um vetor de tamanho $n+1$, sendo $F(n)$ o elemento da sequência que se deseja calcular.

Pede-se: considerando o paradigma de projeto de algoritmo programação dinâmica, analisar a complexidade computacional de ambos os algoritmos e implementar os dois algoritmos para uma entrada de tamanho $n=10$ e, em seguida, para $n=100$. Observe o desempenho em relação ao tempo de execução.

3. Considere a seguinte tabela de distâncias aéreas, em milhas, entre seis das maiores cidades do mundo, respectivamente: Londres (L), Cidade do México (CM), Nova Iorque (NY), Paris (Pa), Pequim (Pe) e Tóquio (T).

$$\begin{bmatrix} * & L & CM & NY & Pa & Pe & T \\ L & 0 & 5558 & 3469 & 214 & 5074 & 5959 \\ CM & 5558 & 0 & 2090 & 5725 & 7753 & 7035 \\ NY & 3469 & 2090 & 0 & 3636 & 6844 & 6757 \\ Pa & 214 & 5725 & 3636 & 0 & 5120 & 6053 \\ Pe & 5074 & 7753 & 6844 & 5120 & 0 & 1307 \\ T & 5959 & 7035 & 6757 & 6053 & 1307 & 0 \end{bmatrix}.$$

Ainda, considere o algoritmo,

Algoritmo: Kruskal.
% Entrada: um grafo $G=(V,E)$ conexo com pesos.
% Saída: uma árvore geradora mínima.
Iniciar com um subgrafo gerador $T=(V,\emptyset)$.
Ordenar as arestas de G com pesos não decrescentes, $e_1,\ldots,e_{|E|}$.
Para $j=1,\ldots,|E|$
  Se $e_1,\ldots,e_{i_j}$ já foram escolhidos
  então adicione a T, e_{i_j+1} tal que
    o subgrafo aresta-induzido $G[\{e_1,\ldots,e_{i_j+1}\}]$ é acíclico.
  Fim de se-então.
Fim de para.

Pede-se: considerando o paradigma de projeto de algoritmo guloso, encontrar uma árvore geradora mínima para o problema acima usando o algoritmo de Kruskal e analisar a sua complexidade computacional.

4. (Questão discursiva 5, Ciência da Computação - ENADE 2014) As técnicas de projeto de algoritmos são essenciais para que os desenvolvedores possam implementar software de qualidade. Essas técnicas descrevem os princípios que devem ser adotados para se projetar soluções algorítmicas para um dado problema. Entre as principais técnicas destacam-se os projetos de algoritmos por tentativa e erro, divisão e conquista, programação dinâmica e algoritmos gulosos. Nesse contexto, faça o que se pede nos itens a seguir.

   (a) Descreva o que caracteriza o projeto de algoritmos por divisão e conquista.

   (b) Apresente uma situação de uso da técnica de projeto de algoritmos por divisão e conquista.

5. Considere o projeto de algoritmo programação dinâmica. Pede-se:

   (a) descreva o que caracteriza o projeto de algoritmos programação dinâmica.

   (b) Apresente uma situação de uso da técnica de projeto de algoritmos programação dinâmica.