Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Bacharelado em Ciência da Computação

Construção e Análise de Algoritmos

Lista de exercícios 3

Algoritmos Gulosos

1. Desenhe um grafo qualquer com pelo menos oito vértices (A até H) e quinze arestas com pesos diferentes entre si. Mostre passo a passo a aplicação dos algoritmos de Kruskal e de Prim sobre esse grafo. Mostre ainda a aplicação do algoritmo de Dijkstra para o vértice A.

2. Informalmente, dizemos que um problema de otimização tem a propriedade de subestrutura ótima se "*um pedaço da solução ótima é solução ótima de um pedaço do problema*". Ou seja, a solução ótima pode ser formada a partir de soluções ótimas de subproblemas. Isso quer dizer que, dividindo a solução ótima em certos pedaços, esses pedaços da solução ótima são soluções ótimas de cada pedaço do problema. Dito isso, considere agora o problema de obter a árvore geradora mínima. Esse problema possui a propriedade de subestrutura ótima? Justifique isso mostrando duas formas diferentes de se dividir a solução ótima e que levam aos algoritmos de Prim e Kruskal.

3. O problema da seleção de atividades é o problema de encontrar, numa dada coleção de intervalos, uma subcoleção de tamanho máximo de intervalos dois a dois disjuntos. Nem todo algoritmo guloso resolve esse problema. Mostre que nenhuma das três idéias a seguir resolve o problema. Idéia 1: Escolha a atividade de menor duração dentre as que são compatíveis com as atividades já escolhidas. Idéia 2: Escolha uma atividade que seja compatível com as já escolhidas e intercepta o menor número possível de atividades ainda não escolhidas. Idéia 3: Escolha a atividade compatível com as já selecionadas que tenha o menor instante de início.

4. Escreva um algoritmo que obtenha um código de Huffman ternário, ou seja, um código de Huffman em que podemos codificar os símbolos com 0, 1 e 2. Escreva agora um algoritmo que obtenha um código de Huffman ao-contrário, ou seja, obtenha o pior código de prefixo possível. Exemplifique esses dois algoritmos e também o código de Huffman normal para o seguinte exemplo: um texto com 100.000 caracteres onde os símbolos são A, B, C, D, E, F e G com frequências 40%, 15%, 11%, 10%, 14%, 7% e 5%. Quantos bits esse texto codificado terá nesses algoritmos? Quantos bits esse texto teria se usássemos uma codificação de tamanho fixo? Compare cada algoritmo: melhorou ou piorou?

5. Considere um conjunto de livros numerados de 1 a n. Suponha que o livro i tem peso p_i e que $0 < p_i < 1$ para cada i. Problema: Dado n e os números $p_1, \ldots, p_n$, acondicionar os livros no menor número possível de envelopes de modo que cada envelope tenha no máximo 2 livros e o peso do conteúdo de cada envelope seja no máximo 1. Escreva um algoritmo eficiente que resolva esse problema. Aplique seu algoritmo a um exemplo interessante. Mostre que seu algoritmo está correto.

6. Escreva um algoritmo eficiente que receba como entrada um conjunto de variáveis $x_1, \ldots, x_n$ e dois conjuntos I e D de pares (x_i, x_j) de variáveis. Os pares (x_i, x_j) em I representam restrições de igualdade $x_i = x_j$ e os pares (x_a, x_b) em D representam restrições de desigualdade $x_a \neq x_b$. Seu algoritmo deve responder se é possível ou não satisfazer todas as restrições em I e em D. Por exemplo, a seguinte entrada não é satisfatível: $I = \{(x_1, x_2), (x_2, x_3), (x_3, x_4)\}$ e $D = \{(x_1, x_4)\}$.

7. Miguel deseja fazer uma festa e está decidindo quem deve chamar. Ele tem n amigos para convidar e tem uma lista dos pares de amigos que se conhecem. Ele quer que ninguém se sinta deslocado, mas também quer que a festa seja interessante e que pessoas que não se conhecem façam amizade. Assim ele se colocou as seguintes restrições: Para cada convidado, devem existir pelo menos dez pessoas na festa que ele conhece e dez pessoas na festa que ele não conhece. Faça um algoritmo que receba uma lista com n pessoas e uma lista com os pares dessas pessoas que se conhecem e devolva o maior número pessoas que poderão ser convidadas sob estas restrições. (a) Descreva com palavras como será o seu algoritmo. (b) Escreva o algoritmo em pseudo-código.