Teoria dos Grafos – COS 242
2024/2

Terceira Lista de Exercícios

ATENÇÃO! Para um melhor rendimento do processo de aprendizagem, responda às perguntas de forma precisa e concisa contemplando adequadamente cada questão.

Questão 1: Considere a lista de funções abaixo. Ordene a lista pela ordem de crescimento das funções. Ou seja, se uma função $f(n)$ antecede qualquer outra função $g(n)$ na lista ordenada, então temos que $f(n) = O(g(n))$.

- $f_1(n) = n^{2.5}$
- $f_2(n) = \sqrt{2n}$
- $f_3(n) = 10^5 n + 10$
- $f_4(n) = n \log_2 n$
- $f_5(n) = n^{\sqrt{n}}$
- $f_6(n) = 2^{n^2}$
- $f_7(n) = n^{\log n}$
- $f_8(n) = 2^{2^n}$

Questão 2: Considere o tempo de execução de diferentes algoritmos, dado pelas funções abaixo. Assuma que estes são os exatos número de operações que o algoritmo precisa executar em função do tamanho da entrada, n. Assuma ainda que você possui um computador capaz de executar 10^{10} operações por segundo (exemplo de processadores modernos, com *clock* de GHz). Assuma que você precisa obter o resultado em no máximo uma hora. Para cada um dos tempos de execução abaixo, determine o maior tamanho do problema (ou seja, valor de n) para o qual o algoritmo termina dentro da limitação de tempo.

- $T_1(n) = 2^5 n$
- $T_2(n) = n^2$
- $T_3(n) = n^3$
- $T_4(n) = 100n^2$
- $T_5(n) = n \log n$
- $T_6(n) = 1.5^n$
- $T_7(n) = n!$
- $T_8(n) = 2^{2^n}$

Questão 3: Dê um exemplo de DAG com n vértices que admite apenas uma ordenação topológica. Dê um exemplo de um DAG com n vértices que admite pelo menos n ordenações topológicas distintas.

Questão 4: Modifique o algoritmo de Dijkstra apresentado em aula de modo a encontrar não somente a distância do vértice inicial a todos os outros, mas também o caminho mínimo. Escreva o pseudo-código do algoritmo modificado.

Questão 5: Considere o grafo ilustrado na figura abaixo. Utilizando uma tabela (conforme apresentado em aula), mostre o funcionamento do algoritmo de Dijkstra passo-a-passo. Utilize o vértice G como vértice inicial.

Questão 6: O algoritmo de Dijkstra assume que os pesos associados às arestas são sempre positivos. Esta premissa foi necessária para provar a corretude do algoritmo, como vimos em aula. Dê um

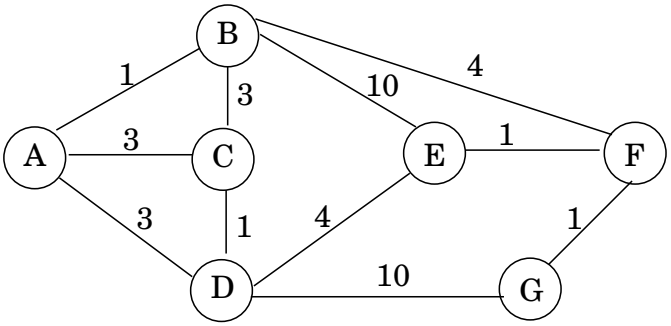

Figura 1: Grafo não-direcionado com pesos.

exemplo de um grafo direcionado com pesos negativos para o qual o algoritmo de Dijkstra produz resultados errados. Por que a prova da corretude do algoritmo falha quando temos pesos negativos?

Questão 7: Qual é a principal diferença entre o algoritmo A* e o algoritmo de Dijkstra? Em particular, qual informação o algoritmo A* necessita e como isto é explorado na execução do algoritmo? Dê um exemplo concreto onde o algoritmo A* utilizando a devida informação realiza bem menos passos que o algoritmo de Dijkstra.

Questão 8: Considere um jogo de quebra-cabeça em um tabuleiro de 3x3 com peças enumeradas de 1 à 8 cujo objetivo é movimentar as peças para atingir uma configuração ordenada. A figura abaixo ilustra um possível estado inicial e o estado final desejado.

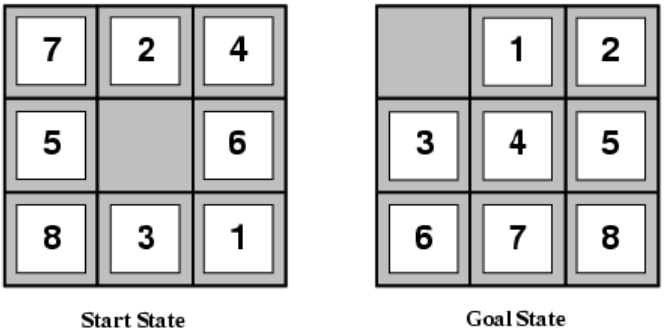

Figura 2: Quebra-cabeça de oito peças em tabuleiro 3x3.

Dado um estado inicial, você gostaria de encontrar o menor número movimentações necessárias para chegar ao estado final. Entretanto, encontrar a sequência mínima pode não ser computacionalmente viável se o tabuleiro for grande (ex. 5x5). Dessa forma, descreva um algoritmo eficiente para encontrar sequências de movimentações que sejam curtas (não necessariamente mínimas), deixando claro as principais ideias e estruturas de dados.

Questão 9: Considere o grafo ilustrado abaixo. Utilizando uma tabela (conforme apresentado em aula), mostre o funcionamento do algoritmo de Prim passo-a-passo. Utilize o vértice E como ponto de partida.

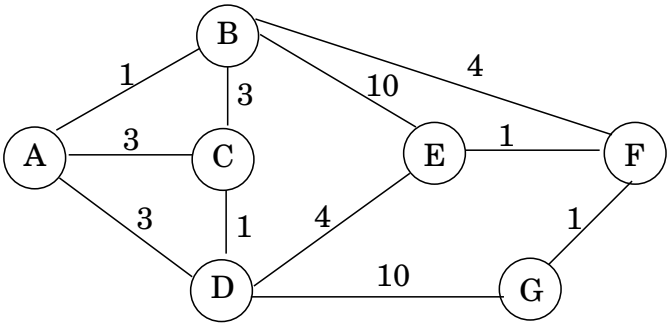

Figura 3: Grafo não-direcionado com pesos.

Questão 10: A essência do algoritmo de Kruskal é iniciar o processo guloso com um grafo sem arestas e ir adicionando arestas ao grafo em ordem crescente de peso, garantindo a cada passo que ciclos não se formem. Uma outra idéia é iniciar o processo guloso com o grafo original e ir removendo arestas em ordem decrescente de peso (mais pesada primeiro, etc). O que precisa ser garantido a cada passo do algoritmo? Descreva, em pseudo-código, um algoritmo baseado nesta idéia.

Questão 11: Considere o problema de encontrar a MST (árvore geradora de custo mínimo) em um grafo $G = (V, E)$ onde cada aresta $e \in E$ possui um custo $c_e \geq 0$ tal que os custos não são necessariamente distintos. Neste caso, G pode possuir várias árvores geradoras de custo mínimo. Suponha que você conheça uma árvore geradora $T \subset E$ tal que cada aresta $e \in T$, e pertence a *alguma* árvore geradora de custo mínimo de G. Podemos concluir que T é uma árvore geradora de custo mínimo de G? Prove este resultado ou dê um contra-exemplo.