Revisão para Prova Teórica

Raissa Pereira Miranda CJ3028941

1. Defina os seguintes conceitos relacionados a Grafos:

- Vértice (ou Nó): são pontos que podem ter valor e que representam entidades ou objetos, podendo estar conectados a outros vértices através de arestas.
- Aresta: são os relacionamentos que ligam um par de vértices, podendo ter um valor associado e ser direcionadas ou não.
- Grafo Direcionado: as arestas desse grafo possuem direções para os vértices.
- Grafo Não Direcionado: as arestas desse grafo não possuem direções para os vértices.
- Grafo Ponderado: as arestas desse grafo possuem pesos.

2. Explique a diferença entre a representação de grafos por Matriz de Adjacência e por Lista de Adjacência, citando vantagens e desvantagens de cada uma.

A matriz de adjacência é representada por uma tabela de n x n, onde n é o número de vértices, em que cada intersecção mostra se há ou não um relacionamento (aresta) entre os vértices, podendo ser representado por 0 e 1 ou pelo peso (se for grafo ponderado). Suas vantagens são: mais eficiente para grafos com muitas conexões, acesso rápido e implementação compacta. Suas desvantagens são: consome muita memória $O(n^2)$ e é menos eficiente para percorrer todos os vizinhos de um vértice.

Já a lista de adjacência é representada por uma lista onde cada vértice armazena uma lista de vértices adjacentes que estão conectados a ele. Suas vantagens são: uso mais eficiente de memória, pois só armazena as conexões existentes, e maior eficiência para percorrer os vizinhos de um vértice. Suas desvantagens são: implementação mais complexa e desempenho menor para grafos com muitas conexões.

3. Descreva o funcionamento do algoritmo de Busca em Largura (BFS) em grafos. Em que tipo de problema ele é mais adequado?

Em grafos a busca em largura é realizada descobrindo os vértices em camadas, partindo de um vértice inicial e visitando todos os seus vizinhos antes de avançar para os vizinhos desses vizinhos.

Primeiro, escolhe um vértice inicial e enfileira ele. Depois, desenfileira o primeiro vértice (marca como visitado), enfileira todos os seus vizinhos. Após isso, desenfileira o próximo da fila (marca como visitado) e enfileira seus vizinhos. Repete isso até ter visitados todos os vértices de camada por camada e a fila ficar vazia.

Esse algoritmo é mais adequado quando se quer encontrar o menor caminho em grafos não ponderados.

4. Explique o objetivo e o funcionamento do Algoritmo de Dijkstra. Qual é a sua principal aplicação?

O Algoritmo de Dijkstra tem como objetivo encontrar o caminho mais curto entre um vértice inicial (origem) e todos os outros vértices de um grafo ponderado com pesos positivos. Sua principal aplicação é em dispositivos GPS para encontrar a menor rota possível entre a localização atual e o destino.

5. Descreva o funcionamento do algoritmo de Busca em Profundidade (DFS) em grafos. Em que tipo de problema ele é mais adequado?

Em grafos a busca em profundidade é realizada descobrindo vértices e percorrendo os caminhos de suas adjacências recursivamente, explorando até o final de um caminho antes de retroceder para explorar outros caminhos, até que todos os vértices sejam visitados. A DFS é mais adequada em algoritmos que precisam explorar todos os caminhos possíveis para depois tomar uma decisão.

6. Qual a principal diferença entre Pesquisa Sequencial e Pesquisa Binária? Em que condições a Pesquisa Binária é mais eficiente?

A pesquisa sequencial verifica os elementos um por um (O(n)), da esquerda para a direita, até encontrar a chave buscada ou chegar no final do array. Ela funciona tanto para arrays ordenados ou não. Já a pesquisa binária divide o array ao meio a cada comparação, verificando se a chave buscada está na metade esquerda ou direita. Ela só funciona se o array estiver ordenado. A pesquisa binária é mais eficiente com arrays grandes e ordenados, reduzindo a quantidade de comparações necessárias O(log n).

7. Descreva o funcionamento do algoritmo de ordenação Bubble Sort. Qual a sua complexidade de tempo no pior caso?

A ordenação no Bubble Sort é feita através de trocas com os elementos adjacentes do array. Sua complexidade de tempo no pior caso é $O(n^2)$, tanto na versão não otimizada do algoritmo quanto na versão otimizada.

8. Explique o princípio de "dividir para conquistar" aplicado ao algoritmo Merge Sort. Qual a sua complexidade de tempo?

O Merge Sort usa o princípio de “dividir para conquistar” ao dividir o array em metades menores até chegar em um único elemento, ordenar esses subarrays recursivamente e, por último, combinar as partes ordenadas para formar o array final ordenado. Sua complexidade de tempo é O(n log n) em todos os casos.

9. Defina o conceito de Tabela Hash (ou Tabela de Dispersão) e explique sua principal finalidade na organização de dados.

A Tabela Hash é uma estrutura de dados desenvolvida para armazenar e buscar elementos rapidamente, usando uma função de hash que transforma uma chave específica em um índice dentro de um array. Sua principal finalidade é otimizar operações de busca, inserção e remoção de dados, possuindo uma complexidade de tempo de O(1).

10. Explique o que é a complexidade de tempo e a complexidade de espaço de um algoritmo. Por que é importante analisar a eficiência de algoritmos?

A complexidade de tempo de um algoritmo é o tempo que ele leva para executar as instruções conforme a entrada cresce, podendo ser um crescimento constante, logarítmico, linear, exponencial, etc. Já a complexidade de espaço está relacionado com a utilização de espaço extra para executar as instruções, sendo O(1) quando se utiliza nada ou quase nada de memória adicional e O(n) quando se utiliza memória adicional.

Questões extra:

11. Dado que o algoritmo Quick Sort tem complexidade média de O(n log n) e o algoritmo Bubble Sort tem complexidade O(n2), qual afirmação é correta?

A. O Quick Sort é sempre mais rápido, independentemente do tamanho do array.

B. O Bubble Sort pode ser mais eficiente para listas totalmente desordenadas.

C. O Quick Sort tende a ser mais eficiente para grandes conjuntos de dados.

D. Ambos têm a mesma eficiência quando o array já está ordenado.

E. O Bubble Sort consome menos memória e, por isso, é mais eficiente.

3. Qual dos seguintes algoritmos é considerado não estável?

A. Insertion Sort

B. Merge Sort

C. Counting Sort

D. Selection Sort

E. Bubble Sort

4. A complexidade de espaço do Merge Sort é O(n) porque:

A. Ele utiliza árvores binárias auxiliares para ordenação.

B. Ele precisa armazenar todos os elementos na memória secundária.

C. Ele cria subarrays temporários em cada etapa de intercalação.

D. Ele realiza muitas trocas que consomem espaço adicional.

E. Ele aloca espaço fixo, independente da entrada.

5. Após a primeira iteração completa do algoritmo Insertion Sort, considerando a entrada [7, 3, 5, 1, 2], qual será a nova sequência?

A. [3, 7, 5, 1, 2]

B. [3, 5, 7, 1, 2]

C. [7, 3, 5, 1, 2]

D. [3, 5, 1, 2, 7]

E. [7, 5, 3, 1, 2]

6. Qual algoritmo apresenta desempenho linear (O(n)) no melhor caso, quando o array já está ordenado?

A. Bubble Sort com otimização

B. Merge Sort

C. Counting Sort

D. Heap Sort

E. Selection Sort

7. Cite uma vantagem do Insertion Sort sobre o Selection Sort.

Uma vantagem que o Insertion Sort tem sobre o Selection Sort seria a sua capacidade de ter complexidade de tempo de O(n) no melhor caso quando o array já está ordenado, visto que o Selection Sort possui complexidade de tempo de $O(n^2)$ em todos os casos. Outra vantagem seria que o Insertion Sort é um algoritmo estável, diferente do Selection Sort que é instável.