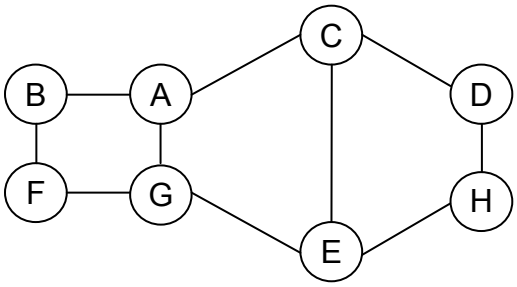
UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CURSO DE CIÊNCIA DA COMPUTAÇÃO – CAMPUS CASCAVEL
DISCIPLINA: ESTRUTURAS DE DADOS
PROFº ADAIR SANTA CATARINA

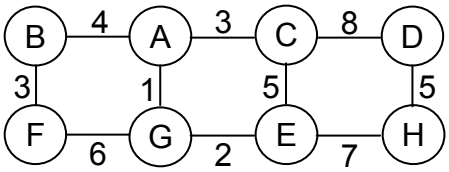
LISTA DE EXERCÍCIOS – GRAFOS, HASHING, ORDENAÇÃO DE DADOS E PROCESSAMENTO DE CADEIAS (BUSCA E COMPRESSÃO)

1) Considere o grafo abaixo:



É solicitado:
   a) Todos os caminhos não cíclicos de a até h;
   b) Todos os caminhos não cíclicos de c até e;
   c) Todos os caminhos não cíclicos de b até f.

2) Ainda considerando a figura anterior, ache todos os nós adjacentes aos nós a, f e g.
3) Qual o passeio da busca em profundidade sobre o grafo do exercício 1, iniciando do vértice A?
4) Qual o passeio da busca em largura sobre o grafo do exercício 1, iniciando do vértice A?
5) Qual a matriz de adjacência do grafo do exercício 1?
6) Qual a lista de adjacência do grafo do exercício 1?
7) Ache o caminho mínimo do nó A para todos os outros nós do grafo abaixo.



8) Dada a matriz de adjacência a seguir, faça o respectivo desenho do grafo orientado.

	A	B	C	D	E	F
A	0	3	4	0	2	1
B	0	0	2	0	0	3
C	0	0	0	2	6	1
D	2	6	1	0	1	2
E	0	0	0	0	0	3
F	0	0	0	0	0	0

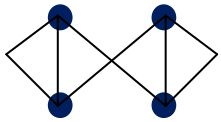
9) Escreva um algoritmo que determina se um nó é conectado aos demais nós de um grafo.
10) Escreva um algoritmo para determinar todos os nós que não são conectados a

nenhum outro nó de um grafo.

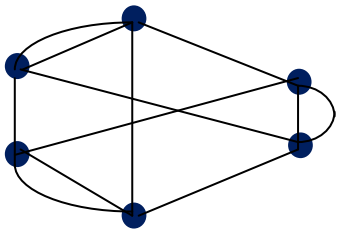
11) Escreva um algoritmo que calcule a soma dos graus de todos os nós de um grafo, usando lista de adjacência.

12) Considerando o teorema dos caminhos Eulerianos, responda as seguintes questões:
   a) Existe o caminho de Königsberg?
   b) Para cada um dos grafos apresentados a seguir, determine se existe um caminho Euleriano.

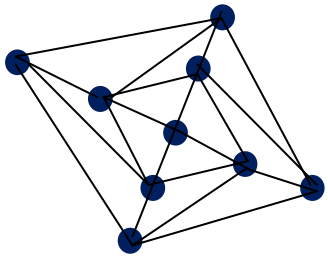
i)



ii)



iii)



13) Escreva um método para determinar o caminho mínimo em um grafo.

14) Escreva um algoritmo que combine os algoritmos Quicksort e Bubblesort da seguinte forma: Utilize Quicksort para obter partições (não ordenadas) de dimensão m (1 <= m <= n); então utilize o algoritmo Bubblesort para completar a operação. Note-se que este último deverá percorrer todo o vetor de n elementos minimizando, consequentemente, o esforço de manutenção de informações. Encontre o valor de m que minimize o tempo total de ordenação. Nota: Naturalmente, o valor ótimo de m será relativamente pequeno. Valerá, portanto, a pena deixar que o algoritmo Bubblesort percorra exatamente m – 1 vezes o vetor, ao invés de incluir um último passo, destinado a determinar o fato de que nenhuma permutação adicional é necessária.

15) Cada linha de um arquivo texto corresponde ao nome, endereço, item mais caro já adquirido pelo cliente, valor da maior compra realizada pelo cliente e valor da última compra realizada pelo cliente. Os campos são separados por “;”. Estima-se que este varejista, chinês, tenha aproximadamente 1 bilhão de clientes. Como o arquivo é muito grande para ser carregado para a memória da Raspberry de estimação deste varejista, ele solicitou que você crie um algoritmo capaz de ordenar o arquivo que se encontra gravado em um cartão Micro-SD. Dica: Crie pequenas corridas (que caibam na memória da Raspberry – 512 KB) e ordene-as com um método convencional de ordenação. Depois vá intercalando estas corridas (Merge), até que todo o arquivo esteja ordenado. Satisfaça a necessidade deste excêntrico bilionário chinês escrevendo um algoritmo em C++ para resolver o problema apresentado.

16) O meio do quadrado é um método bastante usado para realizar espalhamentos uniformes. Dada uma chave inteira k, com |k| bits, o método consiste em calcular k^2 e tomar m < |k| bits do meio do quadrado. Com m bits, temos 2^m valores possíveis, logo os valores gerados pela função estarão na faixa $[0..2^{m-1}]$ e o número de encaixes n da tabela a ser usada com este método deve ser uma potência de 2 ($n = 2^m$ ou $m = \log_2 n$). Observe no quadro a seguir, o espalhamento realizado pela função mq() que toma m = 3 bits no meio do quadrado:

k	k^2	Binário de k^2	mq(k)
7	49	0000000 **000** 110001	0
12	144	0000000 **010** 010000	2
15	225	0000000 **011** 100001	3
22	484	0000000 **111** 100100	7
25	625	0000001 **001** 110001	1
48	2304	0000100 **100** 000000	4
60	3600	0000111 **000** 010000	0

a) Note que, se k é um número pequeno, a maioria dos bits mais à esquerda do seu quadrado será zerada. Se a função resolve tomar m bits mais à esquerda, então os valores de hashing não serão muito aleatórios. Sabendo-se que as chaves k a serem espalhadas estão no intervalo [0..64], quais bits estariam sempre zerados em k^2, para qualquer k?
b) Uma determinada tabela não pode ter mais que 10 encaixes e as chaves estão entre 0 e 64. Codifique a função de hashing para realizar a espalhamento de chaves nesta tabela.
c) Escreva um programa para mostrar a distribuição realizada pela função definida no item anterior, quando todas as chaves (de 0 a 64) são armazenadas na tabela.

17) Considere uma tabela de espalhamento T com 20 encaixes, e uma função de espalhamento f(k) = ((k shl 10) shr 10). Sabendo-se que k tem 16 bits, explique por que motivo a função f não é uma boa escolha para realizar o espalhamento da tabela T.

18) Dadas as chaves 36, 26, 12, 5, 95 e uma tabela de espalhamento com 10 encaixes, desenvolva uma função de espalhamento que não produza colisões para este caso específico.

19) Considere uma empresa em que os produtos são designados por um código alfanumérico (contendo somente dígitos e hífen) no formato 999-99-9999. Projete uma função de hashing para distribuir estes códigos (chaves) numa tabela de espalhamento com 555 encaixes.

20) Dada a string: “inconstitucionaliscimamente e a palavra mais longa da lingua portuguesa”, encontre o vetor de padrões (string de busca = “iniquidade”) a ser empregado no algoritmo de busca de Knuth-Morris-Pratt. Depois disso execute o algoritmo de Knuth-Morris-Pratt, mostrando todos os passos realizados neste algoritmo considerando as strings fornecidas.

21) Os algoritmos de compressão de Huffman e LZW diferem em sua natureza. O primeiro é um método estatístico e o segundo baseado em dicionários. Comente sobre as características gerais de ambos evidenciando os pontos fortes e fracos de cada um deles.

22) O ponto crítico do algoritmo LZW é a busca por uma string no dicionário, no intuito de obter o código numérico associado àquela string. Usar um vetor para

armazenar o dicionário implicaria em utilizar uma busca sequencial. Quais estruturas de dados podemos utilizar para armazenar este dicionário, otimizando o processo de compressão do algoritmo LZW?

23) Monte a árvore de prefixos de Huffman para a seguinte string: “O rato roeu a roupa do rei de Roma. Enquanto Rosa foi dizer à Rita que o rato roia, o Rui Ramalho do rato a roer se ria! O rato também roeu a roupa refinada da rainha: enquanto o rei da Rússia resumia as rimas, a rainha - com raiva - resolveu remendar.”