1 - Escreva um algoritmo em C para verificar se, dados um Grafo G, representado por uma matriz de adjacências e dois vertices i e j, existe um caminho ligando o vértice i ao j. (Cuidado com loop infinito)

R:

```
bool existeCaminho(int i,int j, TMatrix M)
{
int n= M[0].lenght;
int k,l;
visitado[i]=true;
for (k=0;k<n;k++)
if((M[i][k]!=0)&&(visitado[k]==false))
{
if (k==j)
return true;
else
return existeCaminho(k,j,M);
}
return false; }
```

2 - O brasileirão é formado por 20 times em turno e returno. Use seus conhecimentos em teoria dos grafos para determinar o número de partidas de cada turno. Justifique sua resposta.

DESENHAR 20 GRAFOS E CONTAR AS ARESTAS...
Ou também fazer 1 rodada de cada turno e multiplicar pelo numero de rodadas no turno.
Da 190

3 - Calcular o tempo para ordenar 100 Mega usando merge-sort

Tempo seek 10 ms
Latencia Rotacional 8.3 ms
Taxa transferencia 1229 bytes/ms
buffer de 10000 bytes para escrita
2 mega de ram

Resposta:

1. Fase de Ordenação:
(a) Leitura dos registros para a memória com objetivo de criar as corridas
(b) Escrita das corridas ordenadas para o disco.
2. Fase de Intercalação
(a) Leitura das corridas para a intercala¸c˜ao (Merge-Sort)
(b) Escrita do arquivo final no Disco.

Vamos criar 50 corridas (100/2 = 50)
Run = 20 x (8.3 + 10) = 366 ms = 0,366
Transferencia = 100000/1229 * 1000 = 100/1229 = 81,366 segundos
A-)Tempo total de leitura e criação das corridas = 81,732 (0,366 + 81,366)

B-)O tempo total de escrita das corridas é identico, já que as operações são iguais 81,732 s.

C-)Tempo de merge: tem-se 50 corridas de 2 mega cada. Logo cada buffer irá armazenar 1/50 de uma corrida e cada corrida necessita de 50 acessos para ser lida por completo. Isso implica num total de 2500 seeks.
2500 * (10 + 8.3) = 2500 * (18,3) = 45750 ms = 45,75 s somados com os 81,73 segudos da transferencia de 100 mega temos 127,48 segundos.

D-) 100 mega * 1000000 = 100000000 / 10000 = 10000 seeks (um seek pra cada vez que o buffer enche)
[10000 * (8.3 + 10] / 1000 = 183 segundos
O tempo total é dê A + B + C + D = 81,73 + 81,73 + 127,48 + 183 = 7 minutos e 54 segundos.

4 - O que é aglomeração primaria ? E secundária ? Pq devemos evitar e como evitar.

Aglomeração primária:
Probabilidade do índice: inicialmente, a probabilidade de um índice ser selecionado é igual para todas as posiçõesP(i)=1T.
Aglomeração primária: depois que uma chave é colocada, a probalidade de alguns índices passam a ser maiores quando eles são a próxima chave vazia em uma lista de colisões. O rehash linear gera uma aglomeração primária bem grande.

Solução para aglomeração primária:
Tabela de permutação: fazer uma tabela de permutação que faz o papel do rehash.
Rehash com dois parâmetros: por exemplo: rh(c,j)=(h(c)+√j)%T. Porém gera uma aglomeração secundária.
Novo rehash: várias rotinas diferentes para rehash().
Split: duas rotinas lineares de rehash com duas constantes distintas. Só na primeira colisão, h(k)<kusa a primeira constante, senão usa a segunda constante. O problema é remover uma chave que é a primeira na lista de colisões, não sabemos de devemos usar a constante 1 ou 2.

5 - Escreva em C um código para inserir um a chave em uma tabela hash supondo que funções hash = h(chave) e rehash = rh(i) já implementadas e que é possível a remoca de elementos.

CASO ALGUEM SAIBA RESPONDA