Segunda lista de exercícios – Estruturas de Dados / Organização de Dados I
Prof. Vinícius Gusmão

1) Simule a inserção das seguintes chaves, na ordem abaixo, em uma tabela hash implementada sobre um array de 10 posições com resolução de colisões por encadeamento externo. PS.: Considere que uma chave é sempre inserida no início da lista encadeada correspondente à sua posição no array. PS.2.: A função hash utilizada é h(x) = (x (x -1)) % 10.

Chaves: 134, 22, 15, 1466, 11, 900, 9000, 90000, 2341, 433.

a) Indique claramente como estarão organizadas as chaves na estrutura final, depois de todas as inserções.
b) Qual o número médio de comparações realizadas durante a busca por uma chave que pertença à tabela hash, supondo que todas as chaves sejam buscadas com a mesma frequência?
c) Essa parece ser uma boa função hash para a dispersão dessas chaves?
d) Você teria uma função hash facilmente calculável que fosse melhor do que essa, para essas chaves?

2) Simule a inserção das mesmas chaves da questão anterior (com a mesma função hash), mas agora utilizando uma resolução de colisões diferente: para a inserção da chave x, se a posição h(x) estiver vazia, a chave x será inserida na posição h(x). Do contrário, a chave x será inserida na primeira posição vazia do array, a partir de h(x). Se o array estiver com todas as posições ocupadas de h(x) até sua última posição, então a busca pela posição vazia onde x deverá ser inserida continua a partir do começo do array.

a) Indique claramente como estarão organizadas as chaves na estrutura final, depois de todas as inserções.
b) Como deverá ser o algoritmo de busca numa tabela hash que utilize esse tipo de resolução de colisões?
c) Qual o número médio de comparações realizadas durante a busca por uma chave que pertença à tabela hash, supondo que todas as chaves sejam buscadas com a mesma frequência?

3) Suponha a existência de uma enorme lista formada por sequências de 3 números naturais distintos. Ex.: [2,7,8], [1,5,9], [3,4,1], [3,7,10], [7,10,3], [9,5,1] etc. Pense em um algoritmo que processe essas sequências (isto é, faça qualquer coisa que ele ache necessária com cada sequência lida), uma por uma, e retorne o número de sequências que foram processadas quando, pela primeira vez durante a execução do programa, todas as 3! = 6 permutações possíveis de um mesmo conjunto de 3 números já tiverem sido processadas, isto é, quando todas as 6 permutações possíveis (para um conjunto qualquer) já tiverem aparecido durante a varredura da lista.

Ex.: o algoritmo deve parar quando já houver lido as sequências [2,7,8], [1,5,9], [3,4,1], [3,7,10], [7,10,3], [9,5,1], [5,1,9], [2,17,3], [5,9,1], [9,5,1], [10,5,2], [1,9,5], respondendo 12, pois a décima-segunda sequência dessa lista, [1,9,5], completa todas as seis permutações possíveis dos números 1, 5 e 9. Em outras palavras, o algoritmo deve parar neste momento porque esta é a primeira vez em que todas as seis permutações possíveis de um certo conjunto qualquer de três números (no caso: 1, 5 e 9) já foram vistas.

[*A Questão 3 acima vale até um ponto extra na média se implementada corretamente e entregue até o dia 7/12 às 20:59 como anexo a um e-mail para vigusmao@dcc.ufrj.br com o subject começando por "[ORGDADOS1]". Pode ser resolvida em duplas. Identifique os autores no e-mail. O bônus na média só poderá ser concedido àqueles autores que estiverem presentes à aula do dia 8/12, quando entregarei as notas da P2. Cada um dos autores poderá ser indagado, isoladamente, sobre qualquer trecho do código ou sobre qualquer ponto do algoritmo que eu ache que precise de esclarecimento.*]

4) Simule a execução do QuickSort e do MergeSort para a sequência abaixo:

60 56 12 45 4 2 9 18 46 1 400 234

PS.: No QuickSort, considere que o pivô escolhido é sempre o primeiro de cada lista.
PS.2: Você deve mostrar, a cada passo, o que é feito pelo algoritmo, não apenas o resultado final.