IMD0029 – ESTRUTURAS DE DADOS BÁSICAS I

PROF. EIJI ADACHI M. BARBOSA

Lista de Exercícios – Ordenação

Obs.: Além das questões nesta lista, vejam também as questões das provas passadas.

Questão 1. Um algoritmo de ordenação é dito estável se não altera a posição relativa dos elementos que têm um mesmo valor. Em outras palavras, se dois elementos possuem valores iguais e a ordem relativa em que eles aparecem na sequência de dados inicial é mantida após a ordenação, então este algoritmo de ordenação é estável. Considere como exemplo o seguinte array v[1..5] = {1', 2', 3´, 1, 2, 3}. Se após a execução de um algoritmo de ordenação o array encontrar-se da seguinte forma: v = {1', 1, 2', 2, 3', 3}, então este algoritmo de ordenação é dito estável. Perceba que os números repetidos não alteraram as posições relativas que tinham antes da ordenação (elementos repetidos com aspas simples apareciam primeiro do que os repetidos sem aspas antes da ordenação e continuaram aparecendo primeiro após a ordenação). Neste contexto, considere o seguinte algoritmo de intercalação:

```
Intercalar( v[n], inicio1, inicio2, fim2 ) :
       DECLARE tmp[n]
       fim1 = inicio2-1, i = inicio1, j=inicio2, k=1
       ENQUANTO i ≤ fim1 E j ≤ fim2 FAÇA:
              SE v[i] < v[j] ENTÃO:
                     tmp[k] = v[i], i = i+1
              SENÃO:
                     tmp[k] = v[j], j = j+1
              FIM_SE
              k = k+1
       FIM_ENQUANTO
       ENQUANTO i ≤ fim1 FAÇA:
              tmp[k] = v[i], i = i+1, k = k+1
       FIM_ENQUANTO
       ENQUANTO j ≤ fim2 FAÇA:
               tmp[k] = v[j], j = j+1, k = k+1
       FIM_ENQUANTO
       COPIE tmp EM v
FIM
```

Suponha que você vai usar o algoritmo *Intercalar* como parte do seu algoritmo de ordenação por intercalação (*MergeSort*). O seu algoritmo de ordenação será estável? Se sim, o que garante sua estabilidade? Se não, o que precisa ser mudado para garantir sua estabilidade?

Questão 2: Considere um array A contendo N itens, sendo que cada item possui atributos *alpha* e *beta*. Neste contexto, projete algoritmos de ordenação de modo que os itens estejam ordenados em ordem crescente do atributo *alpha*. Em caso de empate entre dois itens com valores iguais para o atributo *alpha*, os itens com maior atributo *beta* deverão aparecer primeiro no array. Projete algoritmos de ordenação com base nos algoritmos: BubbleSort, InsertionSort e SelectionSort.

Dica 1: Crie uma função auxiliar chamada "comparar" que compara dois itens do array de entrada. Esta função será responsável por abstrair a lógica da relação de ordem. Se o item i1 vier primeiro do que i2, então a função retorna -1. Se o item i1 vier depois do que i2, então a função retorne +1. Se houver empate entre i1 e i2, então a função irá retorna 0. Daí na sua função de ordenação você irá simplesmente invocar a função comparar.

Dica 2: Não faça uma solução que primeiro ordena pelo atributo alpha e só depois pelo atributo beta. Use a função comparar para fazer uma única ordenação no array.

Questão 3: O algoritmo de ordenação por inserção funciona mantendo num array v[N] duas regiões distintas: uma região que fica entre [0...i], onde ficam os elementos já ordenados, e uma outra região entre [i+1...N-1], onde ficam os elementos não ordenados. A cada iteração, o algoritmo pega o primeiro elemento na região não ordenada e insere-o na região ordenada. Para isto, é necessário identificar a posição correta da inserção, bem como deslocar os elementos já ordenados para "abrir espaço" para o novo elemento. Projete uma versão do algoritmo de ordenação por inserção que usa a busca binária para encontrar a posição correta de onde o primeiro elemento da região não ordenada deve ser inserido na região ordenada. O uso da busca binária melhora a eficiência do algoritmo de ordenação por inserção? Justifique.

Questão 4: O algoritmo de ordenação por intercalação (merge sort) segue o princípio "Dividir para conquistar". A ideia geral deste algoritmo é dividir o vetor de entrada em duas partes e ordenar estas duas partes recursivamente. Após ordenadas, estas duas partes são intercaladas num array final com todos seus elementos ordenados. Projete um algoritmo de ordenação por intercalação que divide o vetor de entrada em 3 partes. Seu projeto deve contemplar também o algoritmo de intercalação para três vetores distintos (na verdade, ele deve receber os parâmetros que indicam os índices de cada um dos 3 sub-vetores, similar a versão vista em sala de aula para 2 sub-vetores).

Questão 5: (Desafio) Em estatística, a mediana é usada como um valor que resume a tendência central de uma determinada amostra, sendo o valor que separa a metade maior e a metade menor de uma amostra. Em outras palavras, a mediana é o valor que divide um conjunto de dados em duas partes iguais. No conjunto {1, 2, 3, 4, 5}, por exemplo, a mediana é 3. Nesta questão, projete um algoritmo que recebe como entrada um array A não ordenado de tamanho N e retorna um índice i tal que A[i] é a mediana do array. Sua solução deverá ter complexidade assintótica $\Theta(n)$.

Dica: Use uma versão modificada do algoritmo de partição (aquele usado no QuickSort) para construir a solução de encontrar a mediana do array.

Questão 6: Suponha que para encontrar o pivot utilizado pelo *QuickSort* nós usássemos o algoritmo de encontrar a mediana de um array não ordenado em Θ (n) (questão anterior). Qual seria a complexidade deste *QuickSort* no pior caso? Justifique sua resposta.

Questão 7: Considere V um array de n números inteiros distintos. Se $i < j$ e $V[i] > V[j]$ então o par (i,j) é chamado uma inversão de V. Por exemplo, o array $V[4] = \{10, 30, 20, 25\}$ possui duas inversões: (2, 3) e (2,4). Neste contexto:

A) Dado o conjunto $A = \{1, 2, 3, 4...N\}$, qual a permutação de A que tem o maior número de inversões? Quantas são?
B) Qual a relação entre o número de inversões num vetor e o tempo de execução do algoritmo de ordenação *BubbleSort*?
C) Considere um vetor v de tamanho n contendo apenas inteiros não repetidos. Projete um algoritmo que recebe v como parâmetro de entrada e retorna o número de inversões. Qual a complexidade do seu algoritmo?

Questão 8: Considere que um array não ordenado possui N elementos, mas apenas 3 elementos distintos (ex.: "a", "b" e "c"). Ou seja, o array possui diversas repetições de apenas 3 elementos. Neste contexto, crie um algoritmo de ordenação capaz de ordenar este array sem utilizar outro array auxiliar e com complexidade assintótica Θ (n).

Questão 9: Um anagrama é um jogo de palavras em que uma palavra é construída a partir do rearranjo das letras de uma outra palavra utilizando cada letra da palavra original exatamente uma vez, isto é, não mais nem menos do que isso. Por exemplo: Iracema é um anagrama da palavra America. Faça uma função que recebe duas strings e retorna verdadeiro se estas palavras são anagramas e falso caso contrário. Sua função deve ter complexidade O(n*lg(n)).

Questão 10: O índice h, ou h-index, é uma métrica utilizada para quantificar o impacto das produções científicas de um pesquisador com base nas suas produções mais citadas. O índice h é definido como o número de artigos que tiveram ao menos 'h' citações. Por exemplo: se um pesquisador tem índice h igual a 10, isto quer dizer que ele tem 10 artigos com pelos menos 10 citações; se um departamento tem índice h igual a 50 tem 50 artigos com pelo menos 50 citações. Suponha que um array de inteiros não-ordenados representa o número de citações dos artigos de um pesquisador, isto é, que cada número inteiro do array de entrada indica o número de citações de um determinado artigo de um pesquisador. Ex.: o array [10, 5, 3, 20] indica que o primeiro artigo do pesquisador teve 10 citações, o segundo artigo teve 5 citações, e assim por diante. Implemente uma função que recebe um array de inteiros representando as citações dos artigos de um pesquisador e retorna o seu índice h.

Sua solução deverá ter complexidade O(n*lg(n)).