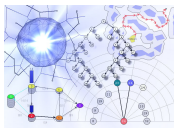

Pesquisa e Ordenação

1. Considere um vector dinâmico de números reais. Implemente uma função em linguagem C que permita calcular o histograma (de n intervalos) do vector. O histograma conta o numero de elementos do vector que pertencem a intervalos regulares entre os valores mínimo e máximo do vector.

   Protótipo: int* histograma(float *v, int tot, int n);

2. Implemente uma função em linguagem C que ordene os elementos de um vector (de inteiros) por ordem decrescente do respectivo numero de ocorrências. Exemplo: [2 3 3 4 2 5 1 2 1] é ordenado para [2 2 2 3 3 1 4 5].

   Protótipo: void ordenaOcorrencias(int *v, int tot);

3. Implemente uma função em linguagem C que, para uma série de valores inteiros representados num vector, coloque a 0 todos as sequências de valores decrescentes.

4. Implemente uma função em linguagem C que mostre os elementos de um vector de reais que são mínimos locais, considerando vizinhança de tamanho "n".

   Protótipo: void mostraMinimos(float *v, int tot, int n);

5. Um vector de co-ocorrências guarda informação sobre a forma como os elementos de um vector variam. Cada elemento na posição "i" guarda o numero de posições adjacentes no vector original em que o valor absoluto da diferença entre elementos é igual a "i". Exemplo: v=[1 3 2 4 2 3], tem como vector de co-ocorrências [0 2 3]. Implemente uma função em linguagem C que devolva o vector de co-ocorrências de um determinado vector.

   Protótipo: int* vectorCoOcorrencias(int *v, int tot, int *totV);
   //v=vector original, tot=total de elementos do vector original
   //totV=total de elementos do vector de co-ocorrências devolvido

6. Implemente uma função que receba um vector e um vector de co-ocorrências e verifique se este pode ter sido extraído a partir do vector original (1=sim, 0=não).

   Protótipo: int confirmaVectores(int *v, int totV, int *c, int totC);
   //v=vector original, c=vector de co-ocorrências.

7. Crie uma função em linguagem C que defina e preencha um vector dinâmico de inteiros com "n" valores aleatórios.

8. Utilize a função construída no exercício anterior para comparar o desempenho dos diferentes algoritmos de ordenação estudados nas aulas. Crie vectores de tamanho 100, 1 000, 100 000, 1 000 000, 100 000 000 e meça o desempenho (utilizando o código colocado em http://www.di.ubi.pt/~hugomcp/doc/tempoExecucao.c) de cada um dos algoritmos de ordenação.