Método dos Incrementos Decrescentes

Estudamos anteriormente o método de classificação/ordenação por inserção e vimos que o mesmo pode ser otimizado utilizando-se a busca binária ou implementando o mesmo sobre uma lista encadeada.

Porém, veremos agora como obter uma otimização mais significativa, o que denominaremos de ***classificação de incremento decrescente*** (ou ***Shell sort***), assim denominada em homenagem a seu descobridor Donald Shell.

UNIVASF

Método dos Incrementos Decrescentes

Esse método classifica subvetores separados do vetor original. Esses subvetores contêm todo *k*ésimo elemento do vetor original. O valor de *k* é chamado ***incremento***.

Por exemplo, se *k* é 5, o subvetor consistindo em *v*[0], *v*[5], *v*[10],... é classificado primeiro. Cinco subvetores, cada um contendo um quinto dos elementos do arquivo original, são classificados dessa maneira. São eles:

Subvetor 0 -> *v*[0] *v*[5] *v*[10] ...

Subvetor 1 -> *v*[1] *v*[6] *v*[11] ...

Subvetor 2 -> *v*[2] *v*[7] *v*[12] ...

Subvetor 3 -> *v*[3] *v*[8] *v*[13] ...

Subvetor 4 -> *v*[4] *v*[9] *v*[14] ...

Método dos Incrementos Decrescentes

O *i*ésimo elemento do *j*ésimo subvetor é *v*[*i* * 5 + *j*]. Se um incremento *k* diferente for escolhido, os *k* subvetores serão divididos de modo que o *i*ésimo elemento do *j*ésimo subvetor seja *v*[*i* * *k* + *j*].

Depois que os primeiros *k* subvetores estiverem classificados (geralmente por inserção simples), será escolhido um novo valor menor que *k* e o vetor será novamente particionado em novos conjuntos de subvetores. Cada um desses subvetores será classificado e o processo se repetirá novamente com um valor ainda menor que *k*.

Método dos Incrementos Decrescentes

Em algum momento, o valor de *k* será definido como 1, de modo que o subvetor consistindo no vetor inteiro será classificado.

Uma sequência decrescente de incrementos é determinada no início do processo inteiro. O último valor nessa sequência deve ser 1.

Por exemplo, se o vetor original for:

75 25 95 87 64 59 86 40 16 49

e a sequência (5, 2, 1) for escolhida, os seguintes subvetores serão classificados em cada iteração:

UNIVASF

Método dos Incrementos Decrescentes

Vetor original: **75 25 95 87 64 59 86 40 16 49**

k	**vetor resultante**
5	59 25 40 16 49 75 86 95 87 64
2	40 16 49 25 59 64 86 75 87 95
1	**16 25 40 49 59 64 75 86 87 95**

Com base no que foi discutido, codifique uma função que receba um vetor (de inteiros) e o número de elementos no mesmo e através do método *shell sort* ordene de forma crescente os elementos do vetor.

OBS.: Inicialize o incremento com n/2 e faça-o decrescer nesta taxa.

```
void shell_sort (int *v, int n) {
   int inc, j, k, y;
   inc=n;
   do{
      inc=inc>1?inc/2:1;
      for (j=inc; j<n; j++)
      {
         y=v[j];
         for (k=j-inc; k>=0 && y<v[k]; k-=inc)
            v[k+inc]=v[k];
         v[k+inc]=y;
      }
   }while(inc>1);  }
```

Método dos Incrementos Decrescentes

A análise da eficiência da classificação de Shell é complicada, em termos matemáticos, e está além do escopo desta disciplina.

No entanto, uma observação se faz importante, o desempenho do processo pode ser prejudicado, se os incrementos forem múltiplos entre si, pois cadeias logicamente adjacentes segundo incrementos múltiplos podem gerar passos redundantes.

UNIVASF

Classificação por Intercalação

Assim como a classificação quicksort, veremos agora, mais um método avançado baseado no preceito dividir para conquistar.

Isto é alcançado *a)* pela subdivisão do problema da ordenação em subproblemas menores e/ou *b)* pelo uso de áreas auxiliares, evitando o problema da liberação de casas ocupadas por elementos fora do lugar para receberem chaves relocalizadas (que ocorre nos métodos *in situ* em geral). Consegue-se, nesse grupo, desempenho de ordem **O**(n log n).

UNIVASF

Classificação por Intercalação

Analisaremos então, o método de **intercalação** ou ***merge sort***. Uma primeira aproximação ao mesmo é propiciada por um estudo da sua aplicação sobre uma *lista*, em vez de um vetor. Para classificar uma lista, pode-se adotar o seguinte procedimento:

1. **se** a lista tem um só elemento, já está ordenada;
2. **senão,** dividir a lista em duas sublistas, ordená-las e intercalá-las.

UNIVASF

Classificação por Intercalação

Os subproblemas que se apresentam são:

i. dividir uma lista;

ii. ordenar uma sublista;

iii. intercalar duas listas.

O subproblema de dividir uma lista pode ser solucionado colocando-se em uma sublista **n/2** elementos da lista original e o restante na outra sublista, onde **n** é o número de elementos na lista original. A ordenação das sublistas é resolvida pela aplicação recursiva da solução do problema principal. A intercalação de duas listas ordenadas consiste apenas em se inserir numa lista resultante cada um dos elementos das listas originais na ordem correta.

Classificação por Intercalação

No processo de intercalação, as listas podem ser vistas como filas: os elementos do início são comparados e o menor deles é retirado da fila correspondente e levado para a lista resposta. Quando uma fila esvaziar, simplesmente copia-se o restante da outra.

Por exemplo, ao se intercalar as listas L1 = (A D E K L P) e L2 = (B C F G) a sequência de movimentos seria a seguinte:

Classificação por Intercalação

Intercalação de listas ordenadas:

L1	L2	lista resultante
(A D E K L P)	(B C F G)	
(D E K L P)	(B C F G)	
(D E K L P)	(C F G)	
(D E K L P)	(F G)	
(E K L P)	(F G)	
(K L P)	(F G)	
(K L P)	(G)	
(K L P)	( )	

Observe como se aplicariam as duas fases (subdivisão e intercalação) ao se ordenar o vetor exemplo (armazenado numa lista). Em cada fase, os trechos resultantes de subdivisão estão sublinhados e os resultantes de intercalação estão em negrito.

Exemplo: Ordenação de lista por intercalação

Lista original	(75 25 95 87 64 59 86 40 16 49)
Subdivisão	(75 25 95 87 64) (59 86 40 16 49)
Subdivisão	(75 25 95) (87 64) (59 86 40) (16 49)
Subdivisão	(75 25)(95) (87) (64) (59 86) (40) (16) (49)
Subdivisão/**intercalação**	(75) (25) (**64 87**) (59) (86) (**16 49**)
Intercalação	(**25 75**) (**59 86**)
Intercalação	(**25 75 95**) (**40 59 86**)
Intercalação	(**25 64 75 87 95**) (**16 40 49 59 86**)
Intercalação	(**16 25 40 49 59 64 75 86 87 95**)

Tendo como base o que foi discutido, uma possível função, na linguagem C, para implementar a classificação de uma lista através do merge sort é:

```
void merge_sort (LISTA_ENC *pl) {
  if (tam(*pl)>1) {
    LISTA_ENC L1, L2;
    cria_lista(&L1);
    cria_lista(&L2);
    subdivide (pl, &L1, &L2);
    merge_sort (&L1);
    merge_sort (&L2);
    intercala (&L1, &L2, pl); } }
```

```
typedef struct nodo
{
  int inf;
  struct nodo * next;
}NODO;
typedef NODO * LISTA_ENC;
void cria_lista (LISTA_ENC *);
int eh_vazia (LISTA_ENC);
int tam (LISTA_ENC);
void ins (LISTA_ENC *, int, int);
int recup (LISTA_ENC, int);
void ret (LISTA_ENC *, int);
```

Classificação por Intercalação - Exercício

Como exercício, implemente as funções subdivide e intercala, constantes no algoritmo anterior.

UNIVASF