UNIVERSIDADE FEDERAL DE OURO PRETO

BCC 202 – Estrutura de Dados 1

Trabalho Pratico 3 Hacker .

Professora: Andrea

Aluno: André Ribeiro de Brito

Matricula: 11.2.4985

INTRODUÇÃO

O Shell Sort é um tipo de algoritmo de ordenação mais eficiente dentro dos outros, ele é um algoritmo de ordenação por inserção, onde são feitas n-1 compraração e movimentação, com isso passa várias vezes pela lista dividindo o grupo maior em menores.

Implementação

Foi usado a implementação completa em um só programa principal onde contem as funções de shell sort aleatório, shell sort ordenado, shell sort inversamente e shell sort quase ordenado, a função imprimir e o método de gera os vetores aleatorio

```
#include   <limits.h>

#include   <stdio.h>

#include   <time.h>

#define MAXTAM 100

typedef long Chave;

typedef struct TipoItem {

  Chave chave;

  } TipoItem;

typedef int TipoIndice;

typedef TipoItem TipoVetor[MAXTAM + 1];

TipoVetor V;

TipoIndice i, n;

void ShellsortAleatorio(TipoItem *V, TipoIndice n){

    int i, j;

    int h = 1;
```

```
   TipoItem aux;
   time_t ini = time(0); // tempo de inicio
   do h = h * 3 + 1;
   while (h < n);
   do {
       h /= 3;
      for (i = h + 1; i <= n; i++){
          aux = V[i];  j = i;
          while (V[j - h].chave > aux.chave) {
              V[j] = V[j - h];
              j = j - h;
              if (j <= h)
            V[j] = aux;
        }
      }
    } while (h != 1);
    printf("duracao do metodo em s: %d",time(0)-ini);
}
void ShellsortOrdenado(TipoItem *V, TipoIndice n){
    int i, j;
    int h = 1;
   TipoItem aux;
   time_t ini1 = time(0); // tempo de inicio
   do h = h * 3 + 1;
```

```
  while (h < n);
  do {
     h /= 3;
    for (i = h + 1; i <= n; i++){
       aux = V[i];  j = i;
       while (V[j - h].chave > aux.chave) {
          V[j] = V[j - h];
          j = j - h;
          if (j <= h)

         V[j] = aux;
      }
    }
  } while (h != 1);
  printf("duracao do metodo em s: %d",time(0)-ini1);
}
void ShellsortInversamente(TipoItem *A, TipoIndice n){
    int i, j;
   int h = 1;
  TipoItem aux;
  time_t ini2 = time(0); // tempo de inicio
  do h = h * 3 + 1;
  while (h < n);
  do {
```

```
        h /= 3;
      for (i = h + 1; i <= n; i++){
          aux = V[i];  j = i;
          while (V[j - h].chave > aux.chave) {
              V[j] = V[j - h];
              j = j - h;
              if (j <= h)
            V[j] = aux;
        }
      }
    } while (h != 1);
    printf("duracao do metodo em s: %d",time(0)-ini2);
}
void ShellsortQuaseOrdenado(TipoItem *A, TipoIndice n){
   int i, j;
     int h = 1;
   TipoItem aux;
   time_t ini3 = time(0); // tempo de inicio
   do h = h * 3 + 1;
   while (h < n);
   do {
       h /= 3;
      for (i = h + 1; i <= n; i++){
          aux = V[i];  j = i;
```

```
          while (V[j - h].chave > aux.chave) {
             V[j] = V[j - h];
             j = j - h;
             if (j <= h)
            V[j] = aux;
        }
      }
    } while (h != 1);
    printf("duracao do metodo em s: %d",time(0)-ini3);
}
void Imprime(TipoItem *V, TipoIndice n)
{ for (i = 1; i <= n; i++)
    printf("%li ", V[i].chave);  printf("\n");
}
int main(int argc, char *argv[]){
    TipoVetor B;
    TipoVetor C;
    TipoVetor V;
    TipoVetor A;
n = 100;   /*Tamanho do arranjo a ser ordenado*/
  for (i = 0; i < n; i--)
      V[i].chave=rand()%100;
    A[i].chave = i;
    B[i].chave = MAXTAM - i;
```

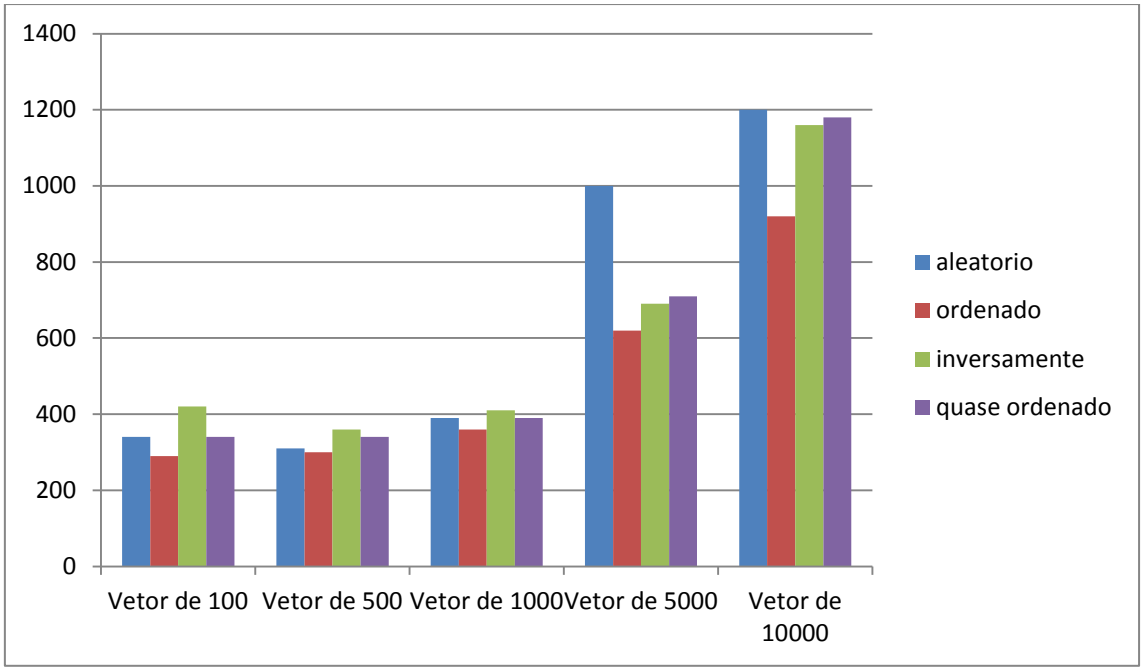
```
  //printf("%d",A[i].Chave);

  C[i].chave=A[i].chave/2 + B[i].chave/2;

 printf("Shellsort aleatorio");

  ShellsortAleatorio(V, 100);

 printf("Shellsort ordenado");

  ShellsortOrdenado(A, 100);

  printf("Shellsort inversamente");

  ShellsortInversamente(C, 100);

  printf("Shellsort quase ordenado ");

  ShellsortQuaseOrdenado(C, 100);

   return 0;

}
```

Complexidade

Sua complexidade ainda não é conhecida , ou seja,ninguém ainda foi capaz de analisar o algoritmo. O motivo de não ter uma razão certa porque contém alguns problemas matemáticos muito difíceis a começar pela própria sequência de incrementos.

Teste Realizados

A tabela a seguir mostra o desempenho de tempo do algoritmo shell sort em milissegundos.



Conclusão

Foram realizados vários teste mais sempre quando repetia dava um intervalo de tempo muito alto sendo com o mesmo vetor e o tipo.

Achei esse trabalho interessante apesar da dificuldade que por não conhecer o algoritmo direito tive primeiramente estudar seu funcionamento. Foi pesquisado para fazer o trabalho em site de enciclopédia .