UNIVERSIDADE ESTADUAL DE PONTA GROSSA

SETOR DE CIÊNCIAS AGRÁRIAS E SUAS TÉCNOLOGIAS

DEPARTAMENTO DE INFORMÁTICA

MAYSA LOVATTO LOPES  
RENANN RODRIGUES DA SILVA

ALGORITIMOS: ESTRUTURA DE DADOS

PONTA GROSSA

2014

MAYSA LOVATTO LOPES

RENANN RODRIGUES DA SILVA

ALGORITIMOS: ESTRUTURA DE DADOS

Trabalho apresentada na disciplina de

Estrutura de dados como requisito

de avaliação parcial referente ao

3º Bimestre. 2º Ano de Informática.

Professor Márcio Augusto de Souza

PONTA GROSSA

2014

//Trabalho de Estrutura de dados Terceiro Bimestre Parte 1

//Academicos Maysa Lovatto Lopes RA: 13015123

// Renann R. da Silva RA: 13106523

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#define TAM 10000

int vetor[TAM], tam\_heap=0, scratch[TAM];

void cria\_vetor() {

int i,aux;

for (i=0;i<TAM;i++)

vetor[i]=rand();//Cria vetor randômico

}

void imprime(){ //imprime o vetor

int i;

for(i=0;i<TAM;i++){

printf(" %d", vetor[i]);

}

}

// A bubble\_sort

void bubble\_sort(){

int pass = 1, i,aux;

bool sorted = false;

while((!sorted) && (pass < TAM)){

sorted = true;

for(i=0; i <= (TAM-pass-1); i++){

if(vetor[i] > vetor[i+1]){

aux = vetor[i];

vetor[i] = vetor[i+1];

vetor[i+1] = aux;

sorted = false;

}

}

pass++;

}

}

//------------------------------------------------------------------------------

// B Insertion sort (inserção simples)

void insertion\_sort(){

int k,i,x,aux;

bool found;

for(k=1;k<=TAM-1;k++){

x = vetor[k];

i = k-1;

found = false;

while((!found) && (i >=0)){

if(vetor[i] > x){

aux = vetor[i];

vetor[i] = vetor[i+1];

vetor[i+1] = aux;

i--;

}

else

found = true;

}

}

}

//------------------------------------------------------------------------------

// C selection sort

void selection\_sort(){

int i, j, atual, aux;

for (i = 0; i < (TAM-1); i++){

atual = i;

for (j = (i+1); j < TAM; j++) {

if(vetor[j] < vetor[atual]) {

atual = j;

}

}

if (i != atual) {

aux = vetor[i];

vetor[i] = vetor[atual];

vetor[atual] = aux;

}

}

}

//------------------------------------------------------------------------------

// D HEAPSORT = ok

void heapify(int i){

int L,r,max=0,aux=0;

L = (2\*i)+1;

r = (2\*i)+2;

if((L<tam\_heap) && (vetor[L]>vetor[i])){

max = L;

}

else

max = i;

if((r<tam\_heap) && (vetor[r]>vetor[max]))

max = r;

if(max != i){

aux = vetor[i];

vetor[i] = vetor[max];

vetor[max] = aux;

heapify(max);

}

}

void build\_healp(){

int i;

tam\_heap = TAM;

for (i=((TAM/2) - 1); i>=0; i--)

heapify(i);

}

void heapsort(){

int aux=0,k;

build\_healp();

for(k=(TAM-1);k >= 1;k--){

aux = vetor[0];

vetor[0] = vetor[k];

vetor[k] = aux;

tam\_heap--;

heapify(0);

}

}

//------------------------------------------------------------------------------

// E mergesort

void mergesort(int baixo, int alto){

int mid = 0, l = 0, h = 0, k = 0;

if(baixo < alto){

mid = (baixo + alto)/2;

mergesort(baixo,mid);

mergesort((mid + 1),alto);

l = baixo;

h = mid+1;

for(k = baixo;k <= alto; k++){

if((l <= mid) && ((h > alto) || (vetor[l] < vetor[h])))

scratch[k] = vetor[l++];

else

scratch[k] = vetor[h++];

}

for(k = baixo; k <= alto; k++)

vetor[k] = scratch[k];

}

}

//------------------------------------------------------------------------------

// F quicksort

int partition(int p, int r) {

int piv = vetor[p], i = (p - 1), j = (r + 1), aux = 0;

while(1) {

while(vetor[--j] > piv);

while(vetor[++i] < piv);

if(i < j) {

aux = vetor[i];

vetor[i] = vetor[j];

vetor[j] = aux;

}

else

return j;

}

}

void quicksort(int p, int r) {

int q=0;

if(p < r) {

q = partition(p, r);

quicksort(p, q);

quicksort((q+1), r);

}

}

//------------------------------------------------------------------------------

// função que verifica se o vetor esta em ordem

int verifica\_ordem(){

int i;

for(i=0; i<=TAM-1; i++){

if(vetor[i] > vetor[i+1]){

return 0;

}

}

return 1;

}

//------------------------------------------------------------------------------

main(){

clock\_t tempo1;

//Contagem de tempo do BubbleSort

cria\_vetor();

tempo1 = clock();

bubble\_sort();

tempo1 = clock() - tempo1;

printf("\nBubbleSorte: %.4f seg \n",(float)tempo1/CLOCKS\_PER\_SEC);

printf("Status: %s\n\n", verifica\_ordem() ? "Vetor ordenado" : "Vetor desordenado");

//--------------------------------------------------------------------------

//Contagem de tempo do InsertionSort

cria\_vetor();

tempo1 = clock();

insertion\_sort();

tempo1 = clock() - tempo1;

printf("InsertionSort: %.4f seg\n",(float)tempo1/CLOCKS\_PER\_SEC);

printf("Status: %s\n\n", verifica\_ordem() ? "Vetor ordenado" : "Vetor desordenado");

//--------------------------------------------------------------------------

//Contagem de tempo do SelectionSort

cria\_vetor();

tempo1 = clock();

selection\_sort();

tempo1 = clock() - tempo1;

printf("SelectionSort: %.4f seg \n",(float)tempo1/CLOCKS\_PER\_SEC);

printf("Status: %s\n\n", verifica\_ordem() ? "Vetor ordenado" : "Vetor desordenado");

//--------------------------------------------------------------------------

//Contagem de tempo do Heapsort

cria\_vetor();

tempo1 = clock();

heapsort();

tempo1 = clock() - tempo1;

printf("HeapSort: %.4f seg \n",(float)tempo1/CLOCKS\_PER\_SEC);

printf("Status: %s\n\n", verifica\_ordem() ? "Vetor ordenado" : "Vetor desordenado");

//--------------------------------------------------------------------------

//Contagem de tempo do MergeSort

cria\_vetor();

tempo1 = clock();

mergesort(0, TAM - 1);

tempo1 = clock() - tempo1;

printf("mergeSort: %.4f seg \n",(float)tempo1/CLOCKS\_PER\_SEC);

printf("Status: %s\n\n", verifica\_ordem() ? "Vetor ordenado" : "Vetor desordenado");

//--------------------------------------------------------------------------

//Contagem de tempo do QuickSort

cria\_vetor();

tempo1 = clock();

quicksort(0, TAM - 1);

tempo1 = clock() - tempo1;

printf("QuickSort: %.4f seg \n",(float)tempo1/CLOCKS\_PER\_SEC);

printf("Status: %s\n\n", verifica\_ordem() ? "Vetor ordenado" : "Vetor desordenado");

//--------------------------------------------------------------------------

getch();

}