# José Geraldo Oliveira de Andrade Júnior - 1430481323022

Diego João Sanches - 1430481323009

Luciano Leite Gustavo - 1430481323023

Relatório algoritmos de ordenação: bubble, insertion e selection

Primeiro escrevemos o bubble sort baseado em código feito em sala de aula. Utilizamos o make como builder do nosso projeto, sendo assim para realizar a compilação utilizamos o comando:

$ make

Todo o código citado está na seção de anexos. Utilizamos como contador uma variável float com mas com

Rodamos o software com o algoritmo *bubble* e podemos ver os seguintes resultados:

| Tipo do vetor | Contador comparação | Contador atribuição |
| --- | --- | --- |
| Aleatório | 16777216 | 67108864 |
| Ordenado crescente | 16777216 | 0 |
| Ordenado decrescente | 16777216 | 67108864 |

Rodamos o software com o algoritmo *insertion* e podemos ver os seguintes resultados:

| Tipo do vetor | Contador comparação | Contador atribuição |
| --- | --- | --- |
| Aleatório | 33554432 | 67108864 |
| Ordenado crescente | 99999 | 0 |
| Ordenado decrescente | 33554432 | 67108864 |

Rodamos o software com o algoritmo *selection* e podemos ver os seguintes resultados:

| Tipo do vetor | Contador comparação | Contador atribuição |
| --- | --- | --- |
| Aleatório | 16777216 | 67108864 |
| Ordenado crescente | 16777216 | 99999 |
| Ordenado decrescente | 16777216 | 67108864 |

Conclusão

Através desse estudo podemos comprovar empiricamente que os algoritmos de ordenação de comportam diferente de acordo com a ordenação do vetor.

O algoritmo bubble sort manteve-se com o contadores de comparação e atribuição constantes com os vetores decrescente e aleatório, o que era esperado. Quando o

Anexos

Conteúdo do arquivo trabalho1.c:

# include "trabalho1.h" // INCLUI A BIBLIOTECA ou HEADER trabalho.h A EXECUCAO DO PROGRAMA

int main() {

int \* vetor\_aleatorio;

int \* vetor\_desc;

int \* vetor\_asc;

vetor\_aleatorio = ( int \* ) malloc ( T \* sizeof(int));

vetor\_desc = ( int \* ) malloc ( T \* sizeof(int));

vetor\_asc = ( int \* ) malloc ( T \* sizeof(int));

int i = 0;

float contador\_comparacao = 0;

float contador\_atribuicao = 0;

printf("Vamos testar o buble para ordenar o vetor aleatorio\n");

for (i=0;i<10;i++) {

printf("Gerando o vetor aleatorio...\n");

gera(vetor\_aleatorio, T);

printf("Vetor aleatorio gerado \n");

printf("Agora vamos ordenar o vetor aleatorio...\n");

bubble(vetor\_aleatorio, T, &contador\_comparacao, &contador\_atribuicao);

}

printf("Vetor aleatorio: Contador comparacao %.f Contador atribuicao %.f \n", contador\_comparacao, contador\_atribuicao);

//Vetor aleatorio: Contador comparacao 16777216 Contador atribuicao 67108864

i = 0;

contador\_comparacao = 0;

contador\_atribuicao = 0;

printf("Vamos testar o bubble para ordenar o vetor crescente\n");

for (i=0;i<10;i++) {

printf("Gerando o vetor crescente...\n");

gera\_vetor\_crescente(vetor\_asc, T);

printf("Vetor crescente gerado \n");

printf("Agora vamos ordenar o vetor crescente...\n");

bubble(vetor\_asc, T, &contador\_comparacao, &contador\_atribuicao);

}

printf("Vetor crescente: Contador comparacao %.f Contador atribuicao %.f \n", contador\_comparacao, contador\_atribuicao);

// Vetor crescente: Contador comparacao 16777216 Contador atribuicao 0

i = 0;

contador\_comparacao = 0;

contador\_atribuicao = 0;

printf("Vamos testar o bubble para ordenar o vetor decrescente\n");

for (i=0;i<10;i++) {

printf("Gerando o vetor decrescente...\n");

gera\_vetor\_decrescente(vetor\_desc, T);

printf("Vetor decrescente gerado \n");

printf("Agora vamos ordenar o vetor decrescente...\n");

bubble(vetor\_desc, T, &contador\_comparacao, &contador\_atribuicao);

}

printf("Vetor decrescente: Contador comparacao %.f Contador atribuicao %.f \n", contador\_comparacao, contador\_atribuicao);

//Vetor decrescente: Contador comparacao 16777216 Contador atribuicao 67108864

return 0;

}

void gera\_vetor\_decrescente(int \* vetor, int n) {

int i=0;

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

vetor[i] = n-i;

}

}

void gera\_vetor\_crescente(int \* vetor, int n) {

int i=0;

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

vetor[i] = i;

}

}

/\*

\* Essa funcao testa se o item já existe na matriz

\*

\* int v: matriz de vetores

\* int n: numero de itens

\* int numero\_gerado: pa

\*\*/

int tem\_repeticao(int \* v, int n, int numero\_gerado) {

int i = 0;

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

if (v[i] == numero\_gerado) {

return 1; // ja existe

}

}

return 0; // nao tem repeticao, urru

}

/\* FUNCAO UTILIZADA PARA GERAR UM VETOR DE NMEROS ALEATRIOS \*/

void gera (int \*v, int n){

int i=0;

int numero;

while(i<n) {

numero = rand();

if (!tem\_repeticao(v, n, numero)) {

v[i] = numero;

++i;

}

}

}

/\* FUNO UTILIZADA PARA MOSTRAR O VETOR NA TELA \*/

void mostra (int \*v, int n, char \*msg){

int i;

printf ( "\n%s\n\n", msg );

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

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

}

printf ("\n");

}

/\* FUNO UTILIZADA PARA MOSTRAR O VETOR "ORDENADO AO CONTRRIO" NA TELA \*/

void mostraInvertido (int \*v, int n, char \*msg){

int i = 0;

printf ( "\n%s\n\n", msg );

for ( i = n - 1; i >= 0; i-- ){ // colocado n - 1 para correo do "primeiro item invertido"

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

}

printf ("\n");

}

/\* FUNCAO UTILIZADA PARA ORDENAR O VETOR UTILIZANDO O ALGORTMO "BUBBLE SORT" \*/

void bubble (int \* v, int n, float \* cc, float \* ca) {

int i, j;

for ( i = 1; i < n; i++ ) {

(\*cc) = (\*cc)+1;

for ( j = 0; j < n - i; j++ ) {

(\*cc) = (\*cc)+1;

if (v[j] > v[j+1]) {

(\*cc) = (\*cc)+1;

troca (v, j);

(\*ca) = (\*ca) + 3;

}

}

}

}

/\* FUNO UTILIZADA PARA TROCAR A POSIO E ORDENAR O VETOR "..." \*/ // MELHORAR A DESCRIO <<<<<<<<<<<

void troca (int \*v, int pos){

int aux = v[pos];

v[pos] = v[pos+1];

v[pos+1] = aux;

}

/\* FUNO UTILIZADA PARA TROCAR A POSIO E ORDENAR O VETOR "..." \*/ // MELHORAR A DESCRIO <<<<<<<<<<<<

void trocaProximo (int \*v, int pos){

int aux = v[pos];

v[pos] = v[pos + 1];

v[pos + 1] = aux;

}

/\* FUNO UTILIZADA PARA TROCAR A POSIO E ORDENAR O VETOR "..." \*/ // MELHORAR A DESCRIO <<<<<<<<<<<

void trocaAnterior (int \*v, int pos){

int aux = v[pos];

v[pos] = v[pos - 1];

v[pos - 1] = aux;

}

/\* FUNO UTILIZADA PARA TROCAR A POSIO E ORDENAR O VETOR "..." \*/ // MELHORAR A DESCRIO <<<<<<<<<<<

void trocaMinimo (int \*v, int pos){

int min;

int aux = v[min];

v[min] = v[pos];

v[pos] = aux;

}

/\* FUNO UTILIZADA PARA ORDENAR O VETOR UTILIZANDO O ALGORTMO "BUBBLE SORT" \*/

void bubbleSort (int \*v, int n){

int i, j;

for ( i = 1; i < n; i++ ){

for ( j = 0; j < n - i; j++ ){

if (v[j] > v[j+1]){

trocaProximo (v, j);

}

}

}

}

/\* FUNO UTILIZADA PARA ORDENAR O VETOR UTILIZANDO O ALGORTMO "INSERTION SORT" \*/

void insertionSort (int \*v, int n){

int i, j, key;

for ( i = 1; i < n; i++ ){

key = v[i];

while ( i > 0 && v[i -1] > key ){

trocaAnterior (v, j);

/\*

j = v[i];

v[i] = v[i -1];

v[i -1] = j;

\*/

--i;

}

}

}

/\* FUNO UTILIZADA PARA ORDENAR O VETOR UTILIZANDO O ALGORTMO "SELECTION SORT" \*/

void selection (int \*v, int n){

int i, j, x, min, k;

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

min = i;

for ( j = i + 1; j < (n - i) ; j++ ){

if (v[min] > v[j]){

min = j;

}

trocaMinimo (v, j);

/\*

x = v[min];

v[min] = v[j];

v[j] = x;

\*/

}

}

}

/\* FUNO UTILIZADA PARA ORDENAR O VETOR UTILIZANDO O ALGORTMO "BUBBLE SORT" \*/

int busca\_binaria (int \*v, int n, int x){

int meio, ini = 0, fim = n - 1;

while ( ini < fim ){

meio = ( ini + fim ) / 2;

if ( v[meio] == x ) {

return meio;

}

if ( x > v[meio] ){

ini = meio + 1;

}

else {

fim = meio - 1;

}

}

return -1;

}

Conteúdo do arquivo trabalho1.h:

#include <stdio.h>

# include <stdlib.h>

# include <time.h>

# define T 100000 // CONSTANTE DEFINIDA PARA A QUANTIDADE DE ITENS GERADOS NO VETOR

void gera\_vetor\_crescente(int \*, int);

void gera\_vetor\_decrescente(int \*, int);

void gera (int \*, int); // FUNCAO CRIADA PARA GERAR O VETOR

void bubble (int \*, int, float \*, float \*); // FUNCAO CRIADA PARA ORDENAR O VETOR UTILIZANDO O ALGORITMO "BUBBLE SORT"

void troca (int \*, int); // FUNCAO CRIADA PARA TROCAR A POSICAO E ORDENAR O VETOR "..."

void insertionSort (int \*, int, float \*, float \*); // FUNCAO CRIADA PARA ORDENAR O VETOR UTILIZANDO O ALGORITMO "INSERTION SORT"

void selection (int \*, int, float \*, float \*); // FUNCAO CRIADA PARA ORDENAR O VETOR UTILIZANDO O ALGORITMO "SELECTION SORT"

void trocaAnterior (int \*, int); // FUNCAO CRIADA PARA TROCAR A POSICAO E ORDENAR O VETOR "..."

void trocaMinimo (int \*, int, int); // FUNCAO CRIADA PARA TROCAR A POSICAO E ORDENAR O VETOR “..."

Conteúdo do arquivo makefile:

all: trabalho.o

gcc -o target/trabalho.sh target/trabalho.o

trabalho.o: trabalho1.c trabalho1.h

gcc -c trabalho1.c -o target/trabalho.o

clean:

rm -f \*.o \*.sh target/\*