**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Algorytmy i struktury danych | **Zadanie** | 11\_01 |
| **Autor** | Kamil Borkowski | **Grupa** | WCY22IY1S1 |
| **Temat** | Merge Sort | | |

1. Treść

Zaimplementuj algorytm sortowania metodą Merge Sort.

* 1. Metoda realizacji

Program wykorzystuje prosty interfejs użytkownika do wyboru operacji, wczytywania danych i sortowania za pomocą algorytmu Merge Sort.

* 1. Założenia / ograniczenia dotyczące danych:
     1. Dane wejściowe *(opis)*

Wartości do sortowania – wczytywana z pliku/klawiatury/generatora

* + 1. Dane wyjściowe *(opis)*

Wartości posortowane – wyświetlone na ekranie.

1. Realizacja
   1. Algorytm *(lista kroków lub schemat blokowy)*
2. Algorytm zaczyna od podziału tablicy na dwie równe części.
3. Oblicza środek tablicy, np. mid = (pocz + kon) / 2, gdzie pocz to początkowy indeks tablicy, a kon to końcowy indeks.
4. Rekurencyjne wywołanie mergeSort:
5. Wywołuje rekurencyjnie funkcję mergeSort dla lewej i prawej połowy tablicy.
6. mergeSort(arr, pocz, mid) dla lewej połowy.
7. mergeSort(arr, mid + 1, kon) dla prawej połowy.
8. Scalanie dwóch posortowanych części:
9. Po zakończeniu rekurencyjnych wywołań, łączy dwie posortowane części.
10. Tworzy dwie pomocnicze tablice (poczarr i konarr), aby przechować lewą i prawą część tablicy.
11. Łączy te dwie tablice w jedną posortowaną tablicę.
12. Kopiowanie do oryginalnej tablicy:
13. Kopiuje scaloną tablicę z powrotem do oryginalnej tablicy.

Kod źródłowy

//================================================

//Zadanie 11\_01 jezyk C++

//Merge Sort

//WCY22IY1S1 Borkowski Kamil

//================================================

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <cstdio>

#include <fstream>

using namespace std;

//funkcja merge sort

void mergeSort(int arr[], int pocz, int kon)

{

if (pocz >= kon)return;

int mid = pocz + (kon - pocz) / 2;

mergeSort(arr, pocz, mid);

mergeSort(arr, mid + 1, kon);

int subarrOne = mid - pocz + 1;

int subarrTwo = kon - mid;

int \*poczarr = (int \*)malloc(subarrOne \* sizeof(int));//tablie pomocnicze

int \*konarr = (int \*)malloc(subarrTwo \* sizeof(int));

for (int i = 0; i < subarrOne; i++)//koiowanie elementow do pomocniczych tablic

poczarr[i] = arr[pocz + i];

for (int j = 0; j < subarrTwo; j++)

konarr[j] = arr[mid + 1 + j];

int indexOfSubarrOne = 0, indexOfSubarrTwo = 0;

int indexOfMergedarr = pocz;

while (indexOfSubarrOne < subarrOne && indexOfSubarrTwo < subarrTwo) {//zlaczamy pomocnicze tablice w jedna

if (poczarr[indexOfSubarrOne] <= konarr[indexOfSubarrTwo]) {

arr[indexOfMergedarr] = poczarr[indexOfSubarrOne];

indexOfSubarrOne++;

}

else {

arr[indexOfMergedarr] = konarr[indexOfSubarrTwo];

indexOfSubarrTwo++;

}

indexOfMergedarr++;

}

while (indexOfSubarrOne < subarrOne) {//kopiowanie pozostalych elementow z lewej pomocniczej

arr[indexOfMergedarr] = poczarr[indexOfSubarrOne];

indexOfSubarrOne++;

indexOfMergedarr++;

}

while (indexOfSubarrTwo < subarrTwo) {//kopiowanie pozostalych elementow z lewej pomocniczej

arr[indexOfMergedarr] = konarr[indexOfSubarrTwo];

indexOfSubarrTwo++;

indexOfMergedarr++;

}

free(poczarr) ;

free(konarr) ;

}

int main()

{

srand((time(0)));

int arr[10000] = {};

int i,k=0,n,g,d,w;

int pom=1,z;//pom oblusguje interfejs, z= nr komendy z interfejsu

cout << "WCY22IY1S1 Borkowski Kamil" << endl;

cout << "Program sortowania Merge Sort" << endl;

cout << "[0] - wyjscie z programu" << endl;

cout << "[1] - wczytaj dane z pliku(we.txt)" << endl;

cout << "[2] - wczytaj dane z generatora" << endl;

cout << "[3] - wczytaj dane z klawiatury" << endl;

cout << "=========================================" << endl;

while(pom){

cout << "Wprowadz nr operacji: ";

cin >> z;

switch (z){

case 0:

cout<<"Wychodzenie..."<<endl;

pom = 0;

break;

case 1:

{

k=0;

ifstream in;

in.open("we.txt");

int p;

i=0;

while(in>>p)

{

arr[i]=p;

i++;

}

n=i;

cout<<"Tablica zostala usupelniona danymi z pliku we.txt"<<endl<<endl;

cout <<"Przed sortowaniem metoda merge sort: "<<endl;;

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

{

cout<<arr[i]<<'\t';

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

mergeSort(arr,0,n-1);

k=0;

cout<<endl<<"Po sortowaniu metoda merge sort: "<<endl;;

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

cout << arr[i] <<'\t';

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

cout<<endl;

in.close();

break;

}

case 2:

{

k=0;

cout<<"Podaj ilosc liczb do sortowania"<<endl;

cin>>n;

cout<<"Podaj zakres liczb do sortowania"<<endl;

cout<<"Granica dolna: ";

cin>>d;

cout<<"Granica gorna: ";

cin>>g;

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

arr[i]=(rand() % (g-d+1))+d; // liczby od 1 do 100

}

cout <<"Przed sortowaniem metoda merge sort: "<<endl;;

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

{

cout<<arr[i]<<'\t';

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

mergeSort(arr,0,n-1);

k=0;

cout<<endl<<"Po sortowaniu metoda merge sort: "<<endl;;

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

cout << arr[i] <<'\t';

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

cout<<endl;

break;

}

case 3:

{

k=0;

cout<<"Podaj ilosc liczb do sortowania"<<endl;

cin>>n;

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

{

w=0;

cout<<"Podaj argument: ";

cin>>w;

arr[i]=w;

}

cout <<"Przed sortowaniem metoda merge sort: "<<endl;;

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

{

cout<<arr[i]<<'\t';

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

mergeSort(arr,0,n-1);

k=0;

cout<<endl<<"Po sortowaniu metoda merge sort: "<<endl;;

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

cout << arr[i] <<'\t';

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

cout<<endl;

break;

}

default:

{

cout<<"Podano bledny nr operacji"<<endl<<endl;

break;

}

}

}

return 0;

}

* 1. Dane wejściowe *(ekran uruchomienia programu)*

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

* 1. Dane wyjściowe *(ekran uruchomienia programu)*

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

1. Wnioski *(złożoność asymptotyczna algorytmu przy użyciu notacji* ***O lub innej i inne wnioski)***

Złożoność obliczeniowa programu Merge Sort to O(n log n).