**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Algorytmy i struktury danych | **Zadanie** | 9\_03 |
| **Autor** | Kamil Borkowski | **Grupa** | WCY22IY1S1 |
| **Temat** | Quick Sort | | |

1. Treść

Zaimplementuj algorytm sortowania metodą Quick Sort (ang. Quick Sort).

* 1. Metoda realizacji

Program wykorzystuje prosty interfejs użytkownika do wyboru operacji, wczytywania danych i sortowania za pomocą algorytmu Quick 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. Funkcja Quick Sort wywołuje funkcję partition do podziału tablicy na dwie części i sortuje obie części rekurencyjnie.
3. Początkowy pivot to pierwszy element tablicy.
4. Funkcja przegląda tablicę, licząc elementy mniejsze lub równe pivotowi.
5. Następnie zamienia pivot z elementem na odpowiedniej pozycji.
6. Kolejno przesuwa elementy mniejsze i większe wokół pivota.

Kod źródłowy

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

//Zadanie 9\_03 jezyk C++

//Quick Sort

//WCY22IY1S1 Borkowski Kamil

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

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <cstdio>

#include <fstream>

using namespace std;

//funkcja dzielaca tablica, znajdujaca pivot

int partition(int arr[], int start, int kon) {

// Wybierz pivot jako pierwszy element tablicy

int pivot = arr[start];

int ilo = 0;

// Policzenie elementow mniejszych lub rownych od pivota

for (int i = start + 1; i <= kon; i++) {

if (arr[i] <= pivot)

ilo++;

}

// Obliczenie indeksu pivota po przesunieciu elementow mniejszych

int pivotIndex = start + ilo;

// Zamiana elementu pivot z jego prawidlowa pozycja

int temp = arr[pivotIndex];

arr[pivotIndex] = arr[start];

arr[start] = temp;

int i = start, j = kon;

// Przesuwanie elementow mniejszych i wiekszych wokol pivota

while (i < pivotIndex && j > pivotIndex) {

while (arr[i] <= pivot) {

i++;

}

while (arr[j] > pivot) {

j--;

}

if (i < pivotIndex && j > pivotIndex) {

// Zamiana elementow

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

i++;

j--;

}

}

return pivotIndex;

}

void quickSort(int arr[], int start, int kon) {

// Warunek zakonczenia rekurencji

if (start >= kon)

return;

// Wykonaj partition, otrzymaj indeks pivota

int p = partition(arr, start, kon);

// Rekurencyjne sortowanie podtablic przed i po pivocie

quickSort(arr, start, p - 1);

quickSort(arr, p + 1, kon);

}

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 Quick 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 quick sort: "<<endl;;

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

quickSort(arr, 0, n - 1);

k=0;

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

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

quickSort(arr, 0, n - 1);

k=0;

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

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

quickSort(arr, 0, n - 1);

k=0;

cout<<endl<<"Po sortowaniu metoda quick 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, wyświetlacz

Opis wygenerowany automatycznie

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

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

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

Złożoność obliczeniowa algorytmu Quick Sort w najgorszym przypadku wynosi O(n^2), ale w przeciętnym przypadku i w przypadku optymalnym wynosi O(nlogn).