**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Algorytmy i struktury danych | **Zadanie** | 9\_01 |
| **Autor** | Kamil Borkowski | **Grupa** | WCY22IY1S1 |
| **Temat** | Shell Sort | | |

1. Treść

Zaimplementuj algorytm sortowania metodą Shella (ang. Shell Sort).

* 1. Metoda realizacji

Program wykorzystuje prosty interfejs użytkownika do wyboru operacji, wczytywania danych i sortowania za pomocą algorytmu Shell 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. Tworzymy tablicę pomocniczą o nazwie arr. Tablica ta będzie przechowywała elementy w trakcie sortowania.
3. Początkowo gap ustawione jest na połowę długości tablicy (n/2)
4. Pętla wykonuje się, dopóki gap jest większe niż 0.
5. Dla każdego i, funkcja zapamiętuje bieżącą wartość elementu tablicy (temp = arr[i]).
6. Iteruje od i w kierunku początku tablicy (j = i) i przesuwa elementy w odstępie gap.
7. Warunek pętli to j >= gap && arr[j - gap] > temp.
8. Po zakończeniu pętli przypisuje zapamiętaną wartość temp do odpowiedniego miejsca w tablicy (arr[j]).

Kod źródłowy

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

//Zadanie 9\_01 jezyk C++

//Shell Sort

//WCY22IY1S1 Borkowski Kamil

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

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <cstdio>

#include <fstream>

using namespace std;

int shellSort(int arr[], int n)

{

// Iteracja przez rozne wartosci odstepu w algorytmie Shell Sort

for (int gap = n/2; gap > 0; gap /= 2)

{

// Iteracja przez elementy tablicy, zaczynajac od elementu o indeksie gap

for (int i = gap; i < n; i += 1)

{

// Zapamietanie wartosci biezacego elementu tablicy

int temp = arr[i];

int j;

// Przesuwanie elementów tablicy w kierunku poczatku,

// jesli sa wieksze niz wartos temp i znajduja sie w odstepie gap

for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)

{

arr[j] = arr[j - gap];

}

// Umieszczenie zapamietanej wartosci temp w odpowiednim miejscu

arr[j] = temp;

}

}

return 0;

}

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 Shell 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;

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

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

shellSort(arr,n);

k=0;

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

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

shellSort(arr,n);

k=0;

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

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

shellSort(arr,n);

k=0;

cout<<endl<<"Po sortowaniu metoda shell 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, oprogramowanie

Opis wygenerowany automatycznie

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

Złożoność obliczeniowa algorytmu Shell Sort zależy od odstępów (gaps), które są używane do sortowania. Złożoność algorytmu nie jest jednoznacznie określona, ponieważ może być dostosowywana poprzez wybór różnych sekwencji odstępów. Jednak ogólnie złożoność Shell Sort wynosi O(nlog^2 n).