**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Algorytmy i struktury danych | **Zadanie** | 10\_02 |
| **Autor** | Kamil Borkowski | **Grupa** | WCY22IY1S1 |
| **Temat** | Counting Sort | | |

1. Treść

Zaimplementuj algorytm sortowania metodą Counting Sort.

* 1. Metoda realizacji

Program wykorzystuje prosty interfejs użytkownika do wyboru operacji, wczytywania danych i sortowania za pomocą algorytmu Counting 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. Znalezienie maksimum.
3. Zliczanie elementów w tablicy pomocniczej i sumowanie kumulacyjne.
4. Przygotowanie tablicy wynikowej i umieszczanie elementów na właściwych pozycjach.
5. Przepisanie posortowanych elementów do oryginalnej tablicy.

Kod źródłowy

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

//Zadanie 10\_02 jezyk C++

//Counting Sort

//WCY22IY1S1 Borkowski Kamil

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

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <cstdio>

#include <fstream>

using namespace std;

//funkcja count sort

void countSort(int arr[], int n) {

int output[10000];

int pom[10000];

int max = arr[0];

for (int i = 1; i < n; i++) {//szukanie najwiekszego elementu tablicy

if (arr[i] > max)

max = arr[i];

}

for (int i = 0; i <= max; ++i) {//wyzerowanie tablicy pomocniczej

pom[i] = 0;

}

for (int i = 0; i < n; i++) {//zliczenie elementow arr w tablicy pomocniczej

pom[arr[i]]++;

}

for (int i = 1; i <= max; i++) {

pom[i] += pom[i - 1];

}

//szukanie indexu elementu z arr w tablicy pomocniczej i umieszczanie elementu w tablicy output

for (int i = n - 1; i >= 0; i--) {//

output[pom[arr[i]] - 1] = arr[i];

pom[arr[i]]--;

}

for (int i = 0; i < n; i++) {//przepisanie elementow z output do arr

arr[i] = output[i];

}

}

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 Counting 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 counting sort: "<<endl;;

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

countSort(arr,n);

k=0;

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

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

countSort(arr,n);

k=0;

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

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

countSort(arr,n);

k=0;

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

Opis wygenerowany automatycznie

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

Złożoność obliczeniowa algorytmu Counting Sort w analizie asymptotycznej wynosi O(n + k), gdzie n to liczba elementów w tablicy, a k to zakres wartości. W praktyce, gdy zakres wartości jest stosunkowo mały w porównaniu z liczbą elementów, algorytm ten może być bardzo efektywny. Jednakże, gdy zakres wartości jest dużo większy niż liczba elementów, może to wpłynąć na zużycie pamięci.