**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Algorytmy i struktury danych | **Zadanie** | 10\_04 |
| **Autor** | Kamil Borkowski | **Grupa** | WCY22IY1S1 |
| **Temat** | Radix Sort | | |

1. Treść

Zaimplementuj algorytm sortowania metodą Radix Sort.

* 1. Metoda realizacji

Program wykorzystuje prosty interfejs użytkownika do wyboru operacji, wczytywania danych i sortowania za pomocą algorytmu Radix 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. Inicjalizacja zmiennej max jako pierwszy element tablicy.
3. Inicjalizacja tablicy pomocniczej pom o rozmiarze max.
4. Zerowanie tablicy pom.
5. Pętla po wszystkich elementach tablicy wejściowej (arr):
6. Obliczenie wartości pom[(arr[i] / poz) % 10]++.
7. Pętla wstecz po tablicy pom:
8. Sumowanie wartości do poprzednich elementów: pom[i] += pom[i - 1].
9. Pętla wstecz po tablicy wejściowej:
10. Umieszczenie elementów w posortowanej kolejności w tablicy wyjściowej (output).
11. Zmniejszenie odpowiedniego elementu w tablicy pom.
12. Funkcja radixsort:
13. Znalezienie maksymalnej liczby w tablicy (max).
14. Pętla po kolejnych cyfrach (dziesiątkach, setkach, itd.) zaczynając od najmniej znaczącej:
15. Wywołanie funkcji countingSort dla danej pozycji.

Kod źródłowy

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

//Zadanie 10\_04 jezyk C++

//Radix Sort

//WCY22IY1S1 Borkowski Kamil

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

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <cstdio>

#include <fstream>

using namespace std;

//funkcja counting sort z uzyciem int poz

void countingSort(int arr[], int n, int poz) {

int max = arr[0];

int output[n];

int pom[max];

for (int i = 0; i < max; ++i)//zerowanie tablicy pom

pom[i] = 0;

for (int i = 0; i < n; i++)//obliczanie tablicy pom od i

pom[(arr[i] / poz) % 10]++;

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

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

for (int i = n - 1; i >= 0; i--) {//umieszczenie elementow w posortowanej kolejnosci

output[pom[(arr[i] / poz) % 10] - 1] = arr[i];

pom[(arr[i] / poz) % 10]--;

}

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

arr[i] = output[i];

}

//funkcja radix sort

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

int max = arr[0];

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

{

if (arr[i] > max)max = arr[i];

}

for (int poz = 1; max / poz > 0; poz \*= 10)//uzycie counting sort

countingSort(arr, n, poz);

}

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 Radix 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 radix sort: "<<endl;;

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

radixsort(arr,n);

k=0;

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

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

radixsort(arr,n);

k=0;

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

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

{

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

k++;

if(k==10)

{

k=0;

cout<<endl;

}

}

radixsort(arr,n);

k=0;

cout<<endl<<"Po sortowaniu metoda radix 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, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

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

Złożoność obliczeniowa jest równa O(d(n+k)) gdzie k to liczba różnych cyfr, a d liczba cyfr w kluczach.