**Міністерство освіти і науки України**

**Національний університет "Львівська політехніка"**

**Кафедра ЕОМ**

****

**Звіт з лабораторної роботи №3**

**з дисципліни “Паралельні та розподілені обчислення ”**

**Варіант 17**

**Виконав: студент .гр. КІ-33**

**Скрипник Д.О.**

**Прийняв: асистент**

**Козак Н.Б.**

**Львів 2020 р.**

**Мета:** Дослідити можливості розв’язання різноманітних задач за допомогою паралельних алгоритмів. Навчитися виділяти незалежні гілки обчислень та виконувати їх паралельно.

**Завдання:**

Ввести п’ять попарно різних цілих чисел a,b,c,d,e. Впорядкувати їх по зростанню, використовуючи не більше 7 порівнянь. Запропонувати узагальнений алгоритм сортування таких послідовностей, зберігаючи пропорцію кількості порівнянь.

**Розв’язок**

Для виконання даної лабораторної роботи мною було використано алгоритм сортування RadixSort – порозрядне сортування, швидкодія О(nlogn), як у інших тестуючих алгоритмів – HeapSort, MergeSort, InsertSort with BinarySearch, але йому достатньо лише найбільше число з заданої послідовності, тому порівнянь завжди буде 4.

**Код програми:**

// A function to do counting sort of arr[] according to

// the digit represented by exp.

void countSort(vector<int> &arr, int n, int exp)

{

int \*output = new int[n]; // output array

int i, count[10] = { 0 };

// Store count of occurrences in count[]

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

count[(arr[i] / exp) % 10]++;

// Change count[i] so that count[i] now contains actual

// position of this digit in output[]

for (i = 1; i < 10; i++)

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

// Build the output array

for (i = n - 1; i >= 0; i--)

{

output[count[(arr[i] / exp) % 10] - 1] = arr[i];

count[(arr[i] / exp) % 10]--;

}

// Copy the output array to arr[], so that arr[] now

// contains sorted numbers according to current digit

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

arr[i] = output[i];

}

// The main function to that sorts arr[] of size n using

// Radix Sort

void radixsort(vector<int> &arr, int &comp)

{

// Find the maximum number to know number of digits

int m = \*max\_element(arr.begin(), arr.end());

comp = arr.size() - 1;

// Do counting sort for every digit. Note that instead

// of passing digit number, exp is passed. exp is 10^i

// where i is current digit number

for (int exp = 1; m / exp > 0; exp \*= 10)

countSort(arr, arr.size(), exp);

}

void radixsort(vector<int>& arr)

{

// Find the maximum number to know number of digits

int m = \*max\_element(arr.begin(), arr.end());

// Do counting sort for every digit. Note that instead

// of passing digit number, exp is passed. exp is 10^i

// where i is current digit number

for (int exp = 1; m / exp > 0; exp \*= 10)

countSort(arr, arr.size(), exp);

}

void testcaseRadixSort()

{

vector <int> saved;

vector <int> a;

set<int> s;

int n = 5, i, t, comparison;

for (int j = 0; j < 100; ++j)

{

a.clear();

s.clear();

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

{

do

{

t = rand() % 100;

} while (s.find(t) != s.end());

s.insert(t);

a.push\_back(t);

}

//cout << "Generated: ";

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

// cout << a[i] << " ";

//cout << endl;

saved = a;

comparison = 0;

radixsort(a, comparison);

testcaseOutput(saved, a, comparison);

}

}

int main()

{

srand(time(NULL));

vector <int> a;

set<int> s;

int n = 5, i, t, sel;

reenter:

cout << "1 - Handwritted" << endl <<

"2 - Random generated" << endl <<

"6 - testcase with radix sort" << endl;

cout << "Choice: ";

cin >> sel;

if (sel == 1)

{

start:

a.clear();

s.clear();

cout << "Input " << n << " diff numbers: ";

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

{

cout << i + 1 << " number: ";

cin >> t;

if (s.find(t) == s.end())

{

a.push\_back(t);

s.insert(t);

}

else

{

cout << "Number was previosly input!" << endl;

break;

}

}

if (i != n)

goto start;

}

else if (sel == 2)

{

a.clear();

s.clear();

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

{

do

{

t = rand() % 100;

} while (s.find(t) != s.end());

s.insert(t);

a.push\_back(t);

}

cout << "Generated: ";

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

cout << a[i] << " ";

cout << endl;

}

else if (sel == 6)

{

testcaseRadixSort();

return 0;

}

else

{

goto reenter;

}

radixsort(a);

cout << "Sorted array:" << endl;

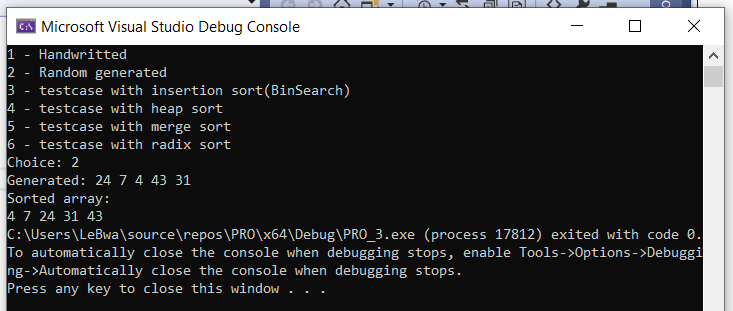
for (i = 0; i < n; i++)

cout << a[i] << " ";

return 0;

}

**Результат виконання програми:**



**Висновок:** На цій лабораторній роботі я дослідив можливості розв’язання різноманітних задач за допомогою паралельних алгоритмів та навчився виділяти незалежні гілки обчислень та виконувати їх паралельно.