**Національний університет „Львівська політехніка”**

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №3

з дисципліни: “ Паралельні та розподілені обчислення ”

Варіант 9

Виконав: ст.гр. КІ-33

Лобай Р.І

Перевірив:

Козак Н.Б

Львів 2020

**Мета роботи:** дослідити можливості розв’язання різноманітних задач за

допомогою паралельних алгоритмів. Навчитися виділяти незалежні гілки обчислень та виконувати їх паралельно.

**Завдання:**

Задати натуральні числа . В послідовності  вибрати підпослідовність  таку, що . Якщо таку підпослідовність вибрати неможливо, то слід вивести відповідне повідомлення.

(При вирішенні даної задачі слід використати наступне міркування. Для того, щоб вибрати необхідну підпослідовність для кожного елемента вхідної послідовності слід визначити чи приймається він у шукану підпослідовність. Може виникнути наступна ситуація: відносно членів  прийняті які-небудь рішення, після чого виявилось, що, як би ми не розпоряджались іншими  членами, нам все одно не вдасться отримати підпослідовність, що задовольняє поставленій умові (наприклад, якщо сума декількох додатних чисел більше m, то неможливо додати до них ще декілька додатних чисел, так, щоб сума стала рівна m). В цьому випадку слід виключити з розгляду всі підпослідовності, перші елементи яких вибрані з  , згідно з прийнятими рішеннями.)

**Аналіз задачі:** в даній задачі нам потрібно знайти послідовність яка задовільняє умову, в стандартному випадку ми йдемо від початку масив і шукаємо послідовність. Дослідивши задачу, я зрозумів що нам не важливо де буде знаходитися ця послідовність, просто нам потрібно швидше її знайти, і для цієї задачі я реалізував паралельний пошук двома методами, перший метод – ми шукаємо послідовність з початку масиву, а в другому методі з кінця, і перший метод який знайде послідовність відправляє сигнал другому методу про закінчення пошуку, і виводиться результат.

//Код програми

using System;

namespace lab3

{

class Program

{

static int sum = 0;

static int sum\_sort = 0;

static int first = 0;

static int last = 0;

static int first\_sort = 0;

static int last\_sort = 0;

static int count\_time = 0;

static int count\_time\_sort = 0;

static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.Unicode;

Random rand = new Random();

int m, n = 10;

int[] arr = new int [n];

//Завовнюєм послідовність рандомними числами

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

{

arr[i] = rand.Next(1, 10);

}

Console.WriteLine("Enter m: ");

string M = Console.ReadLine();

m = Convert.ToInt32(M);

//Виводимо послідовність

printArray(arr,n);

//Виконуємо стандартний пошук послідовності

Console.WriteLine("\nВиконаня пошуку послідовності звичайним способом)");

int sumArray = SortArray(arr,n,m);

if (sumArray == 0)

{

Console.WriteLine("Такої послідовності не існує!!!");

}else

{

//Вивовимо послідовність яка в сумі дає m

printSortArray(arr);

Console.WriteLine("Кількість операцій для стандартного способу пошуку послідовності = " + count\_time);

}

//Виконання пошуку послідовності на двох незалежних потоках

Console.WriteLine("\nВиконаня пошуку послідовності оптимізованим способом)");

int sumArraySort = SortArray2(arr, n, m);

if (sumArraySort == 0)

{

Console.WriteLine("Такої послідовності не існує!!!");

}

else

{

//Вивовимо послідовність яка в сумі дає

int count = 0 ;

if (count\_time >= count\_time\_sort)

{

printSortArray2(arr);

count = count\_time\_sort;

}

else

{

printSortArray(arr);

count = count\_time;

}

Console.WriteLine("Кількість операцій для покращеного способу пошуку послідовності = " + count);

}

Console.ReadKey();

}

//Функція виводу послідовності

static void printArray(int []arr, int n)

{

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

{

Console.Write(arr[i] + " ");

}

Console.WriteLine();

}

//Функція виводу вибраної послідовності

static void printSortArray(int[] arr)

{

for (int i = first; i < last; i++)

{

Console.Write(arr[i] + " ");

}

Console.WriteLine();

}

//Функція виводу вибраної послідовності2

static void printSortArray2(int[] arr)

{

for (int i = first\_sort+1; i < last\_sort+1; i++)

{

Console.Write(arr[i] + " ");

}

Console.WriteLine();

}

//Функція пошуку послідовності для стандартного методу

static int SortArray(int []arr,int n,int m)

{

int k = 0;

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

{

first = k;

for (int j = k ; j < n; j++) {

if (sum <= m)

{

if (sum == m)

{

last = j;

return sum;

}

else

{

count\_time++;

sum += arr[j];

}

}

else {

count\_time++;

sum = 0;

break;

}

}

k++;

}

first = 0;last = 0;sum = 0;count\_time = 0;

return sum;

}

//Функція пошуку послідовності для покращеного методу

static int SortArray2(int[] arr, int n, int m)

{

int k = n-1;

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

{

last\_sort = k;

for (int j = k; j >= 0; j--)

{

if (sum\_sort <= m)

{

if (sum\_sort == m)

{

first\_sort = j;

return sum\_sort;

}

else

{

count\_time\_sort++;

sum\_sort += arr[j];

}

}

else

{

count\_time\_sort++;

sum\_sort = 0;

break;

}

}

k--;

}

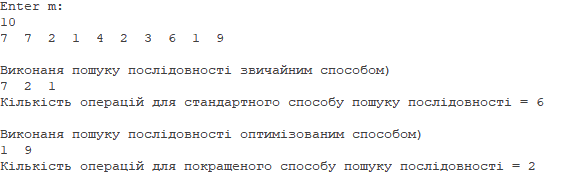
first\_sort = 0; last\_sort = 0; sum\_sort = 0; count\_time\_sort = 0;

return sum;

}

}

}



**Висновки:** на цій лабораторній роботі я застосував паралельне представлення алгоритмів. Створена мною програма використовує такий спосіб паралелізму, як розбиття задачі на підзадачі, а також демонструє можливість використання паралельних обчислень.