МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра програмної інженерії та інформаційних технологій управління

Звіт з лабораторної роботи № 9
з дисципліни «Основи теорії алгоритмів»

Виконав:
ст. гр. КН-221в
Шулюпов Є.Р.
Перевірила:
доцент каф. ПІТУ
Солонська С.В.

Харків
2022

ТЕМА: ЖАДІБНІ АЛГОРИТМИ

ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

Розв’язати задачу про вибір найбільшої кількості заявок для аудиторії. Вхідні дані: кількість заявок N (1 < N < 256), N пар натуральних чисел - початок та кінець заявок s_i, f_i. Вихідні дані: заявки, відсортовані за зростанням часу закінчення, та номери заявок, які потрібно обрати.

МЕТА РОБОТИ

Навчання використовувати жадібні алгоритми та оцінка їхньої складності.

1 ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Жа́дібний алгори́тм — простий і прямолінійний евристичний алгоритм, який приймає найкраще рішення, виходячи з наявних на кожному етапі даних, не зважаючи на можливі наслідки, сподіваючись урешті-решт отримати оптимальний розв'язок. Легкий в реалізації і часто дуже ефективний за часом виконання. Багато задач не можуть бути розв'язані за його допомогою. Наприклад, використання жадібної стратегії для задачі комівояжера породжує такий алгоритм: «На кожному етапі вибирати найближче з невідвіданих міст». Придатний набір варіантів — такий, що обіцяє не просто отримання розв'язку, а отримання оптимального розв'язку задачі.

На відміну від динамічного програмування, за якого задача розв'язується знизу догори, за жадібної стратегії це робиться згори донизу, шляхом здійснення одного жадібного вибору за іншим, зведенням великої задачі до малої. Жадібний алгоритм добре розв'язує деякі задачі, а інші — ні. Більшість задач, для яких він спрацьовує добре, мають дві властивості: по-перше, до них можливо застосувати *принцип жадібного вибору*, по-друге, вони мають властивість *оптимальної підструктури*.

До оптимізаційної задачі можна застосувати принцип жадібного вибору, якщо послідовність локально оптимальних виборів дає глобально оптимальний розв'язок. В типовому випадку доведення оптимальності здійснюється за такою схемою: спочатку доводиться, що жадібний вибір на першому етапі не унеможливлює шляху до оптимального розв'язку: для будь-якого розв'язку є інший, узгоджений із жадібним і не гірший від першого. Далі доводиться, що підзадача, яка виникла після жадібного вибору на першому етапі, аналогічна початковій, і міркування закінчується за індукцією. Інакше кажучи, за жадібного алгоритму ніколи не переглядаються попередні вибори для здійснення наступного, на відміну від динамічного програмування.

«Задача має оптимальну підструктуру, якщо оптимальний розв'язок задачі містить оптимальний розв'язок для підзадач»[2]. Інакше кажучи, задача має оптимальну підструктуру, якщо кожен наступний крок веде до оптимального розв'язку. Прикладом «неоптимальної підструктури» може бути ситуація в шахах, коли взяття ферзя (хороший наступний крок) веде до програшу партії в цілому.

Жадібні алгоритми можна охарактеризувати як «короткозорі» і «невідновлювані». Вони ідеальні лише для задач з «оптимальною підструктурою». Попри це, жадібні алгоритми найкраще підходять для простих задач. Для багатьох інших задач жадібні алгоритми зазнають невдачі у продукуванні оптимального розв'язку, і можуть навіть видати найгірший з можливих розв'язків. Один з прикладів — алгоритм найближчого сусіднього міста, згаданий вище.

Прикладом застосування цього алгоритму є задача про вибір заявок. Задача заключається в тому, що на конференції, щоб відвести більше часу на неформальне спілкування, різні секції рознесли по різних аудиторіях. Вчений із надзвичайно широкими інтересами бажає відвідати декілька доповідей у різних секціях. Відомі початок і кінець кожної доповіді. Визначити, яку найбільшу кількість доповідей можна відвідати.

Наведемо жадібний алгоритм, який розв'язує цю задачу. При цьому вважатимемо, що заявки впорядковано за зростанням часу закінчення. Якщо це не так, то можна відсортувати їх за час; заявки з однаковим часом закінчення розташовуємо в довільному порядку.

На вхід алгоритму подаються масиви початків і закінчень доповідей. Множина A складається з номерів вибраних заявок, а j — номер наступної заявки. Жадібний алгоритм шукає заявку, що починається не раніше від закінчення j-ї, потім знайдену заявку включає в A, а j присвоює її номер. Алгоритм працює за O, тобто за складністю сортування, оскільки вибірка має меншу складність. На кожному кроці вибирається найкращий розв'язок. Покажемо, що результатом буде оптимальний розв'язок.

2 ОПИСАННЯ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

Програма реалізована в одному файлі source.cpp

```
#include <iostream>
#include<algorithm>

using namespace std;


struct Activitiy
{
    int start, finish;
};


bool activityCompare(Activitiy s1, Activitiy s2)
{
    if (s1.finish != s2.finish)
        return (s1.finish < s2.finish);
    else
        return(s1.start < s2.start);
}


void printMaxActivities(Activitiy arr[], int n)
{
    sort(arr, arr + n, activityCompare);
    cout << "Відсортована послідовність за часом закінчення: " << endl;
    for (int i = 0; i < n; i++)
    {
        cout << "(" << arr[i].start << ", " << arr[i].finish << ")" << endl;
    }
    cout << "Найбільша кількість заявок: \n";

    int i = 0;
    cout << "(" << arr[i].start << ", " << arr[i].finish << ")   ";

    for (int j = 1; j < n; j++)
    {

        if (arr[j].start >= arr[i].finish)
        {
            cout << "(" << arr[j].start << ", " << arr[j].finish << ")   ";
            i = j;
```

```
        }
    }
}

int main()
{
    setlocale(LC_ALL, "Ukr");
    int n;
    cout << "Введіть кількість заявок" << endl;
    cin >> n;
    if (n < 0)
    {
        cout << "Неправильно введені дані" << endl;
        exit(0);
    }
    Activitiy arr[256];
    for (int i = 0; i < n; i++)
    {
        cout << "Введіть початок заявки" << endl;
        cin >> arr[i].start;
        cout << "Введіть кінець заявки" << endl;
        cin >> arr[i].finish;
        if (arr[i].start > arr[i].finish || arr[i].start < 0|| arr[i].finish < 0)
        {
            cout << "Неправильно введені дані" << endl;
            exit(0);
        }
    }
    printMaxActivities(arr, n);
    return 0;
       }
```

Результат роботи представлений на рисунку 2.1.

```
Введіть кількість заявок
5
Введіть початок заявки
1 6
Введіть кінець заявки
Введіть початок заявки
2
Введіть кінець заявки
3
Введіть початок заявки
4 5
Введіть кінець заявки
Введіть початок заявки
5 8
Введіть кінець заявки
Введіть початок заявки
7
Введіть кінець заявки
9
Відсортована послідовність за часом закінчення:
(2, 3)
(4, 5)
(1, 6)
(5, 8)
(7, 9)
Найбільша кількість заявок:
(2, 3)   (4, 5)   (5, 8)
```

Рисунок 2.1 – Результат

Крім того була оцінена складність алгоритму, що зображено в таблиці 2.2.

№ рядка	Алгоритм	Кількість операцій (вартість)	Скільки разів буде виконаний рядок
1	int i = 0;	c_1	1
2	for (int j = 1; j < n; j++)	c_2	n+1
3	if (arr[j].start >= arr[i].finish)	c_3	nlog(n)
4	cout << "(" << arr[j].start << ", " << arr[j].finish << ") ";	c_4	<=nlog(n)
5	i = j;	c_5	<=nlog(n)

Поєднавши члени всіх рядків, отримаємо:

$T(m) = c_1 + c_2(n+1) + c_3 n\log(n) . + c_4 n\log(n) . + c_5 n\log(n)$.

Перетворимо даний вираз, відкинувши члени менших порядків і коефіцієнти при n. Тоді отримаємо $T(n) = O(n\log(n))$.

ВИСНОВКИ

Під час цієї лабораторної роботи була розроблена програма, яка виконує жадібний алгоритм згідно варіанту завдання, а саме будування жадібного алгоритму та вибору найбільшої кількості заявок, що мають початкове та кінцеве положення, для того, щоб оцінити можливості задачі. Дана програма виконує всі необхідні функції, які описані у завданні до даної лабораторної роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Методичні вказівки до виконання лабораторних робіт з курсу "Алгоритми і структури даних": для студентів, які навчаються за спец. 121 "Інженерія програмного забезпечення" [Електронний ресурс] / уклад. Н. К. Стратієнко, І. О. Бородіна ; Харківський політехнічний інститут, національний технічний університет університет –

Електрон. текстові дані. – Харків, 2017. – 36 с. – Режим доступу: http://repository.kpi.kharkov.ua/handle/KhPI-Press/26426.

2. Алгоритми і структури даних: практикум: навч. посіб./ Н.К. Стратієнко, М.Д. Годлевський, І.О. Бородіна.- Харьков: НТУ"ХПИ", 2017. - 224 с.