

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
дисциплины «Алгоритмизация»

Выполнила:
Кубанова Ксения Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А.

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: жадные алгоритмы

Цель: освоить и построить жадные алгоритмы согласно предоставленным на лекции.

Порядок выполнения работы

Задание 1.

Вход: множество n точек на прямой x_1, \dots, x_n .

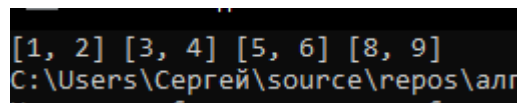
Выход: минимальное кол-во отрезков единичной длины, которыми можно покрыть все точки.

Ниже представлен алгоритм для выполнения покрытия точек отрезками (алгоритм 1).

```
Функция POINTSCOVER( $x_1, \dots, x_n$ )  
→  $x_1, \dots, x_n \leftarrow \text{SORT}(x_1, \dots, x_n)$   
   $i \leftarrow 1$   
  пока  $i \leq n$ :  
    добавить к решению отрезок  $[\ell, r] = [x_i, x_i + 1]$   
     $i \leftarrow i + 1$   
    пока  $i \leq n$  и  $x_i \leq r$ :  
       $i \leftarrow i + 1$   
  вернуть построенное решение
```

Рисунок 1 – алгоритм 1

Реализация этого алгоритма представлена в файле *алгб_1.cpp*, а на рисунке 2 представлен результат выполнения этого алгоритма.



```
[1, 2] [3, 4] [5, 6] [8, 9]  
C:\Users\Сергей\source\repos\алг
```

Рисунок 2 – итог работы алгоритма 1

Время работы данного алгоритма: $O(n^2)$.

Задание 2.

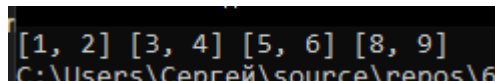
Алгоритм, представленный ниже, является улучшенным алгоритму 1, который был реализован в 1 задании. Он имеет более оптимальное решение.

Функция POINTSCOVER(x_1, \dots, x_n)

```
 $S \leftarrow \{x_1, \dots, x_n\}$   
пока  $S$  не пусто:  
     $x_m \leftarrow$  минимальная точка  $S$   
    добавить к решению отрезок  $[\ell, r] = [x_m, x_m + 1]$   
    выкинуть из  $S$  точки, покрытые отрезком  $[\ell, r]$   
вернуть построенное решение
```

Рисунок 3 – алгоритм 2

Реализация алгоритма представлена в файле *6_2.cpp*, а его итог на рисунке 3.



```
[1, 2] [3, 4] [5, 6] [8, 9]  
C:\Users\Сергей\source\repos\6_2.cpp
```

Рисунок 4 – итог алгоритма 2

Время работы улучшенного алгоритма: $T(\text{sort}) + O(n) = O(n \log n)$.

Задание 3.

Далее будет реализована задача о выборе заявок.

Ввод: множество n отрезков на прямой

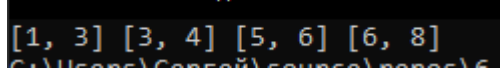
Вывод: максимальное кол-во попарно не пересекающихся отрезков.

Функция ACTSEL($\ell_1, r_1, \dots, \ell_n, r_n$)

```
 $S \leftarrow \{[\ell_1, r_1], \dots, [\ell_n, r_n]\}$   
пока  $S$  не пусто:  
     $[\ell_m, r_m] \leftarrow$  отрезок из  $S$  с мин. правым концом  
    добавить  $[\ell_m, r_m]$  к решению  
    выкинуть из  $S$  отрезки, пересекающиеся с  $[\ell_m, r_m]$   
вернуть построенное решение
```

Рисунок 5 – алгоритм 3

Решение алгоритма представлено в файле *6_3.cpp*, его результат на рисунке 6:



```
[1, 3] [3, 4] [5, 6] [6, 8]
```

Рисунок 6 – итог алгоритма 3

Время работы данного алгоритма: $O(n^2)$.

Задание 4.

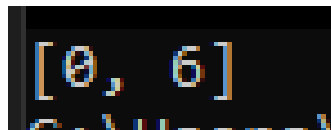
Алгоритм, представленный ниже, является улучшенным алгоритму 3, который был реализован в 3 задании. Он имеет более оптимальное решение.

Функция $ACTSEL(\ell_1, r_1, \dots, \ell_n, r_n)$

отсортировать n отрезков по правым концам
для всех отрезков в полученном порядке:
 если текущий отрезок не пересекает
 последний добавленный:
 взять его в решение
вернуть построенное[†] решение

Рисунок 7 – алгоритм 4

Решение алгоритма представлено в файле `6_4.cpp`, а его вывод на рисунке 8:



```
[0, 6]
```

Рисунок 8 – итог алгоритма 4

Время работы улучшенного алгоритма: $T(\text{sort}) + O(n) = O(n \log n)$.

Задание 5.

Задача под названием «Планирование вечеринки в компании» имеет следующие составляющие:

Вход: дерево

Выход: независимое множество (множество не соединённых друг с другом вершин) максимального размера.

Алгоритм работы задачи представлен ниже:

Функция MAXINDEPENDENTSET(T)

```
пока  $T$  не пусто:  
    взять в решение все листья  
    выкинуть их и их родителей из  $T$   
вернуть построенное решение
```



Рисунок 9 – алгоритм 5

Его реализация организована в файле *6_5.cpp* и имеет следующий результат:

```
11 10 9 8 5 3  
10 9 8 5 3  
9 8 5 3  
8 5 3  
7 5 3  
6 5 3  
5 3  
4 3  
3 1  
2 1
```

Рисунок 10 – итог алгоритма 5

Время работы алгоритма составляет $O(|T|)$.

Задание 6.

Задача под названием «*Непрерывный рюкзак*» имеет следующие составляющие:

Вход: веса w_1, \dots, w_n и стоимости c_1, \dots, c_n данных n предметов, вместимость рюкзака w .

Выход: максимальная стоимость частей предметов суммарного веса не более w .

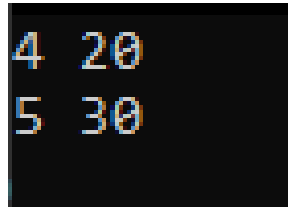
Алгоритм работы задачи «*Непрерывный рюкзак*» представлена ниже:

Функция $\text{KNAPSACK}(w_1, c_1, \dots, w_n, c_n)$

отсортировать предметы по возрастанию c/w
для всех предметов в полученном порядке:
взять по максимуму текущего предмета
вернуть построенное решение

Рисунок 11 – алгоритм 6

Реализация алгоритма задачи «Непрерывный рюкзак» представлена в файле `6_6.cpp` и имеет следующий вывод:



```
4 20
5 30
```

Рисунок 12 – итог алгоритма 6

Время работы данного алгоритма: $T(\text{sort}) + O(n) = O(n \log n)$.

Вывод: в ходе лабораторной работы были изучены и реализованы жадные алгоритмы согласно предоставленным на лекции.