БУ ВО Ханты-Мансийского автономного округа – Югры

«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

ОТЧЁТ

по лабораторной работе №10

по дисциплине: «Алгоритмы и структуры данных»

Выполнил: студент(ка) группы №609-31,

Гаврилов Е.Е.

Принял: старший преподаватель кафедры АиКС

Назаров Е.В.

Сургут

2025г.

**Цель работы:** изучить принципы построения алгоритмов с использованием градиентного метода («жадные алгоритмы»), закрепить навыки определять асимптотическую сложность алгоритмов, решать задачи алгоритмизации и программирования.

**Задание:**

1. Построить детальный алгоритм решения одного варианта задания и реализовать его.
2. Доказать, что алгоритм находит оптимальное решение. Определить, существует ли «жадный» алгоритм решения выбранного варианта.
3. Определить асимптотическую сложность алгоритма и доказать ее корректность.
4. Если алгоритм не приводит к оптимальному решению, найти пример (используя конкретные значения), который демонстрирует это.

**Вариант 4:**

Спорт. Известно, что оптимальным выбором лыж является такой, когда длина лыж максимально приближенна к высоте спортсмена. Команда состоит из n спортсменом, чей рост составляет h1, h2, … , hn. На команду выдан комплект n пар лыж длиной l1, l2, …, ln. Необходимо реализовать алгоритм выдачи лыж спортсменам, который бы обеспечивал минимум суммы абсолютной разности между ростом спортсмена и парой лыж, доставшейся ему.

**Ход работы:**

«Жадный» алгоритм решения задачи:

1. Положить, что последовательности H и L не отсортированы.

2. Пройтись по всем h, для каждого h пройтись по всем l.

3. Для каждой h определить такую l, что |h-l| минимальна. При нахождении такого l, что h-l = 0, можно прекратить поиск по lj и перейти к следующему h.

4. Когда такое l будет найдено, записать пару {h,l}, найденное l убрать из L, далее оно использоваться не будет.

5. В итоге получится n пар {hi, lj}, сумма модулей разностей который будет стремиться к минимальной из возможных.

На каждой итерации алгоритм будет выбирать локально самый лучший вариант, поэтому он будет являться «жадным»

Время работы алгоритма: O(n2), поскольку проверяются все сочетания n спортсменов и n пар лыж. (более точно - O( n2/2), так как после проверки очередного h, один l будет удален из множества L).

Листинг 1 (решение «жадным» способом):  
int solution(const vector<int> &heights, const vector<int> &\_lengths,

vector<Pair> &sol) {

sol.clear();

vector<int> lengths(\_lengths.begin(), \_lengths.end());

for(auto h: heights){

int mx = -1;

for(int i = 0; i< lengths.size(); i++){

if(lengths[i]!=-1 && (mx==-1 || (abs(lengths[mx] - h) > abs(lengths[i] - h))) ){

mx = i;

if(lengths[i]-h == 0) break;

}

}

sol.push\_back(Pair{h,lengths[mx]});

lengths[mx] = -1;

}

int res = 0;

for (auto i : sol) {

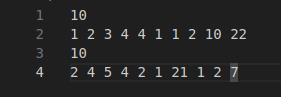
res += abs(i.higth - i.lenght);

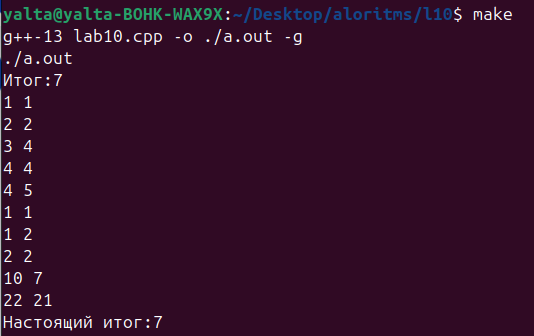
}

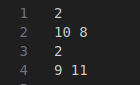
return res;

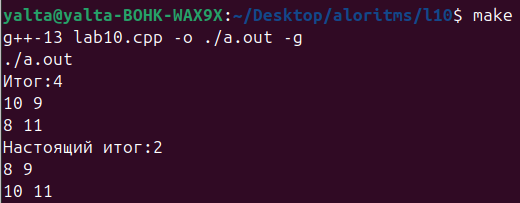
}

Однако «жадность» данного алгоритма не позволяет ему быть оптимальным. Например для последовательностей H={10,8}, L={9,11}, образуются пары {10,9} и {8, 11} что даст сумму 4, но в действительности минимальной будет сумма 2, для пар {10,11} и {9,8}. Не оптимальное решение было получено из-за «жадности» алгоритма. Пары {10,9} и {10,11} одинаковы по разности, но 9 встретилась в последовательности раньше чем 11 и было удалено.

Рис. 1: Пример ввода

Рис. 2: Пример вывода (решение совпадает с потимальным)

Рис. 3: Пример ввода

Рис. 4: Пример вывода (решение не совпадает с оптимальным)

Оптимальным решением будет отсортировать оба множества H и L и составить пары {hi, li}. Можно показать, что в таком случае, при любой перестановке двух элементов из отсортированного L сделают итоговую сумму не меньше чем для отсортированных H и L.

Время работы такого алгоритма: O(n log n) (n log n для быстрой сортировки и n для прохода по n парам).

Листинг 2 (решение оптимальным способом):

int realSolution(vector<int> &heights, vector<int> &lengths,

vector<Pair> &sol) {

if (heights.size() != lengths.size()) {

return -1;

}

sol.clear();

sort(heights.begin(), heights.end());

sort(lengths.begin(), lengths.end());

int res = 0;

for (int i = 0; i < lengths.size(); i++) {

res += abs(heights[i] - lengths[i]);

sol.push\_back(Pair{heights[i], lengths[i]});

}

return res;

}

**Вывод:** в ходе работы изучены принципы построения алгоритмов с использованием градиентного метода.