Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

им. В.И. Ульянова (Ленина)»

кафедра систем автоматизированного проектирования

**ОТЧЕТ**

**по лабораторной работе**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: «Жадные алгоритмы»**

Выполнил : Ахметзянов Дамир Альбертович

Группа № 1301

Преподаватель: Родионова Е. А.

Санкт-Петербург

2022

Оглавление

[Постановка задачи](#_heading=h.30j0zll) **3**

[**Теоретическая асимптотическая сложность**](#_heading=h.26mwkgdbv5zx) **3**

[**Оценка временной сложности эмпирическим способом**](#_heading=h.gn9j51y2nby8) **5**

[**Пример работы программы**](#_heading=h.o1n9phw2k3za) **5**

[Листинг](#_heading=h.2et92p0) **5**

# Постановка задачи

Решить задачу с использованием принципа «Разделяй и властвуй».

Определить теоретическую асимптотическую сложность решения.

Эмпирически оценить временную сложность решения для «average-case».Описание реализуемого класса и методов и оценка временной сложности.

Задача: О влюбленных улитках. На квадрате земли 1 км 2 располагаются улитки-

гермафродиты. В момент времени каждая из улиток с постоянной

скоростью 1 cм/с ползет к улитке, являющейся ближайшей к ней в момент

времени t = 0, выбрав её в качестве спутника жизни. Определить время,

через которое первая пара улиток достигнет друг друга или наличие

ситуации, приводящей улиток в замешательство, наиболее эффективным

способом

# Теоретическая асимптотическая сложность

Если применить стратегию “разделяй и властвуй”, использованная в сортировке слиянием: мы находим ближайшую пару среди точек в “левой половине” P и ближайшую пару среди точек в “правой половине” P, после чего эта информация используется для получения общего решения за линейное время. Если разработать алгоритм с такой структурой, то решение базового рекуррентного отношения обеспечит время выполнения O(n\*log n). Докажем это.

T(n) = aT(n/b) + f(n), где  
n = объем входных данных  
a = количество подзадач в рекурсии  
n/b = размер каждой подзадачи. Предполагается, что все подзадачи имеют одинаковый размер.  
f(n) = оценка выполненной работы вне рекурсивных вызовов.

T(n) имеет следующие асимптотические оценки:

Если f(n) = O(nlogb a-ϵ), то T(n) = Θ(nlogb a).

Если f(n) = Θ(nlogb a), то T(n) = Θ(nlogb a\*log n).

Если f(n) = Ω(nlogb a+ϵ), то T(n) = Θ(f(n)).

ϵ > 0 — константа.

Каждое из этих условий можно интерпретировать следующим образом:

1)Если стоимость решения подзадач на каждом уровне увеличивается на некий коэффициент, то значение f(n) станет полиномиально меньше, чем nlogb a. То есть, временная сложность зависит от стоимости последнего уровня — nlogb a.

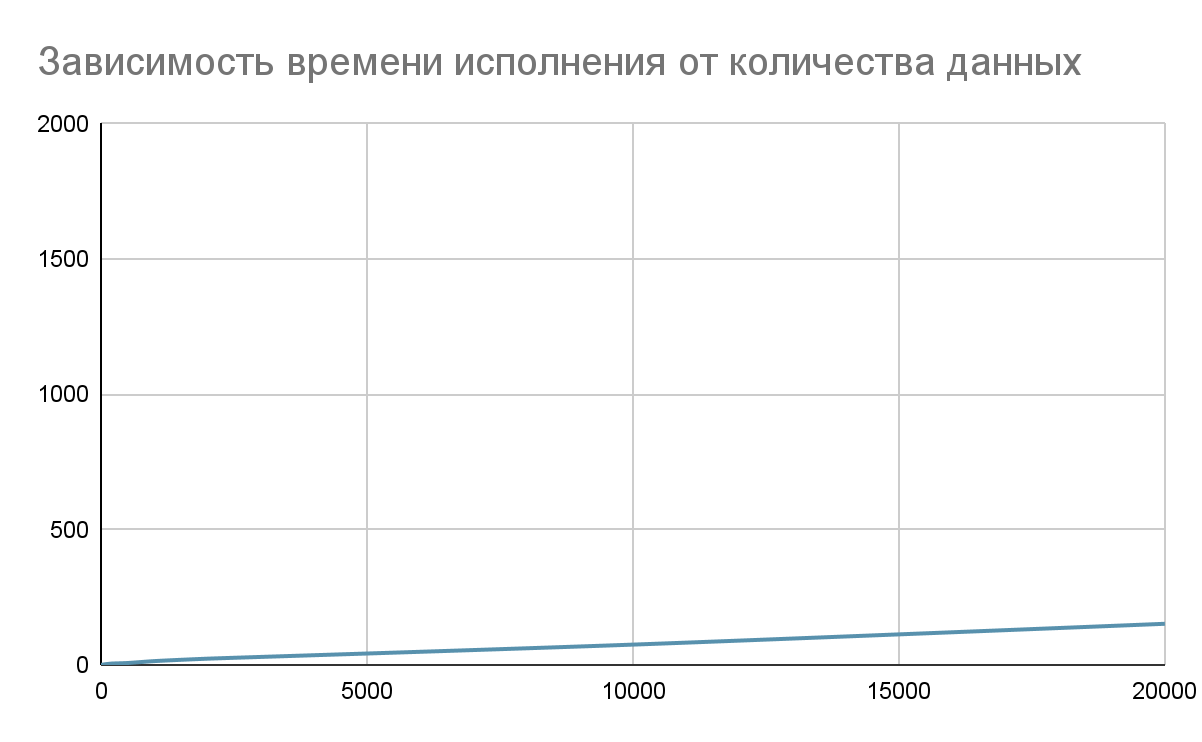
2)Если стоимость решения подзадач на каждом уровне примерно одинакова, то значение f(n) станет равно n^logb a. То есть, временная сложность будет равна f(n), умноженной на количество уровней — nlogb a\*log n.

3)Если стоимость решения подзадач на каждом уровне уменьшается на некий коэффициент, то значение f(n) станет полиномиально больше, чем n^logb a. То есть, временная сложность зависит от стоимости f(n).

В задаче как раз 2ой случай. a=2, n/b=n/2. Поэтому сложность данного алгоритма O(n\*log n).

# Оценка временной сложности эмпирическим способом

Для оценки временной сложности построим график время работы программы(мс) от количества входных данных.



# 

# Пример работы программы

На вход программы дается файл с координатами точек.

Программа выводит пару точек которые ближе всего друг к другу и расстояние между ними, а так же время работы.

Вывод программы:



# Листинг

https://github.com/DamirAhm/AADS/tree/master/Greedy/Greedy