Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информатики

Отчет по практическому заданию на тему

**Задача о паре ближайших точек**

Выполнили:

студенты гр. 153503

Звягинцева В.А.

Киселёва Е. А.

Филипеня А. Д.

Проверила:

Калугина М. А.

Минск 2023

Постановка проблемы:

Для заданных *n* точек на плоскости найти две, расположенные ближе всего друг к другу.

**Ограничение:**

Любые две точки не могут иметь одинаковых ординат или абсцисс.

**Метрика:**

*2*

**Алгоритм полного перебора**

Пара ближайших точек может быть вычислена за время путем выполнения полного перебора. Чтобы это сделать, можно вычислить расстояние между всеми парами точек, затем выбрать пару с наименьшим расстоянием, как показано ниже (The Brute force solution).

Листинг функции:

double bruteForce(std::vector<Point>::iterator points, size\_t sz, std::function<double(const Point&, const Point&)> distance) {

double min = FLT\_MAX;

for (size\_t i = 0; i < sz; ++i) {

for (size\_t j = i + 1; j < sz; ++j) {

min = std::min(min, distance(points[i], points[j]));

}

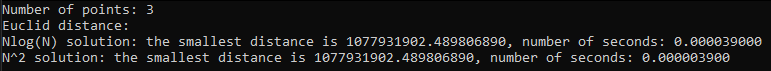
}

return min;

}

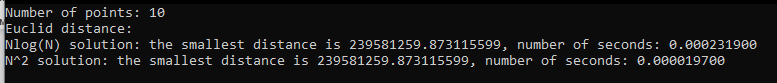
Эффективнее всего данный метод покажет себя на количестве точек .

Количество точек 3:

****

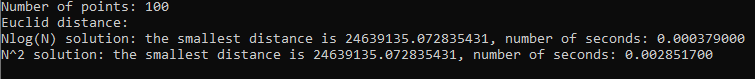
**Алгоритм «Разделяй и властвуй»**

1. Количество точек 10:



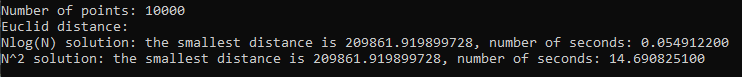
Разница времени выполнения алгоритмов: 0.0002122 (алгоритм полного перебора еще работает быстрее)

1. Количество точек 100:



Разница времени выполнения алгоритмов: 0.0024727 (алгоритм полного перебора работает медленнее)

1. Количество точек 10000:



Для 10000 точек количество времени, требующегося на алгоритм полного перебора на 14 с превосходит алгоритм «Разделяй и властвуй».

**Оценка временной сложности алгоритма:**

*merge\_sort()* – сортирует список сортировкой слиянием, которая имеет сложность времени выполнения O(nlogn).

*closestDistance ()* – рекурсивная функция, которая делит список точек на две половины, рекурсивно находит ближайшую пару точек в каждой половине, а затем объединяет результаты, чтобы найти ближайшую пару точек в целом.

*distanceEuclide ()* – вычисляет расстояние между двумя точками с помощью теоремы Пифагора, которая имеет сложность времени выполнения O(1).

*distanceManhattan()*– вычисляет расстояние между двумя точками по метрике расстояния городских кварталов с помощью сложения модулей чисел, которое имеет сложность времени выполнения O(1).

**Реккурентное соотношение**:

Таким образом, общая сложность выполнения алгоритма такова:

T(n) = 2T(n/2) + O(n).

Следовательно, сложность времени выполнения алгоритма составляет O(nlogn).

**Вывод:**

По представленным выше скриншотам выполнения работы программы, можно увидеть, что увеличение количества числа точек до довольно больших чисел (в нашем случае до 10 тысяч точек) увеличивает время работы программы, пусть и не значительно, но явно заметно (мы использовали встроенные функции языка C++ для подсчёта процессорного времени). В случае алгоритма полного перебора значения получаются и того слишком большими. Следовательно, чем большее количество точек будет подаваться в качестве входных данных, тем медленнее будет работать программа. В случае с небольшим количеством точек (в нашем случае от 10 до 100 точек) программа будет совершать работу за незначительное время.

В зависимости от выбранного языка программирования может измениться скорость выполнения алгоритма. В случае данного практического задания, алгоритмы «Разделяй и властвуй» и полного перебора были реализованы на двух языках программирования: C++ и Python.

Для сравнения представляем результат процессорного времени на C++ для тысячи точек:



Результат процессорного времени на Python для тысячи точек:

Также помимо подсчетов в Евклидовой метрике, были проведены расчеты в метрике расстояния городских кварталов.

Для сравнения представляем результат на C++ для тысячи точек:

