

Содержание

1	Фурмолировка задачи	2
2	Условия упрощения алгоритма	2
2.1	Фурмалировка условий упрощения алгоритма	2
2.2	Уточним первое условие	2
2.3	Уточним второе условие	3
2.4	Уточним третье условие	3
3	Входные данные	3
3.1	Область обработки входных данных	3
3.2	обработка входных данных	4

1 Фурмолировка задачи

2 Условия упрощения алгоритма

2.1 Фурмалировка условий упрощения алгоритма

Так как мы пишем программу, что бы облегчить себе задачу, будем искать куда поставить почтаamt методом переюора. Но перебор длжен быть организован так, что:

1. Точек куда можно поставить почтаamt конечно, и их количество должно быть $a < n < b$ ($10^5 < n < 10^6$).

2. Для каждой точки можно определить скалярную функцию входных данных(центры активности(далее ЦА), зоны запрета полета(далее NFZ), населенности в районе).

3. Входные данные должны быть осмысленны.

2.2 Уточним первое условие

Теперь определим a , b . Переменная a отвечает за то что бы не упрастить задачу до одной точки. Я думаю a должно быть таким, что расстояние между точками меньше 100метров. Площадь Маската 3500 km^2

откуда получаем $a \approx \frac{35 \cdot 10^8}{100^2} = 35 \cdot 10^4$, а если учесть что $1/3$ маската это малонаселенные горы получаем $a \approx 10^5$. Значение b можно определить из времени выполнения программы. Дадим напрмер на алгорим 10 минут. количество операций которое можно провести за это время можн опонять из такой программы:

```
1 import datetime
2 t = datetime.datetime.now()
3 i = 0
4 while (datetime.datetime.now() - t).seconds < 30:
5     i += 1
6 print(i * 20)
```

Вывод программы зависит от устройства на котором она запущена. У меня выдала 792801540 или примерно $8000 \cdot 10^5$ и если на одну точку

брать хотя бы 800 простых операций получаем $b \approx 10^6$
Итак первое условие $10^5 < n < 10^6$

2.3 Уточним второе условие

Для начала определим еще один важный фактор. Наша программа расставляет почтамты последовательно и от наиболее загруженного к наименее загруженному, отсюда получаем ситуацию: стоит почтамт и не так далеко находится ЦА. Мы конечно хотим поставить почтамт рядом с ЦА, но тут уже стоит один. Отсюда следствие: функция должна учитывать так же уже поставленные почтамты.

2.1 функция должна опираться на уже поставленные почтамты

2.4 Уточним третье условие

Проблема в том, что точных данных по населенности у нас нет. Поэтому мы пришли к упрощению: Маскат распределен на регионы, где население распределено равномерно. Если внутри региона видно сильное разделение, то мы искусственно разделим этот регион на "подрегионы" и поделим население так, как нам покажется правильным, что бы население в "подрегионах" было \pm равномерно.

То-есть все данные должны быть проработаны на правдивость и поэтому результат программы должен быть так же осознан (должен подвергнуться критике со стороны человека)

3 Входные данные

3.1 Область обработки входных данных

Как мы понимаем, люди не будут использовать почтамат если он находится в 10km от их дома. Так же очевидно, что чем дальше почтамт от их дома, тем меньше они будут им пользоваться. Я бы взял расстояние до 2km и функцию от расстояния $f(R) \sim \frac{1}{R^2}$ или даже $f(R) \sim \frac{1}{R^3}$.

Так же с ЦА. люди из ЦА пойдут к почтамту только если он очень близко, поскольку в бизнес центре или складе время-деньги, а в магазинах люди веселятся так что не готовы идти далеко в этом и смысл торговых центров. Отсюда радиус $< 500m$ а $f(R) \sim \frac{1}{R^3}$.

NFZ будут обрабатываться просто как точки, где поставить почтаматы нельзя.

3.2 обработка входных данных

На вход алгоритм получит точки(определяющие регионы), населенность регионов, точки(определяющие ЦА), рейтинг(рейтинг ЦА который создан субъективно, но пропорционален количеству посылок, отправляемых из ЦА).

Мы хотим их привести к данным, которые проще обрабатывать. Поэтому превратим регионы в набор точек каждой из которых будет присвоено значение населенности.

Проблема: Регионы - не всегда выпуклые многоугольники значит определить принадлежит ли точка многоугольнику нельзя просто проверив принадлежит ли точка всем полуплоскостям образованными гранями.

Решение: Изменим многоугольник вырезая вершины так, что он станет выпуклым. Получим