

СОДЕРЖАНИЕ

Введение	3
1 Аналитический раздел	4
1.1 Задача коммивояжёра	4
1.2 Методы ее решения	4
1.3 Муравьиные алгоритмы	5
1.4 Вывод	6
2 Конструкторский раздел	7
2.1 Схемы алгоритмов	7
2.2 Параметризация и трудоёмкость муравьиного алгоритма	9
2.3 Вывод	10
3 Технологический раздел	11
3.1 Средства реализации	11
3.2 Листинг кода	11
3.3 Собираемые данные	15
3.4 Вывод	15
4 Исследовательский раздел	16
4.1 Анализ результатов	16
Заключение	18
Список использованных источников	19
Приложение А	20

ВВЕДЕНИЕ

Цель работы: провести сравнительный анализ метода полного перебора и эвристического метода на базе муравьиного алгоритма.

При выполнении лабораторной работы поставлены такие задачи:

- 1) реализовать метод полного перебора и метод на базе муравьиного алгоритма для решения задачи коммивояжёра с возвращением последнего в город, с которого он начал обход;
- 2) провести параметризацию второго метода для выбранного класса задач;
- 3) сделать выводы о результатах параметризации.

1 Аналитический раздел

В данном разделе будет представлено понятие задачи коммивояжера, рассмотрены алгоритм полного перебора и муравьиный алгоритм как способы ее решения.

1.1 Задача коммивояжера

Цель задачи коммивояжера заключается в нахождении самого выгодного маршрута (кратчайшего, самого быстрого, наиболее дешевого), проходящего через все заданные точки (пункты, города) по одному разу, с последующим возвратом в исходную точку [1].

Условия задачи должны содержать критерий выгодности маршрута (т. е. должен ли он быть максимально коротким, быстрым, дешевым или все вместе), а также исходные данные в виде матрицы затрат (расстояния, стоимости, времени и т. д.) при перемещении между рассматриваемыми пунктами.

Особенности задачи в том, что она довольно просто формулируется и найти хорошие решения для нее также относительно просто, но вместе с тем поиск действительно оптимального маршрута для большого набора данных - непростой и ресурсоемкий процесс.

Для решения задачи коммивояжера ее надо представить как математическую модель. При этом исходные условия можно записать в формате матрицы - таблицы, где строкам соответствуют города отправления, столбцам - города прибытия, а в ячейках указываются расстояния (время, стоимость) между ними; или в виде графа - схемы, состоящей из вершин, которые символизируют города, и соединяющих их ребер, длина которых соответствует расстоянию между городами.

1.2 Методы ее решения

Полный перебор - заключается в последовательном рассмотрении всех возможных маршрутов и выборе самого оптимального из них. Он является самым простым методом, который при этом всегда даёт верный ответ.

Идея муравьиного алгоритма – моделирование поведения муравьёв, связанное с их способностью быстро находить кратчайший путь от муравейника к источнику пищи и адаптироваться к изменяющимся условиям, находя новый кратчайший путь. При своём движении муравей метит свой путь феромоном, и эта информация используется другими муравьями для выбора пути[2].

Моделирование муравьёв связано с распределением феромона на тропе – ребре графа в задаче коммивояжёра. При этом вероятность включения ребра в маршрут отдельного муравья пропорциональна количеству феромона.

1.3 Муравьиные алгоритмы

Для решения задачи коммивояжера можно описать локальные правила поведения муравьев при выборе пути.

— Муравьи имеют собственную «память». Поскольку каждый город может быть посещен только 1 раз, у каждого муравья есть список уже посещенных городов – список запретов. Обозначим через $J_{i,k}$ список городов, которые необходимо посетить муравью k , находящемуся в городе i ;

— муравьи обладают «зрением» – видимость есть эвристическое желание посетить город j , если муравей находится в городе i . Будем считать что видимость обратно пропорциональна расстоянию между городами i и j – D_{ij}

$$\eta_{ij} = 1/D_{ij} \quad (1.1)$$

— муравьи обладают «обонянием» – они могут улавливать след феромона, подтверждающий желание посетить город j из города i , на основании опыта других муравьёв. Количество феромона на ребре (i,j) в момент времени t обозначим через $\tau_{ij}(t)$.

Таким образом вероятностно-пропорциональное правило, определяющее вероятность перехода k -ого муравья из города i в город j :

$$\begin{cases} P_{ij,k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} [\eta_{ij}]^{\beta}}{\sum_{l \in J_{i,k}} [\tau_{il}(t)]^{\alpha} [\eta_{il}]^{\beta}}, j \in J_{i,k} \\ P_{ij,k} = 0, j \notin J_{i,k} \end{cases} \quad (1.2)$$

где α, β – параметры, задающие веса следа феромона, при $\alpha = 0$ алгоритм вырождается до жадного.

Пройдя ребро (i, j) муравей откладывает на нем некоторое количество феромона, которое должно быть связано с оптимальностью сделанного выбора. Пусть $T_k(t)$ есть маршрут, пройденный муравьём k к моменту времени t , а $L_k(t)$ – длина этого маршрута. Пусть также Q – параметр, имеющий значение порядка длины оптимального пути. Тогда откладываемое количество феромона может быть задано в виде

$$\delta\tau_{ij,k}(t) = \begin{cases} \frac{Q}{L_k(t)}, & (i, j) \in T_k(t); \\ 0, & (i, j) \notin T_k(t). \end{cases} \quad (1.3)$$

Правила внешней среды определяют, в первую очередь, испарение феромона. Пусть $p \in [0, 1]$ есть коэффициент испарения, тогда правильно испарения имеет вид

$$\tau_{ij}(t+1) = (1-p) * \tau_{ij}(t) + \delta\tau_{ij}(t); \quad \delta\tau_{ij}(t) = \sum_{k=1}^m \delta\tau_{ij,k}(t) \quad (1.4)$$

где m – количество муравьев в колонии.

1.4 Вывод

В данном разделе была представлена задача коммивояжёра, рассмотрены методы ее решения.

2 Конструкторский раздел

В данном разделе будут рассмотрены схемы алгоритма полного перебора и муравьиного алгоритма для решения задачи коммивояжёра, а также трудоёмкость и параметризация последнего.

2.1 Схемы алгоритмов

На рисунках 2.1 и 2.2 представлены схемы алгоритма полного перебора и муравьиного алгоритма.

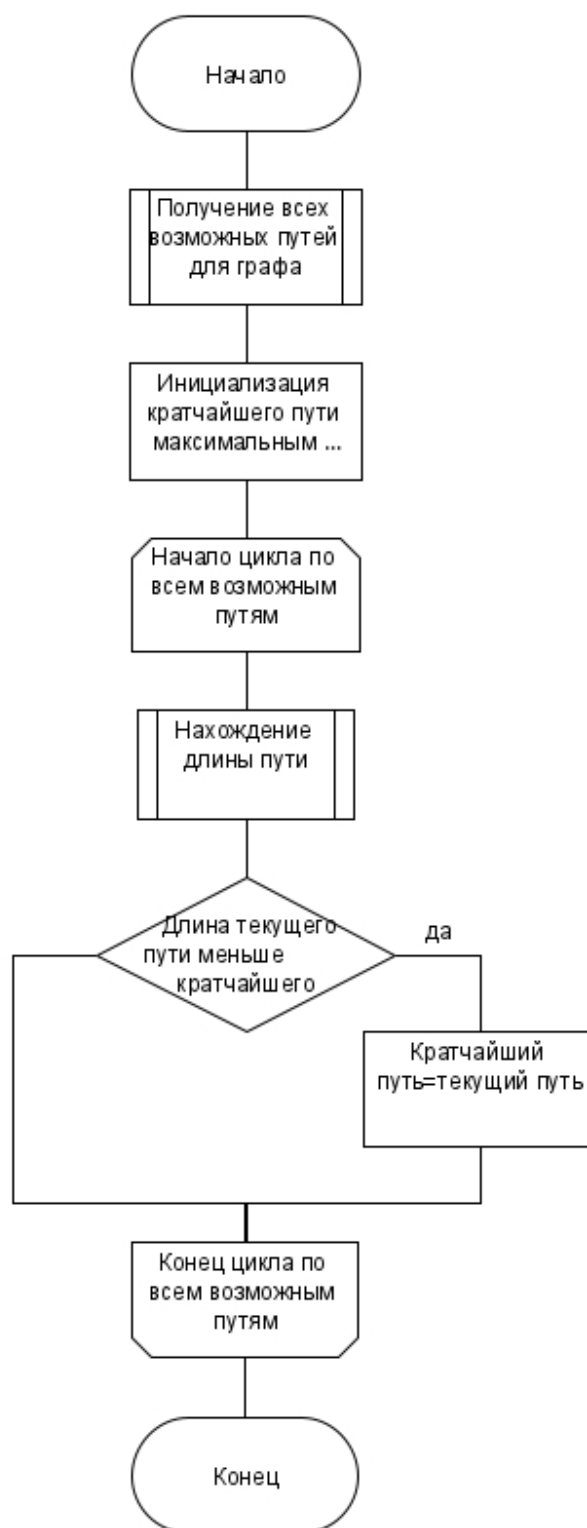


Рисунок 2.1 — Алгоритм полного перебора

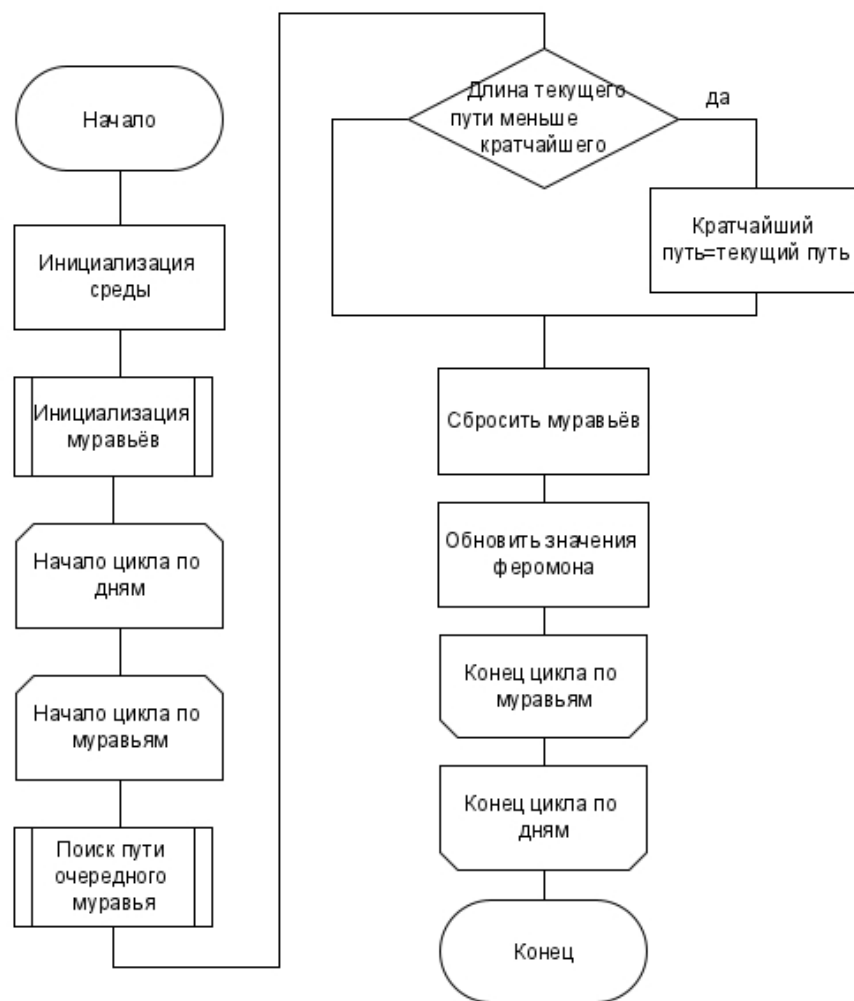


Рисунок 2.2 — Муравьиный алгоритм

2.2 Параметризация и трудоёмкость муравьиного алгоритма

На эффективность алгоритма влияют параметры: коэффициенты жадности и стадности, коэффициент испарения феромона, количество поколений.

Для получения их оптимальной комбинации производится параметризация – полный перебор в цикле этих параметров. Для каждой их комбинации производится 2 поиска маршрута для каждой из 2-х карт, из этих двух маршрутов выбирается по минимальному и она записывается в файл вместе с разностями длин полученных маршрутов и эталонного.

Трудоёмкость его $O(t_{max} * \max(m, n^2))$.

2.3 Вывод

В данном разделе были рассмотрены схемы алгоритма полного перебора и муравьиного алгоритма, описана параметризация последнего.

3 Технологический раздел

В данном разделе будут представлены листинги кода реализованных алгоритмов и дано описание получаемых результатов параметризации программы.

3.1 Средства реализации

В данной работе используется язык программирования Python. Среда разработки Visual Studio Code. Для замера процессорного времени используется функция `process_time()` из библиотеки `time`.

Замеры времени были произведены на: Intel(R) Core(TM) i5-8250U CPU @1.60GHz 1.80 Ghz, 4 ядра, 8 логических процессоров.

3.2 Листинг кода

В листинге 3.1 приведен код алгоритма полного перебора.

Листинг 3.1 — Алгоритм полного перебора

```
1 class Graph():
2     def __init__(self, _n, _c=None):
3         self.nodes = _n
4         self.c = _c
5
6     def ets(self):
7         start = [i for i in range(1, self.nodes)]
8         min_len = float("inf")
9         min_rout = [0 for _ in range(self.nodes)]
10        routs = permutations(start)
11        for rout in routs:
12            rout = list(rout)
13            rout.insert(0, 0)
14            rout.append(0)
15            cur_len = self.count_cost(rout)
16            if cur_len < min_len:
17                min_len = cur_len
18                min_rout = rout
19        return min_rout
```

В листинге 3.2 представлен код муравьиного алгоритма.

Листинг 3.2 — Муравьиный алгоритм

```
1 class Graph():
2     def __init__(self, _n, _c=None):
3         self.nodes = _n
4         self.c = _c
5     def ant_solve(self, herd, greed, evaportion, tmax):
6         pheromones = [[0.1 for _ in range(self.nodes)] for _ in
7             range(self.nodes)]
8         visibility = [[1 / self.c[j][i] for j in range(self.nodes)] for i
9             in range(self.nodes)]
10        q = self.calculate_q()
11
12        min_len = 0
13        for i in range(1, self.nodes):
14            min_len += self.c[i-1][i]
15            min_len += self.c[self.nodes-1][0]
16        min_rout = [i for i in range(self.nodes)]
17        min_rout.append(0)
18
19        possibilities = [i for i in range(self.nodes)]
20        ants = [Ant(i, possibilities, herd, greed) for i in range(self.nodes)]
21
22        for i in range(tmax):
23            delta_pher = 0
24            for i in range(self.nodes):
25                ants[i].find_way(pheromones, visibility)
26                l = self.count_cost(ants[i].rout)
27                ph = q / l
28                delta_pher += ph
29
30                if l < min_len:
31                    min_len = l
32                    min_rout = [ants[i].rout[j] for j in
33                        range(len(ants[i].rout))]
34
35                ants[i].reset(possibilities)
36
37            for i in range(self.nodes):
38                for j in range(self.nodes):
39                    if i != j:
40                        pheromones[i][j] *= (1 - evaportion)
41                        pheromones[i][j] += delta_pher
42
43        return min_rout
```

В листинге 3.3 представлен код класса муравья.

Листинг 3.3—Класс муравья

```
1 class Ant():
2     def __init__(self, start, possibilities, _herd, _greed):
3         self.possibilities = [possibilities[i] for i in
4                               range(len(possibilities))]
5         self.possibilities.remove(start)
6         self.base_pos = start
7         self.pos = start
8         self.herd = _herd
9         self.greed = _greed
10        self.rout = []
11        self.rout.append(start)
12
13    def find_way(self, pheromones, visibility):
14        while 1:
15            chances = []
16            zn = 0
17            for city in self.possibilities:
18                t = pow(pheromones[self.pos][city], self.herd) *
19                    pow(visibility[self.pos][city], self.greed)
20                zn += t
21            chances.append([city, t])
22            for i in range(len(self.possibilities)):
23                chances[i][1] = chances[i][1] / zn
24
25            city = self.make_choice(chances)
26            self.pos = city
27            self.possibilities.remove(city)
28            self.rout.append(city)
29            if len(self.possibilities) == 0:
30                break
31
32        self.rout.append(self.base_pos)
33
34    def make_choice(self, chances):
35        chances_sum = 0
36        for chance in chances:
37            chances_sum += chance[1]
38        choice = uniform(0, chances_sum)
39        next_city = 0
40
41        t = 0
42        while choice > 0:
43            choice -= chances[t][1]
44            t += 1
45        next_city = chances[t-1][0]
```

```

44         return next_city
45
46     def reset(self, possibilities):
47         self.possibilities = [possibilities[i] for i in
                                range(len(possibilities))]
48         self.possibilities.remove(self.base_pos)
49         self.pos = self.base_pos
50         self.rout = [self.base_pos]

```

В листинге 3.4 представлен код процесса параметризации.

Листинг 3.4 — Параметризация

```

1  def parametrization():
2      g = Graph(10, matr1)
3      g2 = Graph(10, matr3)
4
5      reference1 = g.ets()
6      reference_l1 = g.count_cost(reference1)
7
8      reference2 = g2.ets()
9      reference_l2 = g2.count_cost(reference2)
10
11     f = open("parametrisation.txt", 'w')
12     for herd in range(0, 10, 2):
13         for greed in range(0, 10, 2):
14             evaporation = 0.1
15             while evaporation < 1:
16                 for days in range(10, 51, 10):
17                     f.write("{:2d} {:.1f} {:2d} ".format(herd, evaporation,
18                                                         days))
19
20                     rout1 = g.ant_solve(herd, greed, evaporation, days)
21                     rout2 = g.ant_solve(herd, greed, evaporation, days)
22                     l11 = g.count_cost(rout1)
23                     l12 = g.count_cost(rout2)
24                     t11 = min(l11, l12)
25                     l1 = reference_l1 - t11
26
27                     rout1 = g2.ant_solve(herd, greed, evaporation, days)
28                     rout2 = g2.ant_solve(herd, greed, evaporation, days)
29                     l21 = g2.count_cost(rout1)
30                     l22 = g2.count_cost(rout2)
31                     t12 = min(l21, l22)
32                     l2 = reference_l2 - t12

```

```
33             f.write("{:2d}  {:2d}  {:2d}  {:2d}\n".format(tl1 , tl2 ,  
34                   11 , 12))  
35             evaporation += 0.2
```

3.3 Собираемые данные

В результате параметризации формируется файл каждая строка которого – комбинация параметров и полученные длины маршрутов и разность их и эталонных.

3.4 Вывод.

В данном разделе были рассмотрены листинги кода реализованных алгоритмов и параметризации, описаны результаты работы программы.

4 Исследовательский раздел

В данном разделе будут приведены результаты работы программы и сделаны выводы.

4.1 Анализ результатов

Замеры времени проводились при количестве городов от 3 до 10, с шагом 1, которые заполнялись случайным образом, для каждой матрицы проводилось 50 испытаний, муравьиный алгоритм выполнялся с параметрами $\alpha = 4, \beta = 6, p = 0.5, t_{max} = 20$. На графике 4.1.

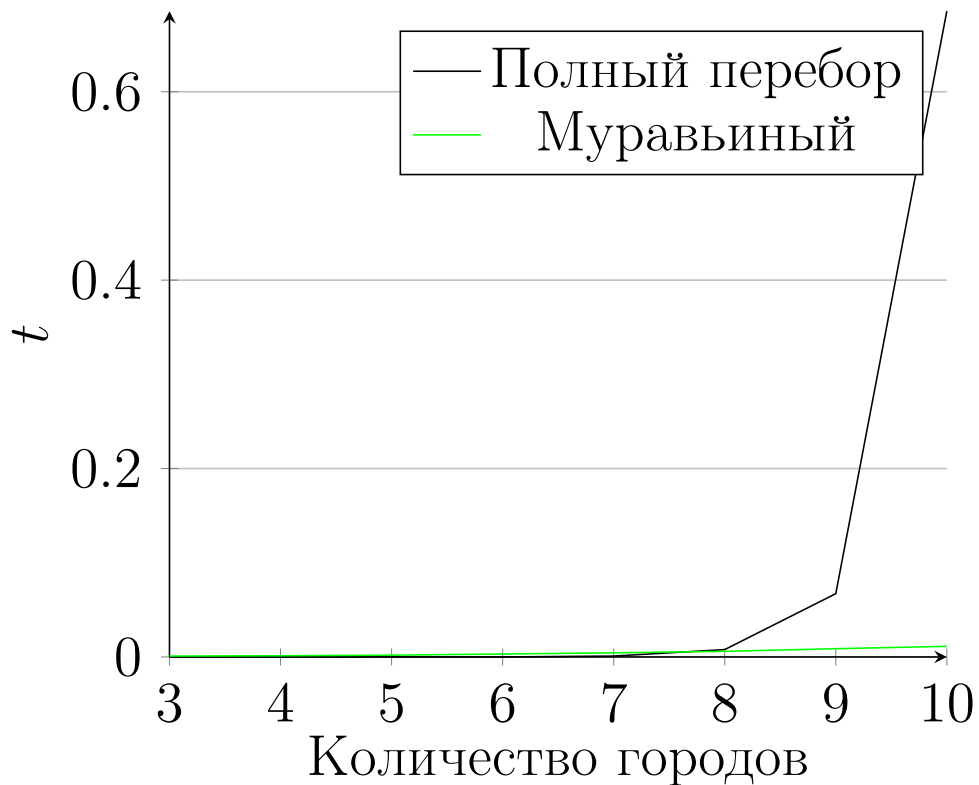


Рисунок 4.1 — Времене на поиск пути

Из графиков видно, что муравьиный алгоритм показывает себя хуже при небольших размерах матриц, однако в связи с тем, что трудоёмкость алгоритма полного перебора $O(n!)$ уже при 9 городах он в несколько раз медленнее.

В таблице 4.1 представлены некоторые строки с лучшими результатами, где α -коэффициент стадности, p -испарения, t_{max} -количество дней, d -расстояние, rd -разность длин. Эталонные длины: 58 и 55.

α	p	t_{max}	d	rd
0	0.3-0.9	20-50	58	0
4	0.1-0.9	30-50	58	0
6	0.5-0.9	20-50	58	0

Таблица 4.1 — Результаты параметризации

Из таблицы видно, что при вырождении до жадного алгоритма, или параметрах стадности 4 и 6, а также коэффициенте испарения 0.5-0.9 с количеством итераций 40 в среднем, даёт наилучшие результаты. Полные данные можно увидеть в таблице А.1.

Вывод

В данном разделе были рассмотрены результаты работы программы, стало ясно, что при количестве городов больше 9 имеет смысл использовать муравьиный алгоритм для решения задачи коммивояжёра при условии правильно подобранных параметров.

ЗАКЛЮЧЕНИЕ

В результате лабораторной работы цель была достигнута – проведён сравнительный анализ метода полного перебора и эвристического метода на базе муравьиного алгоритма.

Стало ясно, что на количестве городов больше 9 муравьиный алгоритм при условии правильно подобранных параметров, например коэффициента стадности 4-6, коэффициенте испарения 0.5 или 0.7 и 30 итераций будет давать результат близкий к эталонному, за время в несколько раз меньшее чем полный перебор.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Р.Р. Галяутдинов. Задача коммивояжера-метод ветвей и границ[Электронный ресурс]. — 2013. — Режим доступа: <http://galyautdinov.ru/post/zadacha-kommivoyazhera> (дата обращения: 03.12.2020).
2. Ульянов М.В. Ресурсо-эффективные компьютерные алгоритмы. Разработка и анализ. — М. : Наука, 2007. — 376 с.

ПРИЛОЖЕНИЕ А

Таблица А.1 — Результат параметризации

α	p	tmax	d1	d2	dr1	dr2
0	0.1	10	60	57	-2	-2
0	0.1	20	62	58	-4	-3
0	0.1	30	61	58	-3	-3
0	0.1	40	61	59	-3	-4
0	0.1	50	61	57	-3	-2
0	0.3	10	62	59	-4	-4
0	0.3	20	62	58	-4	-3
0	0.3	30	60	58	-2	-3
0	0.3	40	61	57	-3	-2
0	0.3	50	60	58	-2	-3
0	0.5	10	61	59	-3	-4
0	0.5	20	62	57	-4	-2
0	0.5	30	61	58	-3	-3
0	0.5	40	59	59	-1	-4
0	0.5	50	60	57	-2	-2
0	0.7	10	60	59	-2	-4
0	0.7	20	62	58	-4	-3
0	0.7	30	60	58	-2	-3
0	0.7	40	62	57	-4	-2
0	0.7	50	60	58	-2	-3
0	0.9	10	61	59	-3	-4
0	0.9	20	61	57	-3	-2
0	0.9	30	61	56	-3	-1
0	0.9	40	61	58	-3	-3
0	0.9	50	60	58	-2	-3
0	0.1	10	59	56	-1	-1
0	0.1	20	59	57	-1	-2
0	0.1	30	59	56	-1	-1

0	0.1	40	59	56	-1	-1
0	0.1	50	59	57	-1	-2
0	0.3	10	60	57	-2	-2
0	0.3	20	60	56	-2	-1
0	0.3	30	60	55	-2	0
0	0.3	40	59	56	-1	-1
0	0.3	50	59	57	-1	-2
0	0.5	10	59	57	-1	-2
0	0.5	20	59	57	-1	-2
0	0.5	30	59	56	-1	-1
0	0.5	40	58	56	0	-1
0	0.5	50	58	56	0	-1
0	0.7	10	59	57	-1	-2
0	0.7	20	60	56	-2	-1
0	0.7	30	58	57	0	-2
0	0.7	40	59	55	-1	0
0	0.7	50	59	56	-1	-1
0	0.9	10	60	56	-2	-1
0	0.9	20	60	57	-2	-2
0	0.9	30	58	56	0	-1
0	0.9	40	59	56	-1	-1
0	0.9	50	58	55	0	0
0	0.1	10	59	55	-1	0
0	0.1	20	59	55	-1	0
0	0.1	30	59	56	-1	-1
0	0.1	40	59	56	-1	-1
0	0.1	50	58	55	0	0
0	0.3	10	59	56	-1	-1
0	0.3	20	58	56	0	-1
0	0.3	30	58	56	0	-1
0	0.3	40	58	55	0	0

0	0.3	50	58	55	0	0
0	0.5	10	59	55	-1	0
0	0.5	20	59	56	-1	-1
0	0.5	30	58	56	0	-1
0	0.5	40	59	56	-1	-1
0	0.5	50	59	56	-1	-1
0	0.7	10	58	57	0	-2
0	0.7	20	59	56	-1	-1
0	0.7	30	58	56	0	-1
0	0.7	40	58	56	0	-1
0	0.7	50	58	55	0	0
0	0.9	10	60	56	-2	-1
0	0.9	20	59	55	-1	0
0	0.9	30	58	56	0	-1
0	0.9	40	58	55	0	0
0	0.9	50	58	55	0	0
0	0.1	10	58	56	0	-1
0	0.1	20	58	56	0	-1
0	0.1	30	58	56	0	-1
0	0.1	40	58	55	0	0
0	0.1	50	58	55	0	0
0	0.3	10	59	55	-1	0
0	0.3	20	58	55	0	0
0	0.3	30	58	55	0	0
0	0.3	40	58	55	0	0
0	0.3	50	58	55	0	0
0	0.5	10	59	56	-1	-1
0	0.5	20	58	55	0	0
0	0.5	30	58	55	0	0
0	0.5	40	58	55	0	0
0	0.5	50	58	55	0	0

0	0.7	10	58	55	0	0
0	0.7	20	58	55	0	0
0	0.7	30	58	55	0	0
0	0.7	40	58	55	0	0
0	0.7	50	58	55	0	0
0	0.9	10	58	56	0	-1
0	0.9	20	58	55	0	0
0	0.9	30	59	55	-1	0
0	0.9	40	58	55	0	0
0	0.9	50	58	55	0	0
0	0.1	10	58	55	0	0
0	0.1	20	58	55	0	0
0	0.1	30	58	55	0	0
0	0.1	40	58	55	0	0
0	0.1	50	58	55	0	0
0	0.3	10	58	55	0	0
0	0.3	20	58	55	0	0
0	0.3	30	58	55	0	0
0	0.3	40	58	55	0	0
0	0.3	50	58	55	0	0
0	0.5	10	58	55	0	0
0	0.5	20	58	55	0	0
0	0.5	30	58	55	0	0
0	0.5	40	58	55	0	0
0	0.5	50	58	55	0	0
0	0.7	10	58	55	0	0
0	0.7	20	59	56	-1	-1
0	0.7	30	58	55	0	0
0	0.7	40	58	55	0	0
0	0.7	50	58	55	0	0
0	0.9	10	58	55	0	0

0	0.9	20	58	55	0	0
0	0.9	30	58	55	0	0
0	0.9	40	58	55	0	0
0	0.9	50	58	55	0	0
2	0.1	10	62	59	-4	-4
2	0.1	20	60	58	-2	-3
2	0.1	30	60	57	-2	-2
2	0.1	40	60	57	-2	-2
2	0.1	50	59	57	-1	-2
2	0.3	10	62	60	-4	-5
2	0.3	20	59	60	-1	-5
2	0.3	30	60	58	-2	-3
2	0.3	40	60	61	-2	-6
2	0.3	50	60	57	-2	-2
2	0.5	10	61	59	-3	-4
2	0.5	20	60	57	-2	-2
2	0.5	30	61	58	-3	-3
2	0.5	40	60	56	-2	-1
2	0.5	50	60	58	-2	-3
2	0.7	10	60	59	-2	-4
2	0.7	20	61	58	-3	-3
2	0.7	30	60	56	-2	-1
2	0.7	40	61	58	-3	-3
2	0.7	50	61	57	-3	-2
2	0.9	10	61	58	-3	-3
2	0.9	20	61	56	-3	-1
2	0.9	30	60	57	-2	-2
2	0.9	40	60	56	-2	-1
2	0.9	50	59	57	-1	-2
2	0.1	10	59	57	-1	-2
2	0.1	20	59	56	-1	-1

2	0.1	30	59	55	-1	0
2	0.1	40	58	57	0	-2
2	0.1	50	59	55	-1	0
2	0.3	10	61	57	-3	-2
2	0.3	20	59	57	-1	-2
2	0.3	30	60	56	-2	-1
2	0.3	40	58	55	0	0
2	0.3	50	59	56	-1	-1
2	0.5	10	60	57	-2	-2
2	0.5	20	58	56	0	-1
2	0.5	30	60	56	-2	-1
2	0.5	40	59	57	-1	-2
2	0.5	50	60	56	-2	-1
2	0.7	10	60	57	-2	-2
2	0.7	20	59	57	-1	-2
2	0.7	30	59	56	-1	-1
2	0.7	40	59	56	-1	-1
2	0.7	50	59	56	-1	-1
2	0.9	10	61	58	-3	-3
2	0.9	20	59	56	-1	-1
2	0.9	30	60	56	-2	-1
2	0.9	40	59	56	-1	-1
2	0.9	50	59	56	-1	-1
2	0.1	10	60	55	-2	0
2	0.1	20	59	55	-1	0
2	0.1	30	59	55	-1	0
2	0.1	40	58	55	0	0
2	0.1	50	58	55	0	0
2	0.3	10	59	57	-1	-2
2	0.3	20	59	55	-1	0
2	0.3	30	58	56	0	-1

2	0.3	40	58	56	0	-1
2	0.3	50	59	55	-1	0
2	0.5	10	58	55	0	0
2	0.5	20	58	55	0	0
2	0.5	30	59	55	-1	0
2	0.5	40	58	55	0	0
2	0.5	50	58	55	0	0
2	0.7	10	58	56	0	-1
2	0.7	20	59	56	-1	-1
2	0.7	30	59	55	-1	0
2	0.7	40	58	55	0	0
2	0.7	50	58	55	0	0
2	0.9	10	58	55	0	0
2	0.9	20	58	56	0	-1
2	0.9	30	59	56	-1	-1
2	0.9	40	59	55	-1	0
2	0.9	50	59	55	-1	0
2	0.1	10	59	55	-1	0
2	0.1	20	58	56	0	-1
2	0.1	30	58	55	0	0
2	0.1	40	58	55	0	0
2	0.1	50	58	55	0	0
2	0.3	10	59	55	-1	0
2	0.3	20	58	55	0	0
2	0.3	30	58	55	0	0
2	0.3	40	58	55	0	0
2	0.3	50	58	55	0	0
2	0.5	10	59	55	-1	0
2	0.5	20	58	55	0	0
2	0.5	30	58	55	0	0
2	0.5	40	58	55	0	0

2	0.5	50	58	55	0	0
2	0.7	10	58	56	0	-1
2	0.7	20	58	55	0	0
2	0.7	30	58	55	0	0
2	0.7	40	58	55	0	0
2	0.7	50	58	55	0	0
2	0.9	10	59	55	-1	0
2	0.9	20	58	55	0	0
2	0.9	30	58	55	0	0
2	0.9	40	58	55	0	0
2	0.9	50	58	55	0	0
2	0.1	10	58	56	0	-1
2	0.1	20	58	55	0	0
2	0.1	30	58	55	0	0
2	0.1	40	58	55	0	0
2	0.1	50	58	55	0	0
2	0.3	10	58	55	0	0
2	0.3	20	58	55	0	0
2	0.3	30	58	55	0	0
2	0.3	40	58	55	0	0
2	0.3	50	58	55	0	0
2	0.5	10	59	55	-1	0
2	0.5	20	58	55	0	0
2	0.5	30	58	55	0	0
2	0.5	40	58	55	0	0
2	0.5	50	58	55	0	0
2	0.7	10	58	55	0	0
2	0.7	20	58	55	0	0
2	0.7	30	58	55	0	0
2	0.7	40	58	55	0	0
2	0.7	50	58	55	0	0

2	0.9	10	58	55	0	0
2	0.9	20	58	55	0	0
2	0.9	30	58	55	0	0
2	0.9	40	58	55	0	0
2	0.9	50	58	55	0	0
4	0.1	10	61	60	-3	-5
4	0.1	20	61	59	-3	-4
4	0.1	30	59	57	-1	-2
4	0.1	40	60	57	-2	-2
4	0.1	50	60	55	-2	0
4	0.3	10	62	58	-4	-3
4	0.3	20	61	57	-3	-2
4	0.3	30	61	58	-3	-3
4	0.3	40	60	58	-2	-3
4	0.3	50	61	58	-3	-3
4	0.5	10	61	60	-3	-5
4	0.5	20	60	58	-2	-3
4	0.5	30	61	58	-3	-3
4	0.5	40	60	58	-2	-3
4	0.5	50	60	58	-2	-3
4	0.7	10	60	59	-2	-4
4	0.7	20	60	56	-2	-1
4	0.7	30	61	57	-3	-2
4	0.7	40	61	58	-3	-3
4	0.7	50	60	57	-2	-2
4	0.9	10	63	60	-5	-5
4	0.9	20	60	59	-2	-4
4	0.9	30	61	58	-3	-3
4	0.9	40	60	59	-2	-4
4	0.9	50	61	57	-3	-2
4	0.1	10	61	56	-3	-1

4	0.1	20	59	58	-1	-3
4	0.1	30	59	57	-1	-2
4	0.1	40	60	55	-2	0
4	0.1	50	59	56	-1	-1
4	0.3	10	58	57	0	-2
4	0.3	20	59	56	-1	-1
4	0.3	30	60	57	-2	-2
4	0.3	40	58	56	0	-1
4	0.3	50	59	55	-1	0
4	0.5	10	61	56	-3	-1
4	0.5	20	58	56	0	-1
4	0.5	30	60	57	-2	-2
4	0.5	40	59	55	-1	0
4	0.5	50	59	56	-1	-1
4	0.7	10	60	57	-2	-2
4	0.7	20	59	55	-1	0
4	0.7	30	59	56	-1	-1
4	0.7	40	60	56	-2	-1
4	0.7	50	60	56	-2	-1
4	0.9	10	60	56	-2	-1
4	0.9	20	59	55	-1	0
4	0.9	30	58	55	0	0
4	0.9	40	59	55	-1	0
4	0.9	50	58	56	0	-1
4	0.1	10	60	56	-2	-1
4	0.1	20	58	55	0	0
4	0.1	30	58	56	0	-1
4	0.1	40	59	56	-1	-1
4	0.1	50	58	55	0	0
4	0.3	10	59	56	-1	-1
4	0.3	20	58	57	0	-2

4	0.3	30	59	56	-1	-1
4	0.3	40	58	55	0	0
4	0.3	50	58	55	0	0
4	0.5	10	59	56	-1	-1
4	0.5	20	59	55	-1	0
4	0.5	30	58	55	0	0
4	0.5	40	58	55	0	0
4	0.5	50	58	55	0	0
4	0.7	10	59	57	-1	-2
4	0.7	20	58	56	0	-1
4	0.7	30	58	55	0	0
4	0.7	40	58	55	0	0
4	0.7	50	58	55	0	0
4	0.9	10	59	56	-1	-1
4	0.9	20	59	55	-1	0
4	0.9	30	59	55	-1	0
4	0.9	40	59	56	-1	-1
4	0.9	50	58	55	0	0
4	0.1	10	58	55	0	0
4	0.1	20	58	55	0	0
4	0.1	30	58	55	0	0
4	0.1	40	58	55	0	0
4	0.1	50	58	55	0	0
4	0.3	10	58	55	0	0
4	0.3	20	58	55	0	0
4	0.3	30	58	55	0	0
4	0.3	40	58	55	0	0
4	0.3	50	58	55	0	0
4	0.5	10	58	55	0	0
4	0.5	20	58	55	0	0
4	0.5	30	58	55	0	0

4	0.5	40	58	55	0	0
4	0.5	50	58	55	0	0
4	0.7	10	59	56	-1	-1
4	0.7	20	58	55	0	0
4	0.7	30	58	55	0	0
4	0.7	40	58	55	0	0
4	0.7	50	58	55	0	0
4	0.9	10	59	55	-1	0
4	0.9	20	58	56	0	-1
4	0.9	30	58	55	0	0
4	0.9	40	58	55	0	0
4	0.9	50	58	55	0	0
4	0.1	10	59	55	-1	0
4	0.1	20	59	55	-1	0
4	0.1	30	58	55	0	0
4	0.1	40	58	55	0	0
4	0.1	50	58	55	0	0
4	0.3	10	58	55	0	0
4	0.3	20	58	55	0	0
4	0.3	30	58	55	0	0
4	0.3	40	58	55	0	0
4	0.3	50	58	55	0	0
4	0.5	10	58	55	0	0
4	0.5	20	58	55	0	0
4	0.5	30	58	55	0	0
4	0.5	40	58	55	0	0
4	0.5	50	58	55	0	0
4	0.7	10	58	55	0	0
4	0.7	20	58	55	0	0
4	0.7	30	58	55	0	0
4	0.7	40	58	55	0	0

4	0.7	50	58	55	0	0
4	0.9	10	58	55	0	0
4	0.9	20	58	55	0	0
4	0.9	30	58	55	0	0
4	0.9	40	58	55	0	0
4	0.9	50	58	55	0	0
6	0.1	10	63	59	-5	-4
6	0.1	20	61	57	-3	-2
6	0.1	30	61	57	-3	-2
6	0.1	40	61	58	-3	-3
6	0.1	50	61	57	-3	-2
6	0.3	10	62	59	-4	-4
6	0.3	20	60	59	-2	-4
6	0.3	30	62	57	-4	-2
6	0.3	40	61	58	-3	-3
6	0.3	50	60	57	-2	-2
6	0.5	10	63	60	-5	-5
6	0.5	20	60	59	-2	-4
6	0.5	30	62	59	-4	-4
6	0.5	40	60	58	-2	-3
6	0.5	50	60	55	-2	0
6	0.7	10	62	60	-4	-5
6	0.7	20	60	57	-2	-2
6	0.7	30	61	58	-3	-3
6	0.7	40	61	58	-3	-3
6	0.7	50	60	58	-2	-3
6	0.9	10	62	59	-4	-4
6	0.9	20	61	59	-3	-4
6	0.9	30	61	58	-3	-3
6	0.9	40	61	57	-3	-2
6	0.9	50	59	58	-1	-3

6	0.1	10	61	55	-3	0
6	0.1	20	60	57	-2	-2
6	0.1	30	58	57	0	-2
6	0.1	40	60	56	-2	-1
6	0.1	50	58	55	0	0
6	0.3	10	60	56	-2	-1
6	0.3	20	59	56	-1	-1
6	0.3	30	60	56	-2	-1
6	0.3	40	59	56	-1	-1
6	0.3	50	58	56	0	-1
6	0.5	10	59	57	-1	-2
6	0.5	20	59	56	-1	-1
6	0.5	30	59	56	-1	-1
6	0.5	40	60	55	-2	0
6	0.5	50	59	55	-1	0
6	0.7	10	59	57	-1	-2
6	0.7	20	60	56	-2	-1
6	0.7	30	60	56	-2	-1
6	0.7	40	59	57	-1	-2
6	0.7	50	59	56	-1	-1
6	0.9	10	58	56	0	-1
6	0.9	20	58	57	0	-2
6	0.9	30	59	56	-1	-1
6	0.9	40	59	56	-1	-1
6	0.9	50	59	55	-1	0
6	0.1	10	59	56	-1	-1
6	0.1	20	58	56	0	-1
6	0.1	30	58	55	0	0
6	0.1	40	58	55	0	0
6	0.1	50	58	55	0	0
6	0.3	10	59	56	-1	-1

6	0.3	20	59	55	-1	0
6	0.3	30	58	55	0	0
6	0.3	40	59	55	-1	0
6	0.3	50	58	55	0	0
6	0.5	10	60	56	-2	-1
6	0.5	20	58	55	0	0
6	0.5	30	58	56	0	-1
6	0.5	40	58	55	0	0
6	0.5	50	59	55	-1	0
6	0.7	10	58	55	0	0
6	0.7	20	59	56	-1	-1
6	0.7	30	58	55	0	0
6	0.7	40	58	55	0	0
6	0.7	50	58	56	0	-1
6	0.9	10	60	56	-2	-1
6	0.9	20	59	56	-1	-1
6	0.9	30	59	56	-1	-1
6	0.9	40	58	55	0	0
6	0.9	50	59	55	-1	0
6	0.1	10	59	56	-1	-1
6	0.1	20	58	55	0	0
6	0.1	30	58	55	0	0
6	0.1	40	58	55	0	0
6	0.1	50	58	55	0	0
6	0.3	10	58	56	0	-1
6	0.3	20	58	55	0	0
6	0.3	30	58	55	0	0
6	0.3	40	58	55	0	0
6	0.3	50	58	55	0	0
6	0.5	10	58	56	0	-1
6	0.5	20	58	55	0	0

6	0.5	30	58	55	0	0
6	0.5	40	58	55	0	0
6	0.5	50	58	55	0	0
6	0.7	10	58	56	0	-1
6	0.7	20	58	55	0	0
6	0.7	30	58	55	0	0
6	0.7	40	58	55	0	0
6	0.7	50	58	55	0	0
6	0.9	10	59	56	-1	-1
6	0.9	20	58	55	0	0
6	0.9	30	58	55	0	0
6	0.9	40	58	55	0	0
6	0.9	50	58	55	0	0
6	0.1	10	59	55	-1	0
6	0.1	20	58	55	0	0
6	0.1	30	58	55	0	0
6	0.1	40	58	55	0	0
6	0.1	50	58	55	0	0
6	0.3	10	58	56	0	-1
6	0.3	20	58	55	0	0
6	0.3	30	58	55	0	0
6	0.3	40	58	55	0	0
6	0.3	50	58	55	0	0
6	0.5	10	58	55	0	0
6	0.5	20	58	55	0	0
6	0.5	30	58	55	0	0
6	0.5	40	58	55	0	0
6	0.5	50	58	55	0	0
6	0.7	10	58	55	0	0
6	0.7	20	58	55	0	0
6	0.7	30	58	55	0	0

6	0.7	40	58	55	0	0
6	0.7	50	58	55	0	0
6	0.9	10	59	55	-1	0
6	0.9	20	58	55	0	0
6	0.9	30	58	55	0	0
6	0.9	40	58	55	0	0
6	0.9	50	58	55	0	0
8	0.1	10	60	58	-2	-3
8	0.1	20	62	59	-4	-4
8	0.1	30	59	58	-1	-3
8	0.1	40	61	57	-3	-2
8	0.1	50	60	58	-2	-3
8	0.3	10	61	59	-3	-4
8	0.3	20	60	58	-2	-3
8	0.3	30	60	57	-2	-2
8	0.3	40	60	58	-2	-3
8	0.3	50	61	57	-3	-2
8	0.5	10	64	57	-6	-2
8	0.5	20	61	59	-3	-4
8	0.5	30	60	56	-2	-1
8	0.5	40	59	56	-1	-1
8	0.5	50	59	57	-1	-2
8	0.7	10	61	58	-3	-3
8	0.7	20	59	58	-1	-3
8	0.7	30	61	58	-3	-3
8	0.7	40	60	57	-2	-2
8	0.7	50	60	58	-2	-3
8	0.9	10	63	58	-5	-3
8	0.9	20	62	59	-4	-4
8	0.9	30	60	56	-2	-1
8	0.9	40	60	58	-2	-3

8	0.9	50	59	57	-1	-2
8	0.1	10	60	55	-2	0
8	0.1	20	61	55	-3	0
8	0.1	30	59	56	-1	-1
8	0.1	40	60	56	-2	-1
8	0.1	50	60	56	-2	-1
8	0.3	10	61	57	-3	-2
8	0.3	20	60	56	-2	-1
8	0.3	30	60	57	-2	-2
8	0.3	40	60	56	-2	-1
8	0.3	50	58	57	0	-2
8	0.5	10	59	56	-1	-1
8	0.5	20	60	56	-2	-1
8	0.5	30	59	56	-1	-1
8	0.5	40	59	57	-1	-2
8	0.5	50	59	55	-1	0
8	0.7	10	59	57	-1	-2
8	0.7	20	59	56	-1	-1
8	0.7	30	60	55	-2	0
8	0.7	40	59	56	-1	-1
8	0.7	50	60	56	-2	-1
8	0.9	10	60	56	-2	-1
8	0.9	20	60	57	-2	-2
8	0.9	30	60	56	-2	-1
8	0.9	40	59	56	-1	-1
8	0.9	50	59	56	-1	-1
8	0.1	10	59	55	-1	0
8	0.1	20	59	55	-1	0
8	0.1	30	58	56	0	-1
8	0.1	40	59	55	-1	0
8	0.1	50	58	55	0	0

8	0.3	10	59	56	-1	-1
8	0.3	20	59	56	-1	-1
8	0.3	30	58	56	0	-1
8	0.3	40	59	55	-1	0
8	0.3	50	58	55	0	0
8	0.5	10	58	56	0	-1
8	0.5	20	58	56	0	-1
8	0.5	30	59	55	-1	0
8	0.5	40	59	55	-1	0
8	0.5	50	58	55	0	0
8	0.7	10	59	56	-1	-1
8	0.7	20	59	55	-1	0
8	0.7	30	58	56	0	-1
8	0.7	40	58	55	0	0
8	0.7	50	58	56	0	-1
8	0.9	10	59	56	-1	-1
8	0.9	20	58	56	0	-1
8	0.9	30	58	55	0	0
8	0.9	40	58	55	0	0
8	0.9	50	58	55	0	0
8	0.1	10	58	56	0	-1
8	0.1	20	58	55	0	0
8	0.1	30	58	55	0	0
8	0.1	40	58	55	0	0
8	0.1	50	58	55	0	0
8	0.3	10	58	55	0	0
8	0.3	20	58	55	0	0
8	0.3	30	58	55	0	0
8	0.3	40	58	55	0	0
8	0.3	50	58	55	0	0
8	0.5	10	59	56	-1	-1

8	0.5	20	58	55	0	0
8	0.5	30	58	55	0	0
8	0.5	40	58	55	0	0
8	0.5	50	58	55	0	0
8	0.7	10	59	55	-1	0
8	0.7	20	58	55	0	0
8	0.7	30	58	55	0	0
8	0.7	40	58	55	0	0
8	0.7	50	58	55	0	0
8	0.9	10	58	55	0	0
8	0.9	20	58	56	0	-1
8	0.9	30	58	55	0	0
8	0.9	40	58	55	0	0
8	0.9	50	58	55	0	0
8	0.1	10	59	56	-1	-1
8	0.1	20	58	55	0	0
8	0.1	30	58	55	0	0
8	0.1	40	58	55	0	0
8	0.1	50	58	55	0	0
8	0.3	10	58	55	0	0
8	0.3	20	58	55	0	0
8	0.3	30	58	55	0	0
8	0.3	40	58	55	0	0
8	0.3	50	58	55	0	0
8	0.5	10	58	55	0	0
8	0.5	20	58	55	0	0
8	0.5	30	58	55	0	0
8	0.5	40	58	55	0	0
8	0.5	50	58	55	0	0
8	0.7	10	58	56	0	-1
8	0.7	20	58	56	0	-1

8	0.7	30	58	55	0	0
8	0.7	40	58	55	0	0
8	0.7	50	58	55	0	0
8	0.9	10	59	55	-1	0
8	0.9	20	58	55	0	0
8	0.9	30	58	55	0	0
8	0.9	40	58	55	0	0
8	0.9	50	58	55	0	0