Тренировки по алгоритмам 4.0 от Яндекса — Занятие 3 (Кратчайшие пути в графах)

2 апр 2024, 06:39:12
старт: 8 ноя 2023, 22:00:00
...

Объявления жюри

E. На санях

Ограничение времени	5 секунд
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

- A. Дейкстра
- B. Дейкстра с восстановлением пути
- C. Быстрый алгоритм Дейкстры
- D. Автобусы в Васюках
- E. На санях

В начале XVIII века еще не было самолетов, поездов и автомобилей, поэтому все междугородние зимние поездки совершались на санях. Как известно, с дорогами в России тогда было даже больше проблем, чем сейчас, а именно на N существовавших тогда городов имелась N-1 дорога, каждая из которых соединяла два города. Из каждого города можно было добраться в любой другой (возможно, через промежуточные города). В каждом городе была почтовая станция («ям»), на которой можно было пересесть в другие сани. При этом ямщики могли долго запрягать (для каждого города известно время, которое ямщики в этом городе тратят на подготовку саней к поездке) и быстро ехать (также для каждого города известна скорость, с которой ездят ямщики из него). Можно считать, что количество ямщиков в каждом городе не ограничено.
Если бы олимпиада проводилась 300 лет назад, то путь участников занимал бы гораздо большее время, чем сейчас. Допустим, из каждого города в Москву выезжает участник олимпиады и хочет добраться до Москвы за наименьшее время (не обязательно по кратчайшему пути: он может заезжать в любые города, через один и тот же город можно проезжать несколько раз). Сначала он едет с ямщиком из своего города. Приехав в любой город, он может либо сразу ехать дальше, либо пересесть. В первом случае он едет с той же скоростью, с какой ехал раньше. Если сменить ямщика, он сначала ждет, пока ямщик подготовит сани, и только потом едет с ним (с той скоростью, с которой ездит этот ямщик). В пути можно делать сколько угодно пересадок.

Определите, какое время необходимо, чтобы все участники олимпиады доехали из своего города в Москву 300 лет назад. Все участники выезжают из своих городов одновременно.

Формат ввода

В первой строке входного файла дано натуральное число N, не превышающее 2000 — количество городов, соединенных дорогами. Город с номером 1 является столицей.
Следующие N строк содержат по два целых числа: T_i и V_i — время подготовки саней в городе i, выраженное в часах, и скорость, с которой ездят ямщики из города i, в километрах в час ($0 \le T_i \le 100$, $1 \le V_i \le 100$).

Следующие N–1 строк содержат описания дорог того времени. Каждое описание состоит из трех чисел A_j, B_j и S_j, где A_j и B_j — номера соединенных городов, а S_j — расстояние между ними в километрах ($1 \le A_j \le N$, $1 \le B_j \le N$, $A_j \ne B_j$, $1 \le S_j \le 10000$). Все дороги двусторонние, то есть если из A можно проехать в B, то из B можно проехать в A. Гарантируется, что из всех городов можно добраться в столицу.

Формат вывода

Сначала выведите одно вещественное число — время в часах, в которое в Москву приедет последний участник.
Далее выведите путь участника, который приедет самым последним (если таких участников несколько, выведите путь любого из них). Выведите город, из которого этот участник выехал первоначально, и перечислите в порядке посещения те города, в которых он делал пересадки. Последовательность должна заканчиваться столицей.

При проверке ответ будет засчитан, если из трех величин «время путешествия по выведенному пути», «выведенное время» и «правильный ответ» каждые две отличаются менее чем на 0.0001.

Пример 1

Ввод

```
4

1 1

10 30

5 40

1 10

1 2 300

1 3 400

2 4 100
```

Вывод

```
31.0000000000

4 2 1
```

Пример 2

Ввод

```
3

1 1

0 10

0 55

1 2 100

2 3 10
```

Вывод

```
3.0000000000

2 3 1
```

Примечания

1. Участник из города 1 уже находится на своем месте и тратит на дорогу 0 часов. Участник из города 2 ждет 10 часов ямщика в своем городе, а затем проезжает 300 км от города 2 до города 1 за 10 часов, т.е. тратит на дорогу 20 часов. Участник из города номер 3 ждет ямщика 5 часов, а затем доезжает до города 1 за 10 часов, т.е. тратит на дорогу 15 часов. Участник из города 4 может доехать до города 1 с ямщиком из города 4 за 1 + 40 = 41 час или доехать до города номер 2 за 1 + 10 = 11 часов, прождать там 10 и доехать до столицы за 10 часов. Таким образом, он может добраться до города 1 минимум за 31 час. Это и есть самое большое время и ответ к задаче.
2. Участнику из города 2 выгоднее добраться сначала до третьего города, где ездят быстрее, а потом поехать в столицу, не делая пересадки в своём городе.

Язык Python 3.12.1

Набрать здесь | Отправить файл

```python
fin = open('input.txt')
N = int(fin.readline())

cities = [(float('inf'), 0)]*(N+1)
for i in range(1, N+1):
    cities[i] = [int(x) for x in fin.readline().split()] # start_time, start_speed
cities[1] = (0, 1)

roads = [[] for _ in range(N+1)]
for _ in range(N-1):
    x, y, dist = [int(x) for x in fin.readline().split()]
    roads[x].append((dist, y))
    roads[y].append((dist, x))

all_roads = [[float('inf')]*(N+1) for _ in range(N+1)]
for i in range(N+1):
    all_roads[i][i] = 0

for start_point in range(1, N+1):
    stack, visited = [(0, start_point)], set([start_point])
    while stack:
        curr_dist, curr_point = stack.pop()

        for next_dist, next_point in roads[curr_point]:
            if (next_point not in visited):
                next_dist = curr_dist+next_dist
                next_time = cities[next_point][0]+next_dist/cities[next_point][1]

                if (next_time < all_roads[start_point][next_point]):
                    all_roads[start_point][next_point] = next_time
                    stack.append((next_dist, next_point))
                    visited.add(next_point)

finish_time, change_point = [float('inf')]*(N+1), [None]*(N+1)
finish_time[1], change_point[1] = 0, 1
max_time, max_city = 0, 1

stack, visited = 1, set()
```

Отправить | осталось 63 попытки

Предыдущая

Время посылки	ID	Задача	Компилятор	Вердикт	Тип посылки	Время	Память	Тест	Баллы	
15 ноя 2023, 03:43:39	97363152	E	Python 3.12.1	TL	-	5.053s	144.51Mb	24	-	отчёт
14 ноя 2023, 19:14:19	97287924	E	Python 3.9 (PyPy 7.3.11)	OK	-	1.362s	94.91Mb	-	-	отчёт
14 ноя 2023, 19:13:39	97287763	E	Python 3.9 (PyPy 7.3.11)	WA	-	189ms	27.18Mb	1	-	отчёт
14 ноя 2023, 19:11:09	97287163	E	Python 3.9 (PyPy 7.3.11)	OK	-	1.234s	93.70Mb	-	-	отчёт
14 ноя 2023, 19:10:28	97287002	E	Python 3.12.1	TL	-	5.06s	156.96Mb	24	-	отчёт
14 ноя 2023, 18:59:32	97284569	E	Python 3.12.1	WA	-	48ms	4.14Mb	2	-	отчёт
12 ноя 2023, 11:26:34	96905573	E	Python 3.12.1	TL	-	5.039s	153.36Mb	24	-	отчёт
12 ноя 2023, 11:23:40	96905455	E	Python 3.9 (PyPy 7.3.11)	OK	-	1.199s	94.85Mb	-	-	отчёт
12 ноя 2023, 11:23:15	96905445	E	Python 3.9 (PyPy 7.3.11)	OK	-	1.235s	93.59Mb	-	-	отчёт
12 ноя 2023, 11:21:04	96905364	E	Python 3.12.1	TL	-	5.085s	157.60Mb	24	-	отчёт