Задача 2 второго дня: Пробки на дорогах

Несмотря на большую площадь Канады, многие её районы до сих пор не обжиты, и большая часть населения живёт вблизи южной границы. Трансканадская магистраль, построенная в 1962 году, соединяет людей вдоль этой полоски земли от Сент-Джона на востоке до Виктории на западе. Общая её протяжённость — 7821 км.

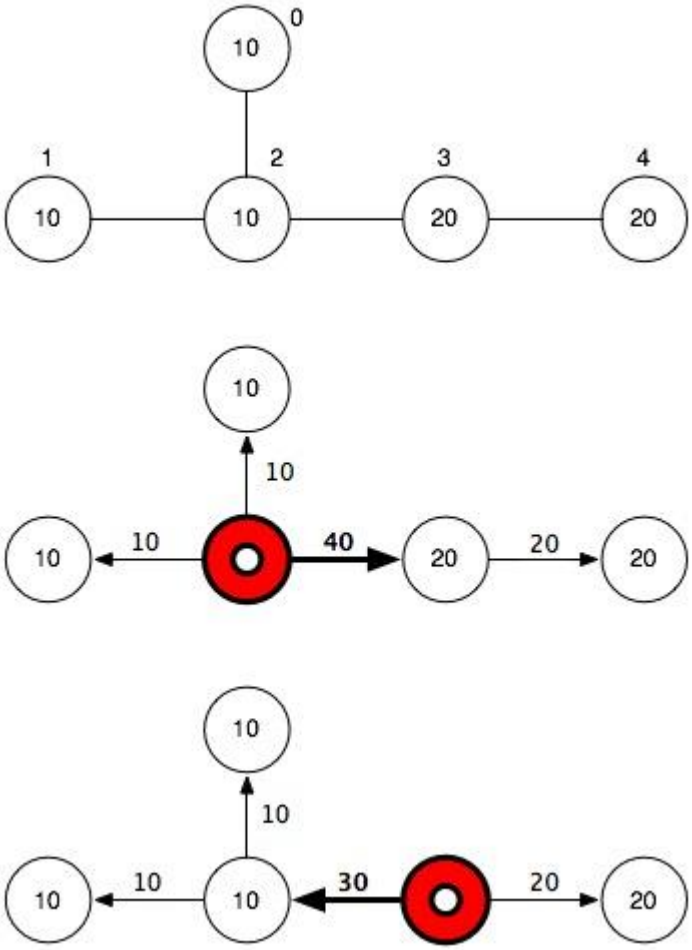
Канадцы любят хоккей. После хоккейного матча тысячи фанатов садятся в свои машины и едут домой от места проведения матча, что вызывает огромные пробки на дорогах. Один богатый инвестор хочет купить хоккейную команду и построить новый хоккейный стадион. Ваша задача — помочь ему выбрать местонахождение стадиона, которое бы минимизировало пробки на дорогах после матча.

Страна состоит из городов, соединённых дорогами. Все дороги двусторонние, и между любыми двумя городами существует ровно один *маршрут*. *Маршрут*, соединяющий города c_0 и c_k — это последовательность различных городов c_0, ..., c_k такая, что существует дорога из c_{i-1} в c_i для всех i. Новый стадион должен быть построен в одном из городов, назовём его городом со стадионом. После хоккейного матча, все фанаты, кроме тех, что живут в городе со стадионом, едут из города со стадионом в свой родной город. Величина пробки на каждой дороге пропорциональна количеству фанатов, которые проедут по этой дороге. Вам необходимо выбрать такое местонахождение стадиона, чтобы величина пробки на дороге с самой большой пробкой была минимально возможной. Если существуют несколько городов, одинаково подходящих для расположения стадиона, вы можете выбирать любой из них.

Вам необходимо реализовать процедуру **LocateCentre(N,P,S,D)**, где **N** — положительное целое число, задающее количество городов. Города пронумерованы от 0 до (N-1). **P** — это массив из N положительных целых чисел. Для каждого i элемент массива P[i] — это количество фанатов, которые живут в городе с номером i. Общее количество фанатов, живущих во всех городах, будет не более 2 000 000 000. **S** и **D** — это массивы из (N-1) целых чисел каждый, которые задают местоположение дорог. Для каждого i существует дорога, соединяющая два города с номерами S[i] и D[i]. Процедура должна возвращать целое число — номер города, в котором необходимо построить стадион.

Пример

Для примера рассмотрим дорожную сеть из пяти городов на верхнем из рисунков справа. На этом рисунке в городах с номерами 0, 1 и 2 живут по 10 фанатов, а в городах с номерами 3 и 4 живут по 20 фанатов. Средний рисунок показывает величину пробок, если стадион будет построен в городе с номером 2. В этом случае наибольшая пробка величиной 40 будет достигаться на дороге, обозначенной жирной стрелкой. Нижний рисунок показывает величину пробок, если стадион будет построен в городе с номером 3. В этом случае наибольшая пробка величиной 30 будет достигаться на дороге, обозначенной жирной стрелкой. Таким образом, город с номером 3 лучше подходит для строительства стадиона, чем город с номером 2. Данные этого примера находятся в файле `grader.in.3a`.



Замечание

Мы напоминаем участникам, что при данных ограничениях можно послать решение, которое проходит подзадачу 3 и не проходит подзадачу 2. Однако, помните, что ваши финальные очки по задаче определяются *только одной* из ваших посылок.

Подзадача 1 [25 баллов]

Предположим, что все города лежат на прямой от востока к западу, и все дороги идут по прямой без ветвлений. Более формально, предположим, что для всех i из диапазона $0 \le i \le N-2$ выполняются соотношения S[i] = i и D[i] = i+1.

Количество городов не превосходит 1 000.

Подзадача 2 [25 баллов]

Решите задачу при таком же предположении, что и в подзадаче 1, но количество городов может достигать 1 000 000.

Подзадача 3 [25 баллов]

Предположение из подзадачи 1 более не выполняется.

Количество городов не превосходит 1 000.

Подзадача 4 [25 баллов]

Предположение из подзадачи 1 более не выполняется.

Количество городов не превосходит 1 000 000.

Детали реализации

- Используйте среду программирования и тестирования RunC
- Папка для реализации: `/home/ioi2010-contestant/traffic/` (прототип: traffic.zip)
- Участник должен реализовать: `traffic.c` или `traffic.cpp` или `traffic.pas`
- Интерфейс участника: `traffic.h` или `traffic.pas`
- Интерфейс системы оценивания: *нет*
- Пример системы оценивания: `grader.c` или `grader.cpp` или `grader.pas`
- Пример ввода для оценивания: `grader.in.1` `grader.in.2`
  Внимание: Первая строка входного файла содержит N. Последующие строки содержат P[i] для i от 0 до (N-1). Последующие (N-1) строк содержат пары `S[i] D[i]` *для i от 0 до (N-2).*
- Ожидаемый вывод на пример ввода: `grader.expect.1` `grader.expect.2` и т.д.
- Компиляция и запуск (из командной строки): `runc grader.c` или `runc grader.cpp` или `runc grader.pas`
- Компиляция и запуск (модуль для gedit): *Control-R*, в момент редактирования файла решения.
- Посылка (из командной строки): `submit grader.c` или `submit grader.cpp` или `submit grader.pas`
- Посылка (модуль для gedit): *Control-J*, в момент редактирования файла решения или системы оценивания.