ふたつの交通機関 (Two Transportations)

JOI 国には N 個の街があり，0 から $N-1$ までの番号が付いている．JOI 国には A 本の鉄道路線があり，0 から $A-1$ までの番号が付いている．鉄道路線 i $(0 \leqq i \leqq A-1)$ は都市 U_i と都市 V_i を双方向に結んでおり，C_i 円で乗ることができる．複数の鉄道路線が同じ都市の組を結んでいることはない．JOI 国には B 本のバス路線があり，0 から $B-1$ までの番号が付いている．バス路線 j $(0 \leqq j \leqq B-1)$ は都市 S_j と都市 T_j を双方向に結んでおり，D_j 円で乗ることができる．複数のバス路線が同じ都市の組を結んでいることはないが，ある鉄道路線とあるバス路線が同じ都市の組を結んでいることはあるかもしれない．どの都市からどの都市へも，いくつかの鉄道路線やバス路線を乗り継ぐことで移動することができる．

Azer は，都市 0 から各都市まで鉄道路線やバス路線を使って移動するときに，少なくともいくらかかるのか知りたい．しかし，Azer は鉄道好きであり鉄道路線のことしか知らないため，バス路線のことしか知らない Baijan と協力することにした．

2 人は，0 または 1 の文字を送受信する機械を用いて通信する．ただし，送受信する文字数の合計を 58 000 以下にしたい．

Azer 側に鉄道路線の情報が，Baijan 側にバス路線の情報が与えられたとき，通信を行うことで，各都市について都市 0 からその都市まで移動するために必要な金額の最小値を Azer 側で求めるプログラムを作成せよ．

実装の詳細

あなたは 2 つのファイルを提出しなければならない．

1 つ目のファイルは `Azer.cpp` という名前である．このファイルは Azer の行動を実装したファイルであり，以下の関数を実装していなければならない．そのファイルは `Azer.h` をインクルードすること．

- `void InitA(int N, int A, std::vector<int> U, std::vector<int> V, std::vector<int> C)`
  この関数は，最初に 1 回だけ実行される．
  - 引数 `N` は，街の数 N である．
  - 引数 `A` は，鉄道路線の数 A である．
  - 引数 `U`，`V` は，長さ A の配列であり，`U[i]`，`V[i]` は鉄道路線 i が結んでいる都市 U_i, V_i を表す $(0 \leqq i \leqq A-1)$.
  - 引数 `C` は，長さ A の配列であり，`C[i]` は鉄道路線 i の運賃 C_i を表す $(0 \leqq i \leqq A-1)$.
- `void ReceiveA(bool x)`
  この関数は，Baijan から文字が送られる度に実行される．
  - 引数 `x` は，Baijan から送信された文字を表す．`true` は文字 1 を表し，`false` は文字 0 を表す．

- `std::vector<int> Answer()`
  この関数は，文字の送受信がすべて完了したときに 1 回だけ実行される．都市 0 から各都市まで移動するために必要な金額の最小値を表す配列 (以下 `Z` とする) を戻り値として返さなければならない．
  - 戻り値 `Z` は，長さ N の配列でなければならない．長さが N でない場合，不正解 **[1]** と判定される．`Z[k]` $(0 \leqq k \leqq N-1)$ は，都市 0 から都市 k まで移動するために必要な金額の最小値でなければならない．特に，`Z[0]` $= 0$ でなければならないことに注意せよ．

このファイルでは以下の関数を呼び出すことができる．

★ `void SendA(bool y)`
この関数を用いて，Baijan に文字を送信する．
- 引数 `y` は，Baijan に送信する文字を表す．`true` は文字 1 を表し，`false` は文字 0 を表す．

2 つ目のファイルは `Baijan.cpp` という名前である．このファイルは Baijan の行動を実装したファイルであり，以下の関数を実装していなければならない．そのファイルは `Baijan.h` をインクルードすること．

- `void InitB(int N, int B, std::vector<int> S, std::vector<int> T, std::vector<int> D)`
  この関数は，最初に 1 回だけ実行される．
  - 引数 `N` は，街の数 N である．
  - 引数 `B` は，バス路線の数 B である．
  - 引数 `S`，`T` は，長さ B の配列であり，`S[j]`，`T[j]` はバス路線 j が結んでいる都市 S_j, T_j を表す $(0 \leqq j \leqq B-1)$.
  - 引数 `D` は，長さ B の配列であり，`D[j]` はバス路線 j の運賃 D_j を表す $(0 \leqq j \leqq B-1)$.
- `void ReceiveB(bool y)`
  この関数は，Azer から文字が送られる度に実行される．
  - 引数 `y` は，Azer から送信された文字を表す．`true` は文字 1 を表し，`false` は文字 0 を表す．

このファイルでは以下の関数を呼び出すことができる．

★ `void SendB(bool x)`
この関数を用いて，Azer に文字を送信する．
- 引数 `x` は，Azer に送信する文字を表す．`true` は文字 1 を表し，`false` は文字 0 を表す．

プログラムは次のように実行されると考えてよい．各テストケースにおいて，Azer が送信した文字を保管するキュー Q_Y と Baijan が送信した文字を保管するキュー Q_X が用意される．最初に `InitA` および `InitB` が呼び出され，そこで送信された文字がそれぞれのキューに追加される．次に，以下の処理を繰り返す．

- Q_X と Q_Y のいずれかが空でない場合，そこから文字を 1 つ取り出し，対応する `ReceiveA` または `ReceiveB` が実行される．ただし，Q_X と Q_Y のどちらも空でない場合，`ReceiveA` と `ReceiveB` のどちらが実行されるかは分からない．
- `ReceiveA` の実行中に `SendA` が実行された場合，送信された文字が Q_Y に追加される．
- `ReceiveB` の実行中に `SendB` が実行された場合，送信された文字が Q_X に追加される．
- どちらのキューも空である場合は，`Answer` が実行された後，プログラムが終了する．

Azer と Baijan が送信する文字数の合計は 58 000 を超えてはならない．もし超えた場合，**不正解 [2]** と判定される．

重要な注意

- 内部での使用のために他の関数を実装したり，グローバル変数を宣言するのは自由である．ただし，提出された 2 つのプログラムは，採点プログラムとまとめてリンクされて 1 つの実行ファイルになるので，各ファイル内のすべてのグローバル変数と内部関数を無名名前空間内で宣言して，他のファイルとの干渉を避ける必要がある．採点時には，このプログラムは Azer 側，Baijan 側として 2 個のプロセスとして実行されるので，Azer 側と Baijan 側でプログラム中のグローバル変数を共有することはできない．
- あなたの提出したプログラムは，標準入力・標準出力，あるいは他のファイルといかなる方法でもやりとりしてはならない．ただし，標準エラー出力にデバッグ情報等を出力することは許される．

コンパイル・実行の方法

作成したプログラムをテストするための，採点プログラムのサンプルが，コンテストサイトからダウンロードできるアーカイブの中に含まれている．このアーカイブには，提出しなければならないファイルのサンプルも含まれている．

採点プログラムのサンプルは 1 つのファイルからなる．そのファイルは `grader.cpp` である．作成したプログラムをテストするには，`grader.cpp`, `Azer.cpp`, `Baijan.cpp`, `Azer.h`, `Baijan.h` を同じディレクトリに置き，次のようにコマンドを実行する．

```
g++ -std=gnu++14 -O2 -o grader grader.cpp Azer.cpp Baijan.cpp
```

コンパイルが成功すれば，`grader` という実行ファイルが生成される．

実際の採点プログラムは，採点プログラムのサンプルとは異なることに注意すること．採点プログラムのサンプルは単一のプロセスとして起動する．このプログラムは，標準入力から入力を読み込み，標準出力と標準エラー出力に結果を出力する．

採点プログラムのサンプルの入力

採点プログラムのサンプルは標準入力から以下の形式で入力を読み込む.

$N\ A\ B$
$U_0\ V_0\ C_0$
$\vdots$
$U_{A-1}\ V_{A-1}\ C_{A-1}$
$S_0\ T_0\ D_0$
$\vdots$
$S_{B-1}\ T_{B-1}\ D_{B-1}$

採点プログラムのサンプルの出力

採点プログラムのサンプルは標準出力と標準エラー出力へ以下の情報を出力する (引用符は実際には出力されない).

- **不正解 [1]** または **不正解 [2]** と判定された場合，標準エラー出力に，不正解の種類が “`Wrong Answer [1]`” のように出力される．標準出力には何も出力されない.
- そうでない場合，標準エラー出力に，送受信された文字数の合計 L が “`Accepted: L`” のように出力される．また，標準出力にあなたの回答 `Z` が

  ```
  Z[0]
  ⋮
  Z[N - 1]
  ```

  のように出力される．採点プログラムのサンプルは，`Z` の値が正しいかどうかは判定しない.

実行するプログラムが複数の不正解の条件を満たした場合，表示される不正解の種類はそれらのうち 1 つのみである.

制約

- $1 \leqq N \leqq 2\,000$.
- $0 \leqq A \leqq 500\,000$.
- $0 \leqq B \leqq 500\,000$.
- $0 \leqq U_i \leqq N - 1$ $(0 \leqq i \leqq A - 1)$.
- $0 \leqq V_i \leqq N - 1$ $(0 \leqq i \leqq A - 1)$.
- $U_i \neq V_i$ $(0 \leqq i \leqq A - 1)$.
- $(U_{i_1}, V_{i_1}) \neq (U_{i_2}, V_{i_2})$ かつ $(U_{i_1}, V_{i_1}) \neq (V_{i_2}, U_{i_2})$ $(0 \leqq i_1 < i_2 \leqq A - 1)$.
- $0 \leqq S_j \leqq N - 1$ $(0 \leqq j \leqq B - 1)$.
- $0 \leqq T_j \leqq N - 1$ $(0 \leqq j \leqq B - 1)$.
- $S_j \neq T_j$ $(0 \leqq j \leqq B - 1)$.
- $(S_{j_1}, T_{j_1}) \neq (S_{j_2}, T_{j_2})$ かつ $(S_{j_1}, T_{j_1}) \neq (T_{j_2}, S_{j_2})$ $(0 \leqq j_1 < j_2 \leqq B - 1)$.
- どの都市からどの都市へも，いくつかの鉄道路線やバス路線を乗り継ぐことで移動することができる．
- $1 \leqq C_i \leqq 500$ $(0 \leqq i \leqq A - 1)$.
- $1 \leqq D_j \leqq 500$ $(0 \leqq j \leqq B - 1)$.

小課題

1. (6 点) $A = 0$.
2. (8 点) $B \leqq 1\,000$.
3. (8 点) $A + B = N - 1$.
4. (38 点) $N \leqq 900$.
5. (14 点) $N \leqq 1\,100$.
6. (10 点) $N \leqq 1\,400$.
7. (16 点) 追加の制約はない．

やり取りの例

採点プログラムのサンプルが読み込む入力の例と，それに対応する関数の呼び出しの例を以下に示す．

入力例 1	関数の呼び出しの例		
	呼び出し	呼び出し	戻り値
4 3 4	InitA(4, 3, {0,2,2}, {1,1,0}, {6,4,10})		
0 1 6		SendA(true)	
2 1 4		SendA(false)	
2 0 10	InitB(4, 4, {1,3,3,3}, {2,1,2,0}, {3,1,3,7})		
1 2 3	ReceiveB(true)		
3 1 1		SendB(true)	
3 2 3	ReceiveA(true)		
3 0 7	ReceiveB(false)		
	Answer()		{0,6,9,7}