Two Transportations

There are N cities in JOI country, numbered from 0 to $N-1$. There are A railway lines, numbered from 0 to $A-1$. The railway line i ($0 \leq i \leq A-1$) connects the city U_i and the city V_i bidirectionally, and its fare is C_i. Different railway lines connect different pairs of cities. There are B bus lines, numbered from 0 to $B-1$. The bus line j ($0 \leq j \leq B-1$) connects the city S_j and the city T_j bidirectionally, and its fare is D_j. Different bus lines connect different pairs of cities, but a railway line and a bus line might connect the same pair of cities. It is possible to travel between any pair of cities by taking railways and/or buses.

Azer wants to know the minimum total fare needed to travel from the city 0 to each city. As Azer knows only about railway lines, he cooperates with Baijan, who knows only about bus lines.

They communicate with each other by sending and receiving characters 0 or 1. The total number of the sent characters should be less than or equal to 58 000.

Write programs which, Azer's program given the information of railway lines, and Baijan's given the information of bus lines, communicating with each other, help Azer find the minimum total fare needed to travel from the city 0 to each city.

Implementation Details

You need to submit two files.

The name of the first file is `Azer.cpp`. It represents the behavior of Azer and should implement the following functions. The file should include `Azer.h`.

- `void InitA(int N, int A, std::vector<int> U, std::vector<int> V, std::vector<int> C)`
  This function is executed exactly once at the beginning.
  - Parameter `N` is N, the number of cities.
  - Parameter `A` is A, the number of railway lines.
  - Parameters `U`, `V` are arrays of length A. `U[i]` and `V[i]` are U_i and V_i, the cities connected by the railway line i ($0 \leq i \leq A-1$).
  - Parameter `C` is an array of length A. `C[i]` is C_i, the fare of the railway line i ($0 \leq i \leq A-1$).
- `void ReceiveA(bool x)`
  This function is executed every time a character is sent from Baijan.
  - The parameter `x` represents the character sent from Baijan: `true` for character 1, and `false` for character 0.

- `std::vector<int> Answer()`
  This function is executed exactly once when all the sent characters have been received. This function must return an array `Z` containing the minimum total fare needed to travel from the city 0 to each city.
  - The return value `Z` must be an array of length N. If its length is not N, your program is judged as **Wrong Answer [1]**. `Z[k]` ($0 \leq k \leq N-1$) must be the minimum total fare needed to travel from the city 0 to the city k. In particular, note that `Z[0]` $= 0$ must be satisfied.

Your program can call the following function in this file.

★ `void SendA(bool y)`
Use this function to send a character to Baijan.
- The parameter `y` represents the character to send to Baijan: `true` for character 1, and `false` for character 0.

The name of the second file is `Baijan.cpp`. It represents the behavior of Baijan and should implement the following functions. The file should include `Baijan.h`.

- `void InitB(int N, int B, std::vector<int> S, std::vector<int> T, std::vector<int> D)`
  This function is executed exactly once at the beginning.
  - Parameter `N` is N, the number of cities.
  - Parameter `B` is B, the number of bus lines.
  - Parameters `S`, `T` are arrays of length B. `S[j]` and `T[j]` are S_j and T_j, the cities connected by bus line j ($0 \leq j \leq B-1$).
  - Parameter `D` is an array of length B. `D[j]` is D_j, the fare of bus line j ($0 \leq j \leq B-1$).
- `void ReceiveB(bool y)`
  This function is executed every time a character is sent from Azer.
  - The parameter `y` represents the character sent from Azer: `true` for character 1, and `false` for character 0.

Your program can call the following function in this file.

★ `void SendB(bool x)`
Use this function to send a character to Azer.
- The parameter `x` represents the character to send to Azer: `true` for character 1, and `false` for character 0.

You can assume that the program is executed as follows. For each test case, two queues are prepared: Q_Y, where characters Azer sends are stored, and Q_X, where characters Baijan sends are stored. First, `InitA` and `InitB` are executed, and the sent characters are pushed to the corresponding queues.

- If either Q_X or Q_Y is not empty, one character is popped from it and corresponding `ReceiveA` or `ReceiveB` is executed. However, if both Q_X and Q_Y are not empty, it is not determined whether `ReceiveA` or `ReceiveB` is executed.
- When `SendA` is executed during executions of `ReceiveA`, the sent character is pushed to Q_Y.
- When `SendB` is executed during executions of `ReceiveB`, the sent character is pushed to Q_X.
- If both queues are empty, `Answer` is executed and the program ends.

The total number of characters Azer and Baijan send should be less than or equal to 58 000. If it is larger, your program is judged as **Wrong Answer [2]**.

Important Notices

- Your program can implement other functions for internal use, or use global variables. Submitted files will be compiled with the grader, and become a single executable file. All global variables and internal functions should be declared in an unnamed namespace to avoid confliction with other files. When it is graded, it will be executed as two processes of Azer and Baijan. The process of Azer and the process of Baijan cannot share global variables.
- Your program should not use the standard input and the standard output. Your program should not communicate with other files by any methods. However, your program may output debugging information to the standard error.

Compilation and Test Run

You can download an archive file from the contest webpage which contains the sample grader to test your program. The archive file also contains sample source files of your program.

The sample grader is the file `grader.cpp`. In order to test your program, put `grader.cpp`, `Azer.cpp`, `Baijan.cpp`, `Azer.h`, and `Baijan.h` in the same directory, and run the following command to compile your program.

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

If the compilation succeeds, the executable file `grader` is generated.

Note that the actual grader is different from the sample grader. The sample grader will be executed as a single process, which will read input data from the standard input and write the results to the standard output and the standard error.

Input for the Sample Grader

The sample grader reads the input data from the standard input in the following format.

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

Output of the Sample Grader

The sample grader writes the following information to the standard output and the standard error (quotes for clarity).

- If your program is judged as Wrong Answer [1] or Wrong Answer [2], it writes its type as "`Wrong Answer [1]`" to the standard error. It writes nothing to the standard output.
- Otherwise, it writes the total number of the sent characters L as "`Accepted: L`" to the standard error. It also writes your answer `Z` to the standard output as follows:

  ```
  Z[0]
  ⋮
  Z[N - 1]
  ```

  The sample grader does not check if the value of `Z` is correct.

If your program is judged as several types of Wrong Answer, the sample grader reports only one of them.

Constraints

- $1 \leq N \leq 2\,000$.
- $0 \leq A \leq 500\,000$.
- $0 \leq B \leq 500\,000$.
- $0 \leq U_i \leq N-1$ $(0 \leq i \leq A-1)$.
- $0 \leq V_i \leq N-1$ $(0 \leq i \leq A-1)$.
- $U_i \neq V_i$ $(0 \leq i \leq A-1)$.
- $(U_{i_1}, V_{i_1}) \neq (U_{i_2}, V_{i_2})$ and $(U_{i_1}, V_{i_1}) \neq (V_{i_2}, U_{i_2})$ $(0 \leq i_1 < i_2 \leq A-1)$.
- $0 \leq S_j \leq N-1$ $(0 \leq j \leq B-1)$.
- $0 \leq T_j \leq N-1$ $(0 \leq j \leq B-1)$.
- $S_j \neq T_j$ $(0 \leq j \leq B-1)$.
- $(S_{j_1}, T_{j_1}) \neq (S_{j_2}, T_{j_2})$ and $(S_{j_1}, T_{j_1}) \neq (T_{j_2}, S_{j_2})$ $(0 \leq j_1 < j_2 \leq B-1)$.
- It is possible to travel between any pair of cities by taking railways and/or buses.
- $1 \leq C_i \leq 500$ $(0 \leq i \leq A-1)$.
- $1 \leq D_j \leq 500$ $(0 \leq j \leq B-1)$.

Subtasks

1. (6 points) $A = 0$.
2. (8 points) $B \leq 1\,000$.
3. (8 points) $A + B = N - 1$.
4. (38 points) $N \leq 900$.
5. (14 points) $N \leq 1\,100$.
6. (10 points) $N \leq 1\,400$.
7. (16 points) No additional constraints.

Sample Communication

Here is a sample input for the sample grader and corresponding function calls.

Sample Input 1	Sample Calls		
	Call	Call	Return
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}`