Autobusy i tramwaje

WWI 2024 – Kolano
18 sierpnia 2024 – 23 sierpnia 2024

Kod zadania: **aut**
Limit pamięci: **512 MiB**

Takie zwykłe masz ciało takie szare
Takie nudne są dni bo takie same
Gdy o świcie do pracy swojej wstajesz
Takie zwykłe masz ciało bo takie szare

Autobusy i tramwaje
Autobusy i tramwaje
Autobusy i tramwaje
Autobusy i tramwaje

T.Love – *Autobusy i tramwaje*

On zna się na autobusach, a ona na tramwajach. Szukają optymalnego miejsca na romantyczną randkę. Nie są bardzo bogaci a komunikacja miejska w Warszawie jest droga jak nie wiadomo co, więc warto wiedzieć, ile kosztuje dojazd w różne miejsca.

Możemy założyć, że Warszawa składa się z n skrzyżowań, numerowanych od zera, połączonych trasami autobusowymi i tramwajowymi. Każda trasa łączy dwa skrzyżowania i można przejechać nią w dowolną stronę. Każda trasa ma swój koszt przejazdu. Oboje mieszkają przy skrzyżowaniu numer 0, koszt przejazdu jest wyliczony dla dwojga. Chcieliby poznać koszty przejazdu do poszczególnych skrzyżowań. Do każdego skrzyżowania da się dojechać na co najmniej jeden sposób.

Niestety, do czasu umówionej randki już się nie zobaczą. Nie przeszkadza to jednak gołąbkom w wymienianiu dużej liczby SMS-ów. Niestety SMS-y poza tym że są dość uciążliwe (każdy SMS może przenosić cały jeden bit danych) również kosztują, zatem chcą ich wysłać jak najmniej. Kiedy on będzie wiedział, ile kosztuje dojazd do każdego skrzyżowania, wybierze miejsce i kupi bilety. Pomóż im wyznaczyć optymalny koszt dojazdu do wszystkich skrzyżowań jak najtaniej.

Komunikacja

Jest to zadanie komunikacyjne, zatem Twój program będzie porozumiewał się z biblioteką. Aby użyć biblioteki, należy załączyć nagłówek `#include "autlib.h"`. Z uwagi na ograniczenia środowiska, to zadanie można rozwiązywać wyłącznie w języku C++.

Wymagane funkcje

Twoim zadaniem jest zaimplementowanie pięciu funkcji:

- `void autobusy(int n, int a, std::vector<int> u, std::vector<int> v, std::vector<int> k)`
  On dowiaduje się o liczbie `n` skrzyżowań w Warszawie ($1 \leq$ `n` $\leq 2\,000$) i liczbie `a` tras autobusowych ($0 \leq$ `a` $\leq 500\,000$). Wektory `u`, `v` i `k` są długości `a` i opisują trasy autobusowe – i-ta (licząc od zera) trasa łączy skrzyżowania `u`$[i]$ z `v`$[i]$, a przejazd nią kosztuje `k`$[i]$ ($1 \leq$ `k`$[i] \leq 500$).

- `void tramwaje(int n, int b, std::vector<int> u, std::vector<int> v, std::vector<int> k)`
  Ona dowiaduje się o liczbie `n` skrzyżowań w Warszawie ($1 \leq$ `n` $\leq 2\,000$) i liczbie `b` tras tramwajowych ($0 \leq$ `b` $\leq 500\,000$). Wektory `u`, `v` i `k` są długości `b` i opisują trasy tramwajowe – i-ta (licząc od zera) trasa łączy skrzyżowania `u`$[i]$ z `v`$[i]$, a przejazd nią kosztuje `k`$[i]$ ($1 \leq$ `k`$[i] \leq 500$).

- `void smsOdNiej(bool x)`
  On odbiera SMS od niej.

- `void smsOdNiego(bool x)`
  Ona odbiera SMS od niego.

- `std::vector<int> odpowiedz()`
  On zwraca wektor, w którym liczba na i-tym (licząc od zera) indeksie oznacza koszt przejazdu do i-tego skrzyżowania. Następnie program zakończy działanie i przydzieli punkty w zależności od poprawności wektora zwróconego przez funkcję `odpowiedz`.

Możesz tworzyć dodatkowe funkcje pomocnicze, ale **nie implementuj** funkcji `main`.

Dostarczane funkcje

Biblioteka dostarcza dwie funkcje, z których można korzystać:

- `void smsDoNiej(bool x)`
  On wysyła do niej SMS zawierający bit `x`.
- `void smsDoNiego(bool x)`
  Ona wysyła do niego SMS zawierający bit `x`.

Działanie programu

Możesz założyć, że program tworzy **kolejki** przekazywanych SMS-ów – kolejkę Q_{XY} zawierającą SMS-y wysłane przez niego oraz kolejkę Q_{XX} zawierającą SMS-y wysłane przez nią. Na początku wykonywane są funkcje `autobusy` i `tramwaje`. Później powtarzana jest sekwencja ruchów:

- Jeśli Q_{XY} albo Q_{XX} nie jest pusta, jeden SMS zostaje usunięty z odpowiedniej kolejki oraz podany odpowiednio funkcji `smsOdNiego` albo `smsOdNiej`. W przypadku, gdy obie kolejki nie są puste nie jest zagwarantowane, która funkcja zostanie wywołana jako pierwsza.
- Jeśli Q_{XY} i Q_{XX} są puste, wywołana zostaje funkcja `odpowiedz`, a następnie zakończone działanie programu.
- Za każdym wywołaniem funkcji `smsDoNiej` SMS podany tej funkcji wysłany zostaje do kolejki Q_{XY}. Analogicznie, przy wywołaniu funkcji `smsDoNiego` jest on wysłany do kolejki Q_{XX}.

Kompilacja na swoim komputerze

Pliki z archiwum `aut_dlazaw.zip` dostępnego w zakładce „Pliki" należy wypakować do folderu z kodem źródłowym programu. Aby program skompilował się, należy załączyć nagłówek `#include "autlib.h"`.

Program należy skompilować razem z biblioteką `autlib.cc`. Można to zrobić za pomocą polecenia:
`g++ aut.cpp autlib.cc -o aut -std=c++20 -O3 -static`.

Wejście i wyjście

Program nie powinien nic czytać ze standardowego wejścia (`stdin`/`cin`), ani wypisywać na standardowe wyjście (`stdout`/`cout`).
Nie powinien także zawierać funkcji `int main()`. Dozwolone jest pisanie na standardowe wyjście diagnostyczne (`stderr`/`cerr`/`clog`), lecz pamiętaj, że zabiera to cenny czas.

Przykładowa biblioteczka

Rozumienie kodu biblioteczki nie jest potrzebne (ani zalecane) do obsługiwania jej. Przykładowa biblioteczka przyjmuje testy w następującej postaci: w pierwszym wierszu wejścia znajdują się dwie liczby całkowite: `n`, `a`, w kolejnych `a` wierszach znajdują się po trzy liczby – `u`[i], `v`[i] i `k`[i] (parametry funkcji `autobusy`). W kolejnym wierszu wejścia znajdują się dwie liczby całkowite: `n`, `b`, w kolejnych `b` wierszach znajdują się po trzy liczby – `u`[i], `v`[i] i `k`[i] (parametry funkcji `tramwaje`). Przykładowa biblioteczka wypisuje długość, a następnie wartości wektora zwróconego przez funkcję `odpowiedz`.

Ocenianie

To jest Kolano. Jeżeli Twój program zmieści się w limicie czasu (wynoszącym 10 sekund) oraz pamięci i odpowie poprawnie, liczba punktów będzie proporcjonalna do łącznej liczby wymienionych SMS-ów (im mniej, tym lepiej). W przeciwnym wypadku, program otrzyma ∞ punktów.

Celem zadania jest optymalizacja przekazywania informacji. Zadanie jest lekko zabezpieczone przed oszustwem, ale wierzę, że wciąż da się oszukać. Rozwiązania wykorzystujące oszustwo nie będą tolerowane.

Przykład

Wejście dla testu `aut0a`:

```
4 3
0 1 6
2 1 4
2 0 10
4 4
1 2 3
3 1 1
3 2 3
3 0 7
```

Wyjście dla testu `aut0a`:

```
4
0 6 9 7
```

Wejście dla testu `aut0b`:

```
7 4
2 3 171
0 6 192
1 2 269
2 4 236
7 2
2 5 318
3 0 240
```

Wyjście dla testu `aut0b`:

```
7
0 680 411 240 647 729 192
```

Wejście dla testu `aut0c`:

```
4 0
4 5
3 2 214
1 2 460
0 1 178
2 0 471
1 3 475
```

Wyjście dla testu `aut0c`:

```
4
0 178 471 653
```