THE 7

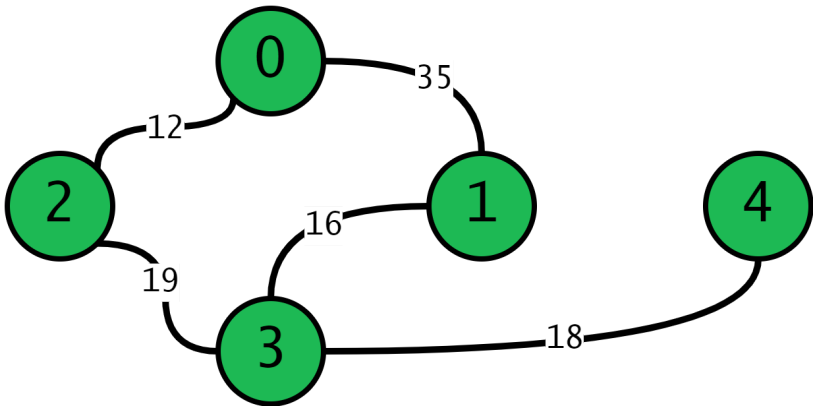
Available from: Thursday, January 13, 2022, 11:55 AM
Due date: Friday, January 14, 2022, 11:59 PM
Requested files: the7.cpp (Download)
Type of work: Individual work

Becoming a Master

Jimmy needs to get his Master's thesis signed by three people to complete his degree. One of those people is his supervisor. Other two are Prof.X and Prof.Y. However, he doesn't have much time left. Jimmy needs to visit all three of the professors to get signatures and then leave the completely signed thesis at his supervisor as soon as possible. If he does not he will be expelled from his university and won't get his degree. He knows that if he takes the shortest route from his house to his supervisor, visiting Prof. X and Y along the way he can get it done. Help him find a way.

Jimmy lives in a city with **N** houses. Each house is numbered from **0** to **N-1**. Each house is connected to another one by a road and there are **M** roads in the city. Each road can be travelled in both directions. Moreover, travelling each road takes **T** minutes. Jimmy 's house no is **S**, his supervisor's house no is **D**, Prof.X's house no is **X** and Prof.Y's house no is **Y**. So, you need to find the quickest route from **S** that ends in **D** which also visits **X** and **Y** ***at least once in any order***. Additionally, Jimmy can visit a house (including S, D, X, and Y) *multiple times*.

Example:



N=5
S=0, **D**=4
X=1, **Y**=2
M=5
Roads=[{0, 1, 35}, {0, 2, 12}, {1, 3, 16}, {2, 3, 19}, {3, 4, 18}]
^
T

You need to write a function called *FindRoute*, which takes the number of houses (***N***), information about the ***M*** Roads, Jimmy 's house no (***S***), Supervisor's house no (***D***), Prof. X's house no (***X***), and Prof. Y's house no (***Y***) and prints the shortest time Jimmy can deliver his thesis to his supervisor and the path he should follow.

Road information is given to you as a vector of **Road** structs. A **Road** struct contains two *endpoints* of the road and the *time* it takes to travel that road.

```
struct Road {
  std::pair<int, int> endpoints;
  int time;
  Road(std::pair<int, int> eps, int t) : endpoints(eps), time(t) {}
};
```

FindRoute's function declaration is:

```
void FindRoute(int n, std::vector<Road> roads, int s, int d, int x, int y);
```

Inputs:
----
n = number of houses
roads = infromation about roads supplied with Road structs.
s = Jimmy 's house no
d = Supervisor's house no
x = Prof. X's house no
y = Prof. Y's house no

Outputs:

You need to print the time it takes Jimmy to deliver his thesis as fast as possible and the path Jimmy should take separated by a single space. Path should be printed in order and by separating the house numbers in the path with '->'. So output must be in the format of "<time> <path>" (without the angled brackets).

An example output can be:

```
42 1->5->2->1->5->3
```

Note that the last element in the path does not have '->' or a space after it and the output ends with a newline char ('\n').

*Hint: You can use the supplied **PrintRange** function to easily print elements of containers with iterators.*

Example IO:

Input: n=5, s=3, d=2, x=1, y=4, m=9,
roads=[{0, 1, 13}, {0, 2, 13}, {0, 3, 18}, {1, 2, 19}, {1, 3, 10}, {1, 4, 18}, {2, 3, 13}, {2, 4, 20}, {3, 4, 18}]

Output:

```
48 3->1->4->2
```

Input: n=7, s=1, d=3, x=5, y=4, m=7,
roads=[{0, 6, 11}, {0, 1, 18}, {1, 3, 19}, {1, 4, 16}, {2, 3, 10}, {3, 4, 10}, {5, 6, 11}]

Output:

```
106 1->0->6->5->6->0->1->4->3
```

Input: n=8, s=4, d=7, x=6, y=5, m=20,
roads=[{0, 2, 16}, {0, 3, 15}, {0, 4, 14}, {0, 6, 12}, {0, 7, 13}, {1, 2, 15}, {1, 3, 11}, {1, 5, 11}, {1, 6, 16}, {1, 7, 11}, {2, 4, 15}, {2, 5, 15}, {2, 6, 19}, {3, 5, 14}, {3, 6, 10}, {3, 7, 17}, {4, 5, 18}, {5, 6, 11}, {5, 7, 16}, {6, 7, 11}]

Output:

```
40 4->5->6->7
```

Note: Road struct is represented as '{endpoints.first, endpoints.second, time}'

Constraints:

- 4 ≤ ***N*** ≤ 500
- 1 ≤ ***T*** ≤ 250
- All of ***N, S, D, X,*** and ***Y*** are integers.

Specifications:

- Jimmy can always reach X, Y and D from S.
- Your output must *exactly* be in the specified format.
- Consider infinity as **INT_MAX** (defined in <climits>)
- Total time for a path won't exceed **INT_MAX**.
- There is **1 task** to be solved in **12 hours** in this take home exam.
- You will implement your solutions in "**the7.cpp**" file.
- You are free to add other functions, classes, structs etc. to "**the7.cpp**"
- **Do not** change the first line of the "**the7.cpp**"
- **Do not** change the arguments of ***FindRoute*** function.
- Some headers, structs and utility functions are defined in "**the7.h**". Full contents of the7.h are below.

```
#pragma once

#include <climits>
#include <iostream>
#include <queue>
#include <stack>
#include <unordered_map>
#include <vector>

template <class ForwardIt>
void PrintRange(ForwardIt first, ForwardIt last, char const *delim = "->") {
  --last;
  for (; first != last; ++first) {
    std::cout << *first << delim;
  }
  std::cout << *first;
}

struct Road {
  std::pair<int, int> endpoints;
  int time;
  Road(std::pair<int, int> eps, int t) : endpoints(eps), time(t) {}
};

void FindRoute(int n, std::vector<Road> roads, int s, int d, int x, int y);
```

- A collection of files have been uploaded to *odtuclass course page* with name ***THE7Files*** that you can use if you want to work locally. Note that included test cases are the same ones in the VPL.
- Your code will be compiled with **-Wall** and **-std=c++11** flags.
- You can test your the7.cpp on virtual lab environment. If you click run, your function will be compiled and executed with sample test cases. If you click evaluate, your work will be temporarily graded for limited number of inputs.
- The grade you see in lab is not your final grade, your code will be reevaluated with completely different inputs after the exam.

Evaluation:

- After your exam, black box evaluation will be carried out. You will get full points if you print the time and the path *exactly* as described to stdout.

The system has the following limits:

- a maximum execution time of 10 seconds (your functions should return in less than 2-3 seconds for the largest inputs)
- a 192 MB maximum memory limit
- an execution file size of 1M.
- Solutions with longer running times **will not be graded**.
- If you are sure that your solution works in the expected complexity constrains but your evaluation fails due to limits in the lab environment, the constant factors may be the problem.

Appendix:

The main in the main.cpp in supplied files has a function that reads input files and calls FindRoute function. The path to an input file is given as a commandline argument to the executable. You can compile and execute the

given code with:

```
g++ -Wall -std=c++11 -o the7 main.cpp the7.cpp
./the7 path/to/test/input
```

The format of given sample input files is as follows.

```
<N>
<S> <D>
<X> <Y>
<M>
<endpoint#1> <endpoint#2> <time>  <- Represents a road.
...
```