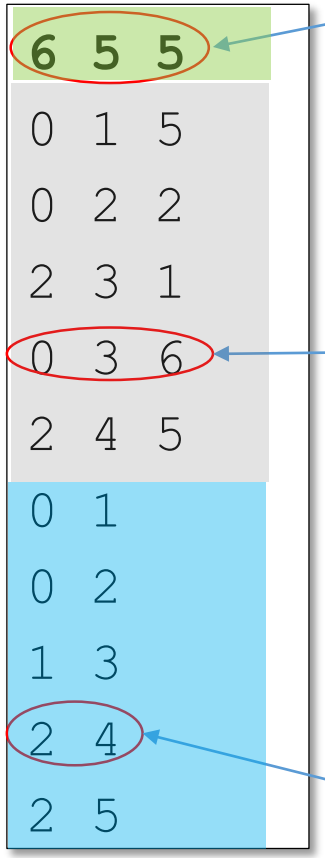
Assignment 5

SSSP

Settings

- In this assignment, we will compute the path and the distance of the shortest path between two nodes. You can assume our graphs are:
  - Directed
  - At least one node but no limitation on the number of edges.
  - Each edge has a non-zero positive weight $w > 0$. Namely, there will be no edge with weight 0 or less
  - You can assume all computations are in integers.

Example5



e edges

q queries

First line: There will v (6) nodes, e (5) edges and q (5) queries.

A directed edge from Node 0 to Node 3 with weight 6.
(Note that it will not create the edge from Node 6 to Node 3)

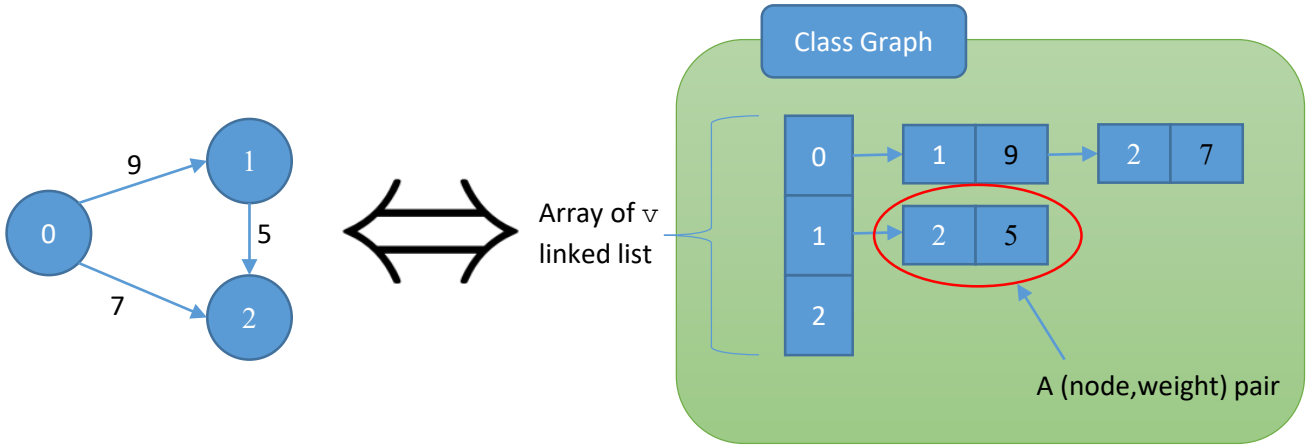
What is the shortest distance from Node 2 to Node 4?

Skeleton Code

- The `Graph` Class
  - `Graph.{h,cpp}`
- The Linked List Skeleton in your Assignment 1 with iterator
  - `simpleLinkedListTemplate.{h,cpp}`
- The driver code to read in the input file and start your computations.
  - `main.cpp`

The Graph Class

- We use adjacency lists as the data structure for our graphs. For a graph with `v` nodes, we will have `v` linked lists and each node `a` will have a list containing the neighbors of `a`, together with the weight of the edges. Please see the following diagram as an example:



The Graph Class

- Please read and understand the functions of the two classes `Graph` and `NodeWeightPair`. For example, the member function `printGraph()` in the `Graph` class will print out the graph for you:

```
Node 0:  (3,6) (2,2) (1,5)
Node 1:
Node 2:  (4,5) (3,1)
Node 3:
Node 4:
Node 5:
```

The Linked List Class with Iterators (Given)

- The files `simpleLinkedListTemplate.{h,cpp}` ***are*** the skeleton code of our Assignment 1. However, there are a few additional functions for iterating through a linked list. An example is already in `Graph.cpp` inside the function `printGraph()`

```
for (_al[i].start(); !_al[i].end(); _al[i].next())
  cout << " (" << _al[i].current().nodeIndex() << "," << _al[i].current().weight() << ")";
```

- `start()` will place a pointer to point to the first item in the Linked List
- `next()` will move the pointer to the next item
- `end()` will return `1` if the current pointer is null, namely, the end of the list, otherwise, return 0.
- `current()` will return the item that the pointer is pointing to currently.

Your Tasks

Task 1 Compute the Shortest Distance for Each Query

- Implement the function `shortestDistance(int s, int d)` in the class `Graph` that will return the shortest distance from the node `s` to the node `d`. If there is no path from `s` to `d`, return -1 instead. For the input file "`example3.txt`", your output should be like this:

```
The shortest distance from the vertex 0 to vertex 1 is 5
The shortest distance from the vertex 0 to vertex 2 is 2
Node 1 and Node 3 are not connected in the same component.
The shortest distance from the vertex 2 to vertex 4 is 5
Node 2 and Node 5 are not connected in the same component.
```

- In this task, you probably need the priority queue. You can add into your project. But please use the same conventions in the last assignment for the heap. Please bear in mind that you are not allowed to use STL or other code that is not from you.

Task 2 Printing the Path

- Print the path inside your function `shortestDistance()` *if* you can find a path. Your output should be like this for “example4.txt”

```
Node 0:  (2,7) (1,1)
Node 1:  (5,15) (3,9)
Node 2:  (4,4)
Node 3:  (5,5) (4,10)
Node 4:  (5,3)
Node 5:
Path: 0 1
The shortest distance from the vertex 0 to vertex 1 is 1
Path: 0 2
The shortest distance from the vertex 0 to vertex 2 is 7
Path: 0 1 3
The shortest distance from the vertex 0 to vertex 3 is 10
Path: 0 2 4
The shortest distance from the vertex 0 to vertex 4 is 11
Node 4 and Node 1 are not connected in the same component.
Path: 0 2 4 5
The shortest distance from the vertex 0 to vertex 5 is 14
```

Task 3 Problem Solving: Train Trips

- Use your code to solve the following problem by creating an input file without modifying your code in Tasks 1 and 2. There are n cities and you can travel from each city to another by trains. Let's assume that
  - In every city, the trains are operating 24 hours.
  - Only some pairs of the cities have trains between them. And luckily, when there is a connection, trains run in both ways, aka, bidirectional. Also, each connection has a different speed limit for the trains
  - Every train will leave a city at every hour. Namely, there will be a train leaving at 12:00, 1:00, 2:00, 3:00, and so on.
  - You are provided with the information of how long the distance is between two cities, and the speed limit of the trains between the two cities, see the table below.
  - You can assume every train will gain its speed to its maximum speed, and brake, in no time.
  - And you can also transfer from train to train in no time.

Here is the distance and maximum speed for the trains between 8 cities. The **first number** in each cell is the distance between the two places in km. The second number is the speed limit in km per hour. Note that the matrix is symmetric and we omit the other half. Namely, the entry with Cities 0 to 1 and 1 to 0 should be the same as (200,40).

Cities	0	1	2	3	4	5	6	7
0		200,40		150,30	100,25			
1			80,40			200,100		
2					160,40	90,45	240,60	
3								
4							300,60	
5							300,30	360,60
6								270,30
7								

Task 3: Problem Solving: Train Trips

- Create an input file that can use your code to answer the following question:

What is the shortest time (in hours) and the path to go from City 0 to City 7?

Submission

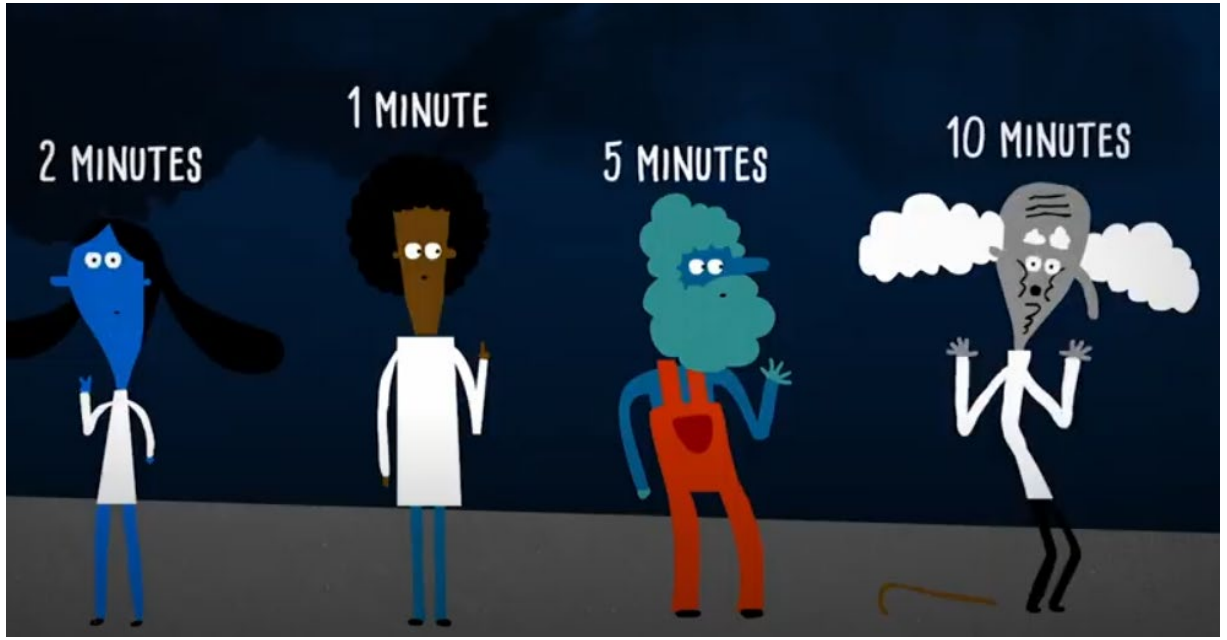
- **Tasks 1 and 2**:
  - You should submit the function `Graph::shortestDistance(int s, int d)` ***only***. And it also means that you should not modify other parts of the given skeleton files. You do not need to submit the heap code.
- **Task 3**:
  - You should submit the input file as "`train.txt`" that will help you to compute the answer of Task 3.

Reading in a File

- For **Mac** users, you can click the “Products” folder in your XCode project.
  - And on the right, you will find the “Full Path”.
  - After clicking the little right grey arrow on the right, the location/folder of the executable will be opened.
  - And you can copy your .txt file to this folder to be opened by your program.
- For **Windows** users, you may copy the file into your executable folder like Mac users.
  - My suggestion is to place the .txt files at the same folder with your project file.

Final Challenge

- Not graded, and no need to submit.
- Try to use your code to solve the Bridge Riddle mentioned in our labs (https://youtu.be/7yDmGnA8Hw0). Your answer should be something like this




```
Path: 0
The shortest distance from the vertex 0 to vertex 4 is 17
```