

Objectives: To understand how a graph can be used to solve graph algorithms.

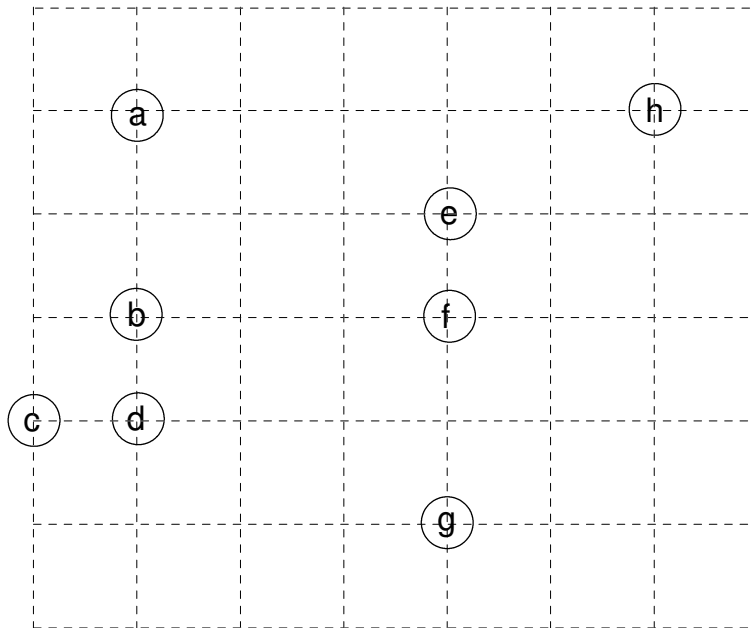
To start the lab: Download and unzip the file lab12.zip

Part A: In IDLE open the binary heap class file: `lab12/binheap.py`. The changes discussed in lecture are included:

- `PriorityQueueEntry` class, and
 - additional `BinHeap` methods: `__contains__` and `decreaseKey`.
- a) Look at the `buildHeap` method. Explain how it takes a list of values and builds them into a heap.

b) Run the `lab12/make_min_spanning_tree.py` program which uses Prim's algorithm on the graph from lecture. Does it give the expected output?

c) Predict the order of edges added by Prim's algorithm if we start at vertex "e":



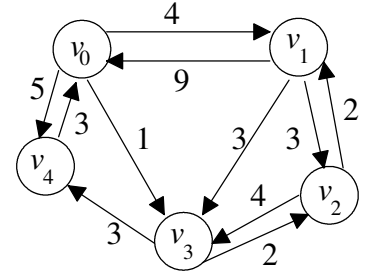
d) Modify the `lab12/make_min_spanning_tree.py` program to verify your prediction.

After you have answered the above questions and completed the code, raise your hand and explain your answers.

Part B: The textbook's Dijkstra's Algorithm code (Listing 7.11 p. 341) needs to be updated similarly to Prim's algorithm.

a) Modify the `dijkstra` method in the `lab12/graph_algorithms.py` file similar to Prim's so that it uses `PriorityQueueEntry` objects among other corrections.

b) Run the `lab12/test_dijkstra.py` program which uses Dijkstra's algorithm on the graph from lecture. Does it give the expected output?



After you have fixed the `dijkstra` method in `lab12/graph_algorithms.py`, raise your hand and demonstrate your code.

EXTRA CREDIT:

Add code to the end of the `test_dijkstra.py` program to print the shortest paths from v_0 to each of the other vertices. One line of output might look something like:

“Shortest path from v_0 to v_4 is $v_0 > v_3 > v_4$ with a total distance of 4”