Competitive Programming

Graph Representation

and

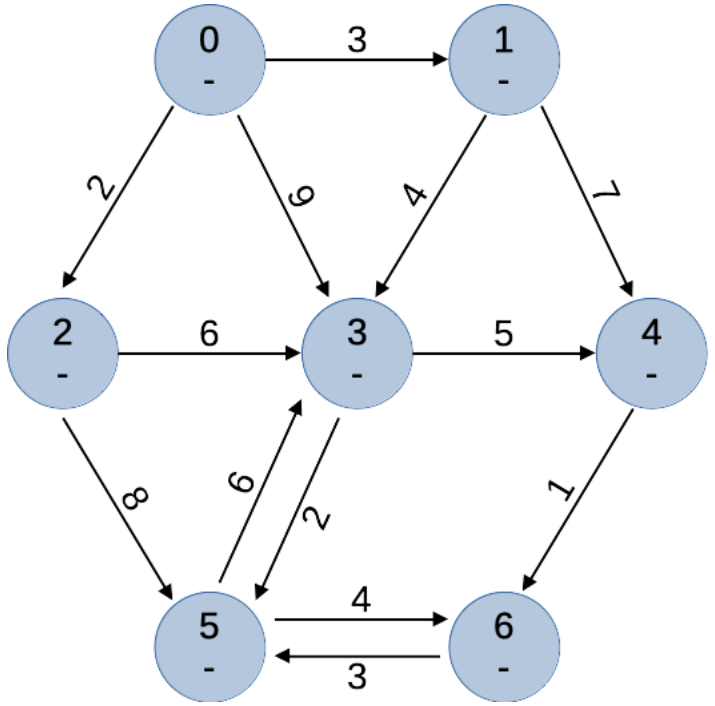
Dijkstra’s Algorithm

Graph Problems

- There are lots of standard algorithms for working with graphs.
- Lots of problems in competitive program depend on applying one or more of these techniques.
- You need to be able to quickly read in an work with graphs
- There are a few common representations.

Adjacency Matrix

- Build a square 2D array
- Each element representing a possible edge between vertices.
- Quick for checking a pair of vertices.
- Expensive for large, sparse graphs.



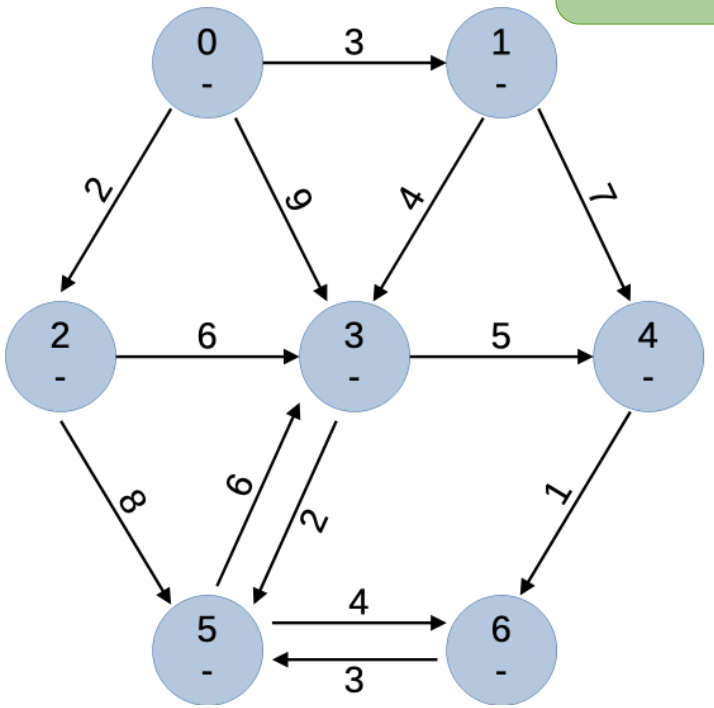
	0	1	2	3	4	5	6
0		3	2	9			
1				4	7		
2				6		8	
3					5	2	
4							1
5				6			4
6						3	

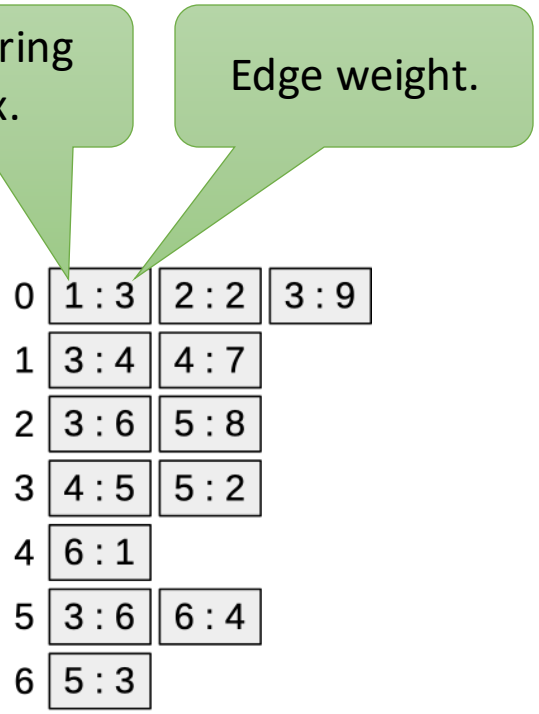
Maybe a nonsense value to indicate no edge.

	0	1	2	3	4	5	6
0	-1	3	2	9	-1	-1	-1
1	-1	-1	-1	4	7	-1	-1
2	-1	-1	-1	6	-1	8	-1
3	-1	-1	-1	-1	5	2	-1
4	-1	-1	-1	-1	-1	-1	1
5	-1	-1	-1	6	-1	-1	4
6	-1	-1	-1	-1	-1	3	-1

Adjacency List

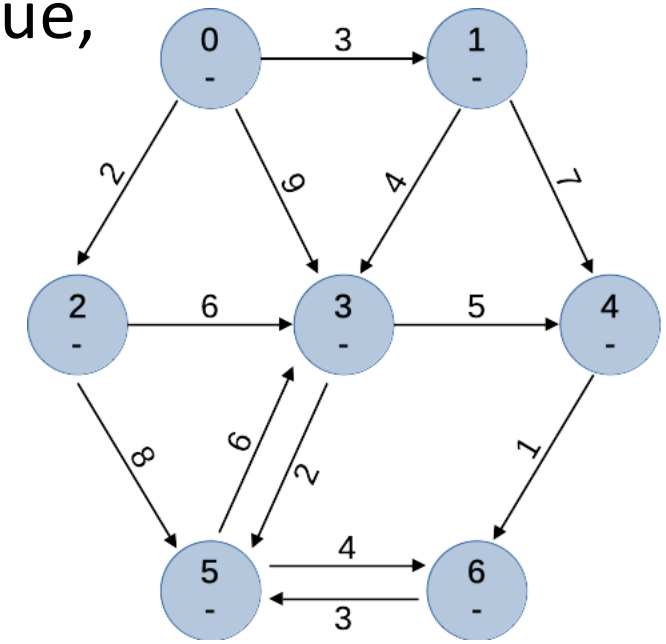
- Array or lists, one for each vertex.
- … listing neighbors of each vertex
  … maybe along with edge weight.
- Memory linear in the size of the graph.
- You want this to be really easy to code.

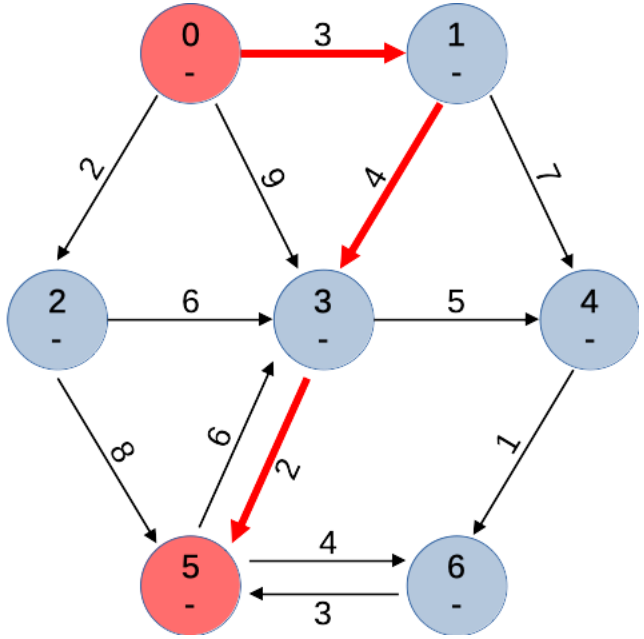




Shortest Path Problem

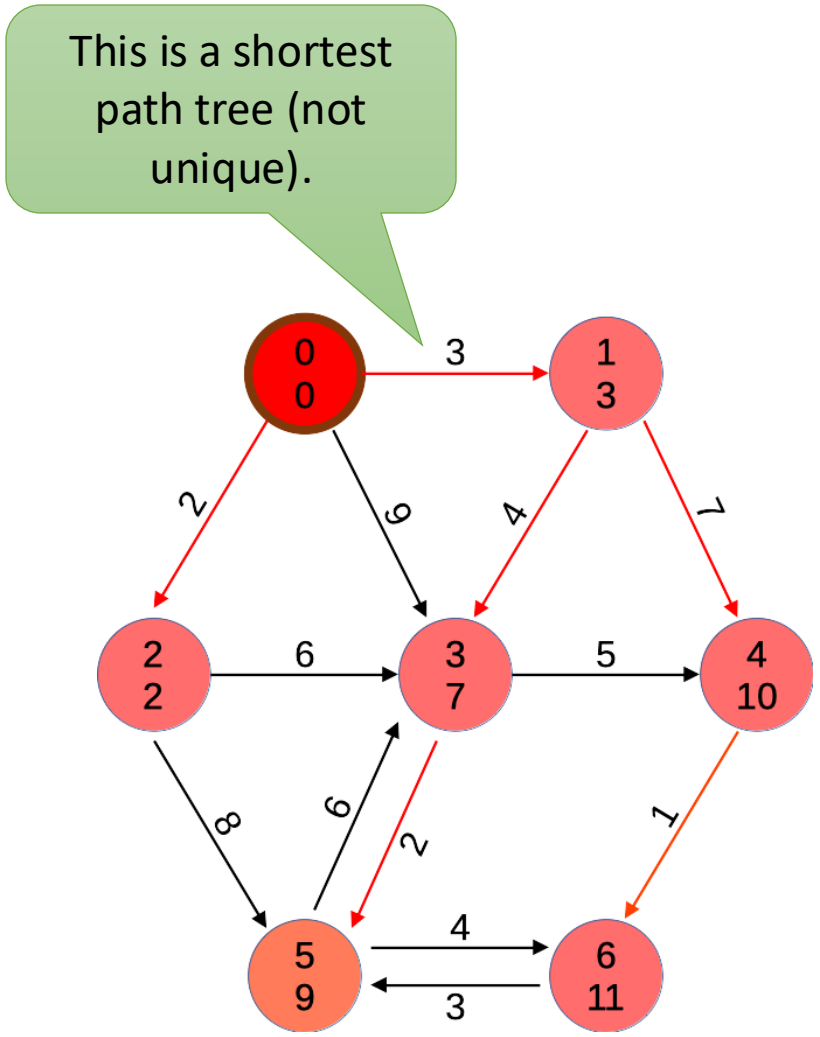
- Given a weighted (directed) graph.
- What's the shortest path between a pair of vertices.
- The path may not be unique,
- … but its length is.



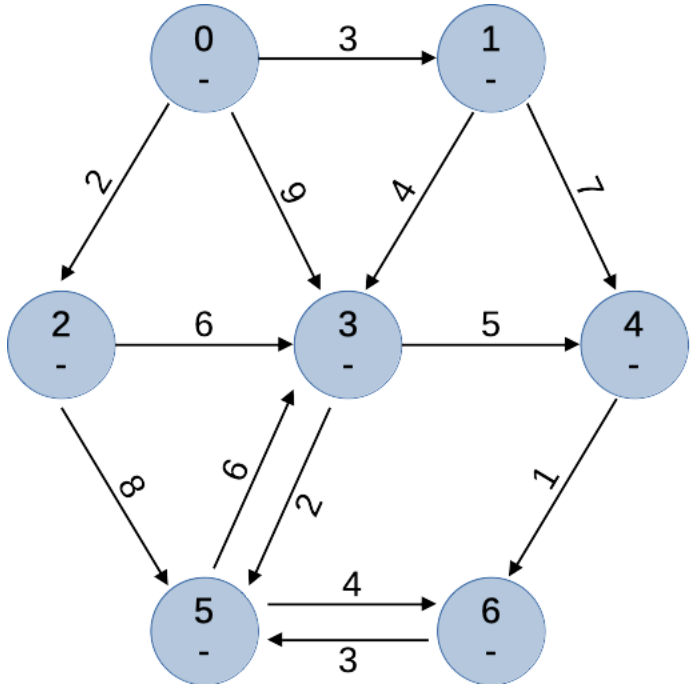


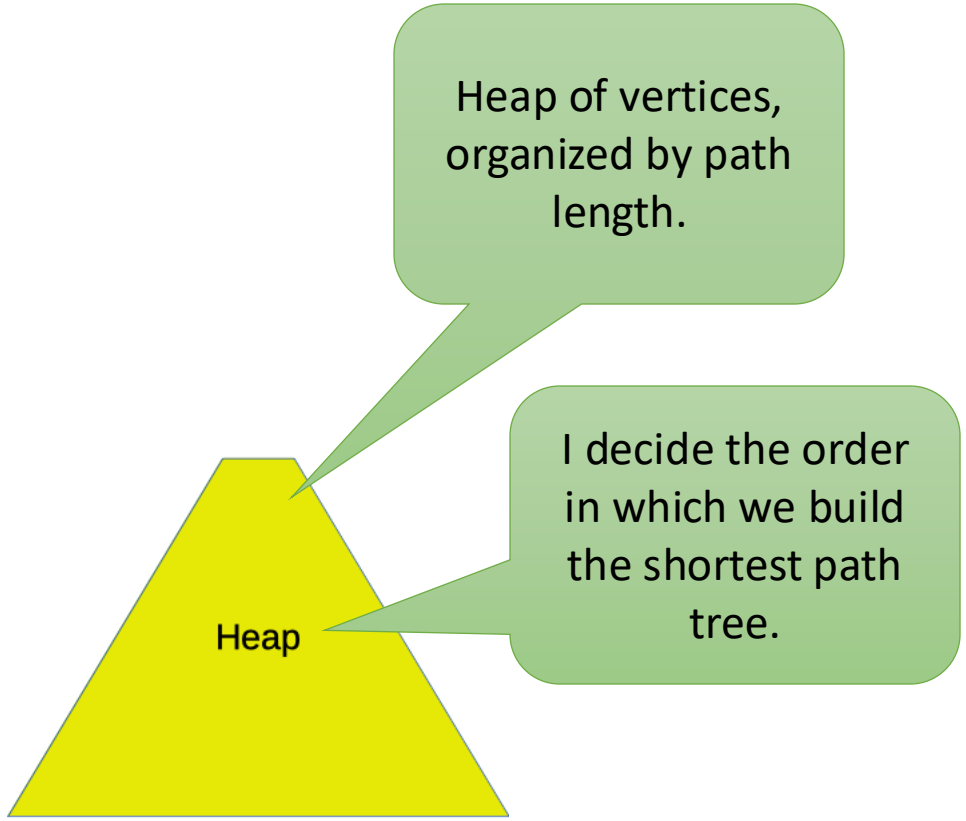
(Single Source) Shortest Path Problem

- Commonly solved by finding **all** shortest paths from a source vertex.
- That's what Dijkstra's algorithm does.
- O(n log(n)) for an n-vertex graph
  really, more like O(E log(V)) or O(E log(E)) depending on how it's implemented.

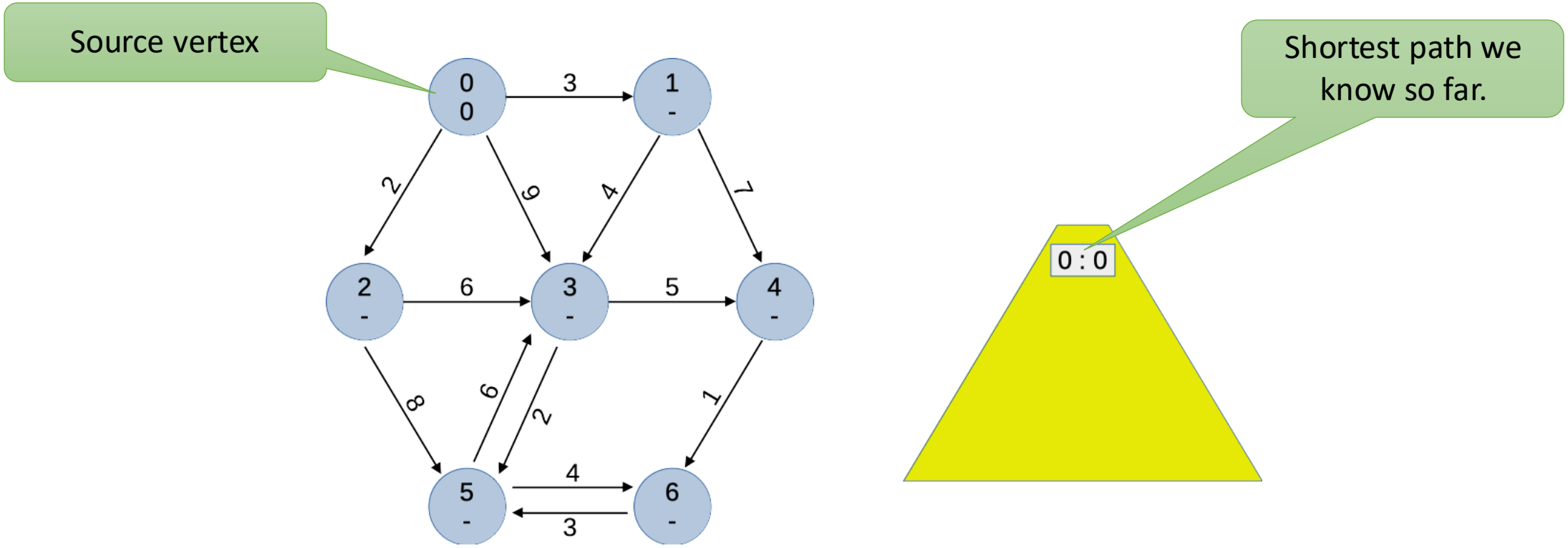


Dijkstra's Algorithm





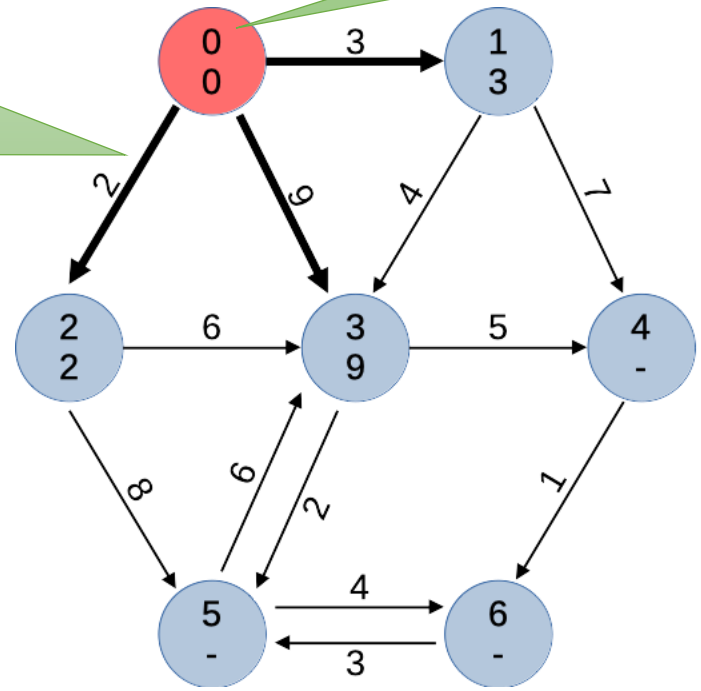
Dijkstra's Algorithm

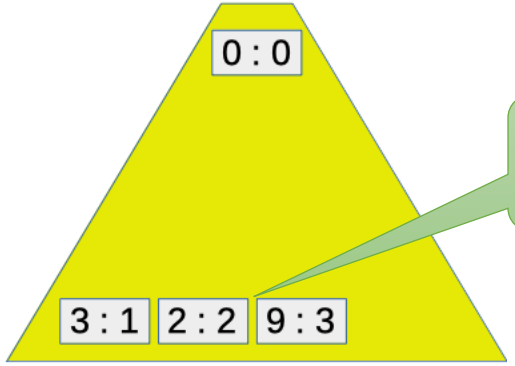


Dijkstra's Algorithm

Vertex 0 is done. We will never find a shorter path to here.

Update shortest paths to neighbors (relax)

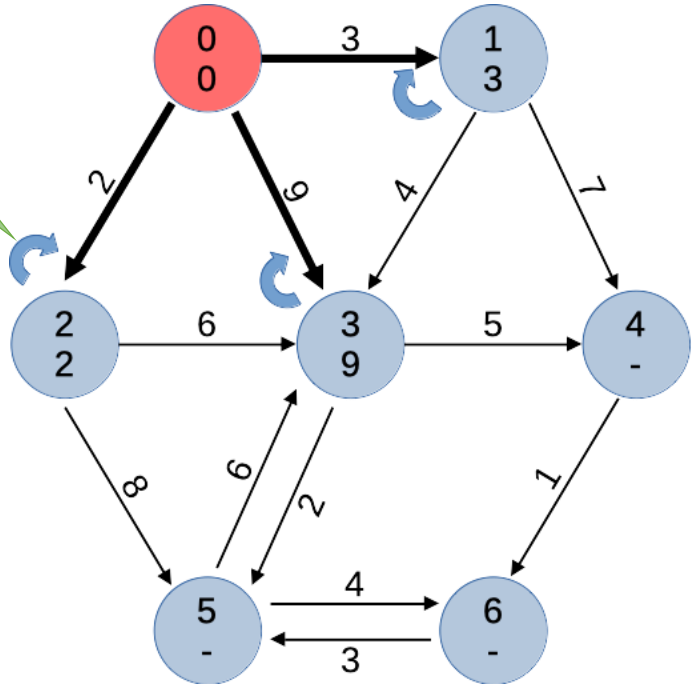


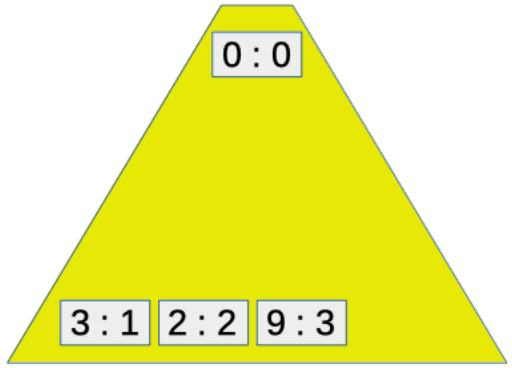


New paths in the heap.

Dijkstra's Algorithm

Each vertex remembers its predecessor.

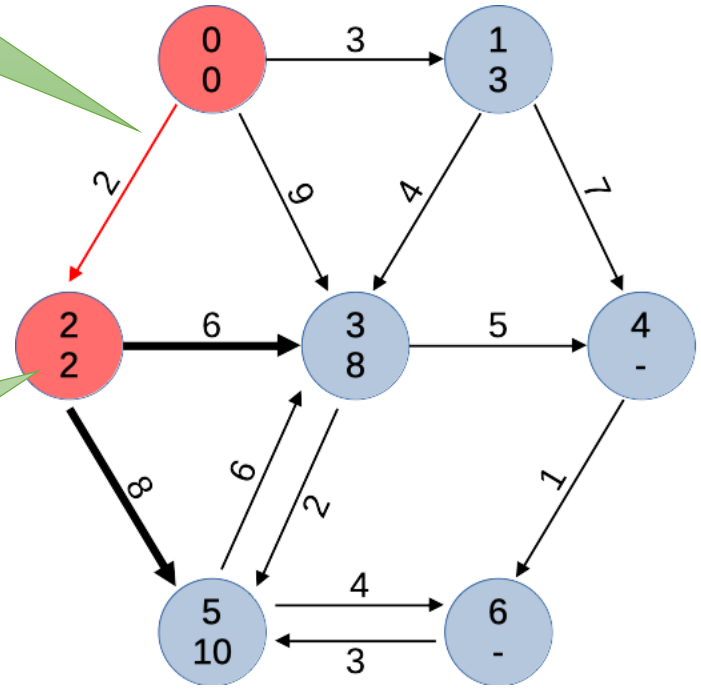




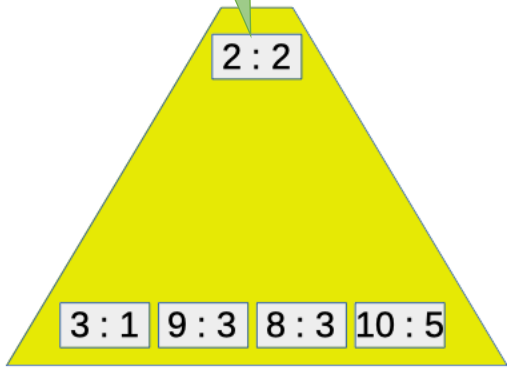
Dijkstra's Algorithm

We're selecting edges for a shortest-path tree.

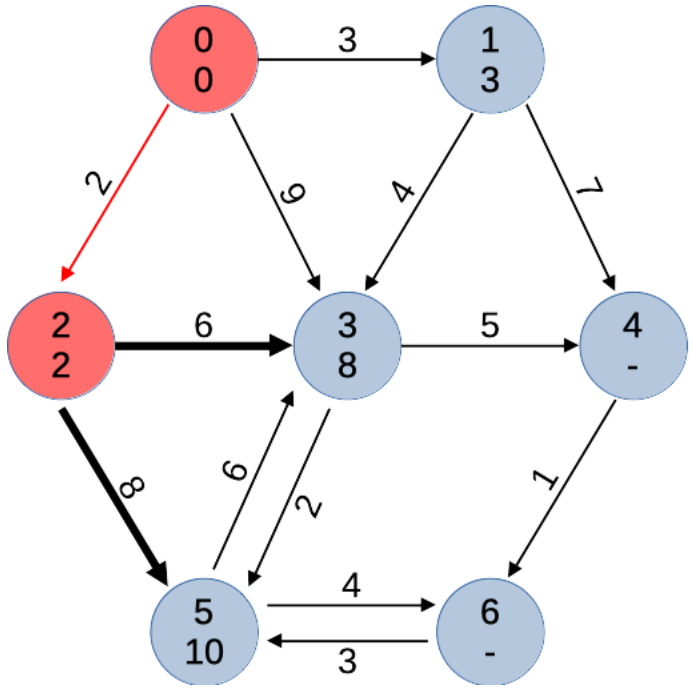
We will never find a shortest path to vertex 2.

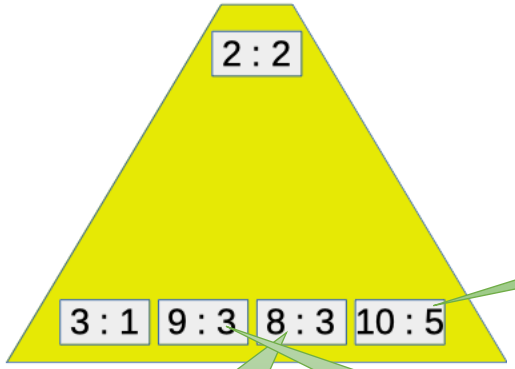


Shortest remaining path.



Dijkstra's Algorithm



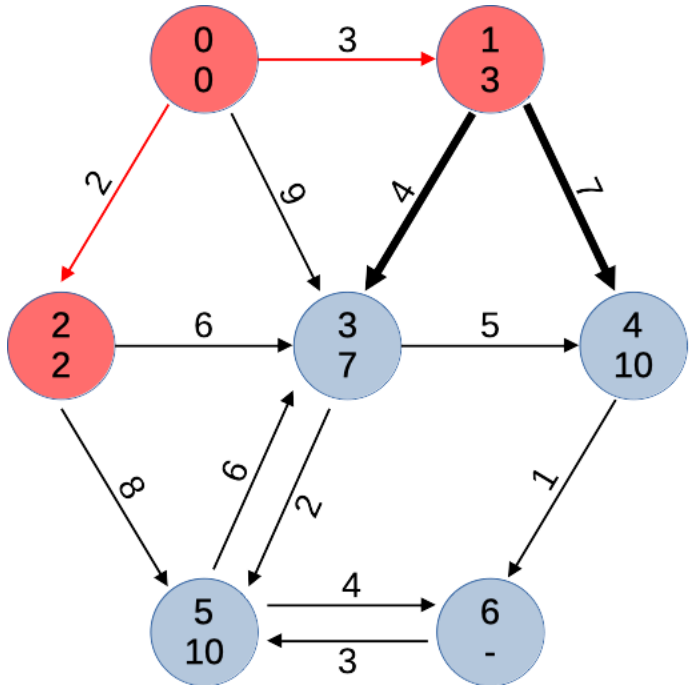


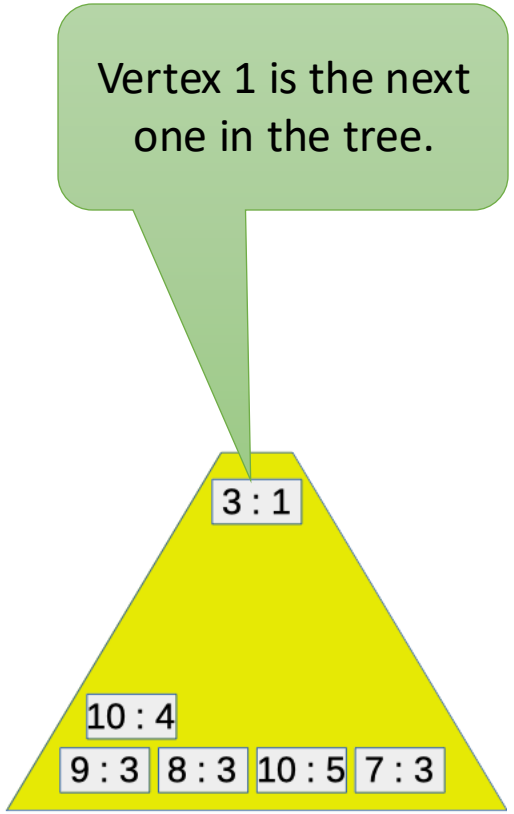
Neighbors of vertex 2 go into the heap.

Including some improved paths.

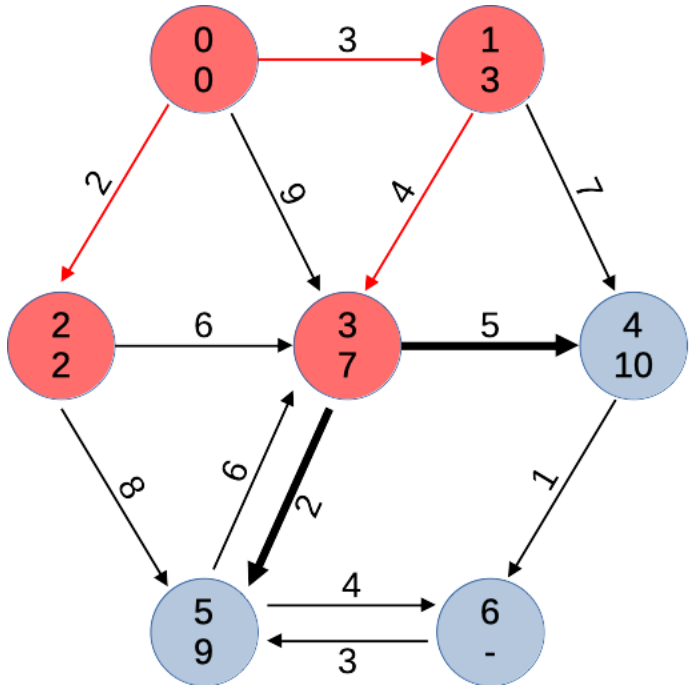
Would be better (but harder) to update this entry.

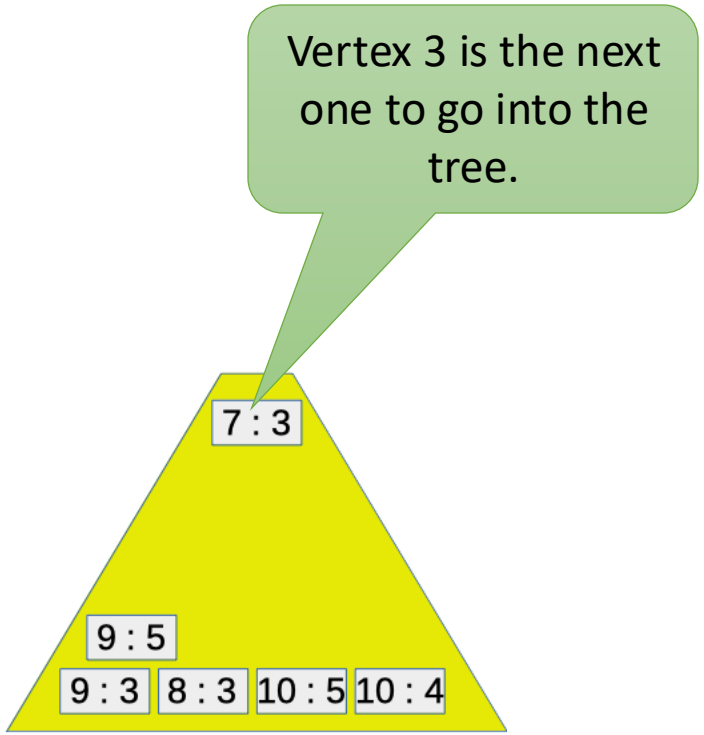
Dijkstra's Algorithm



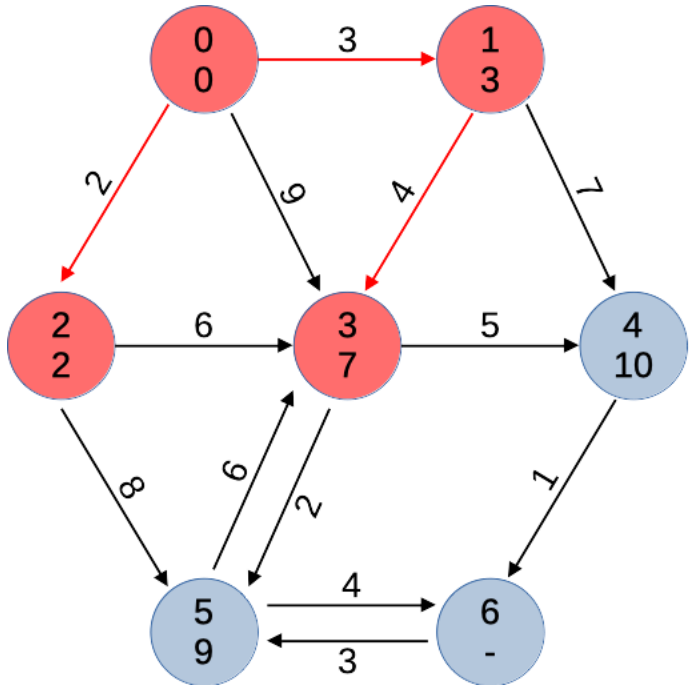


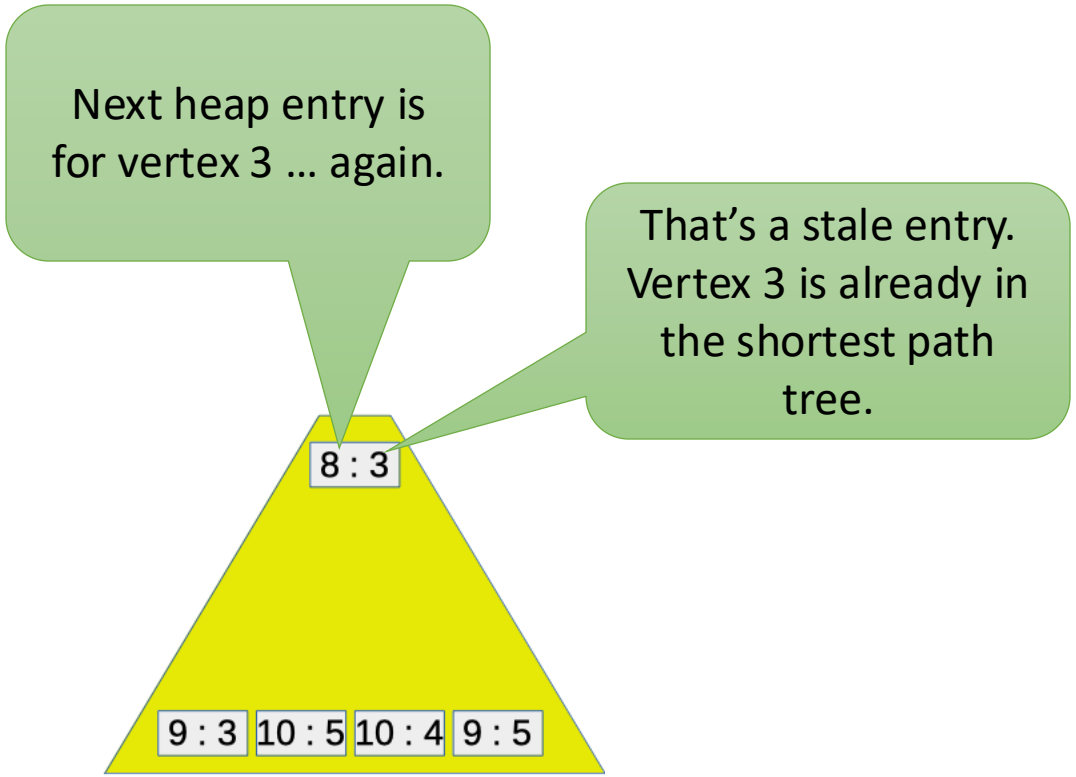
Dijkstra's Algorithm



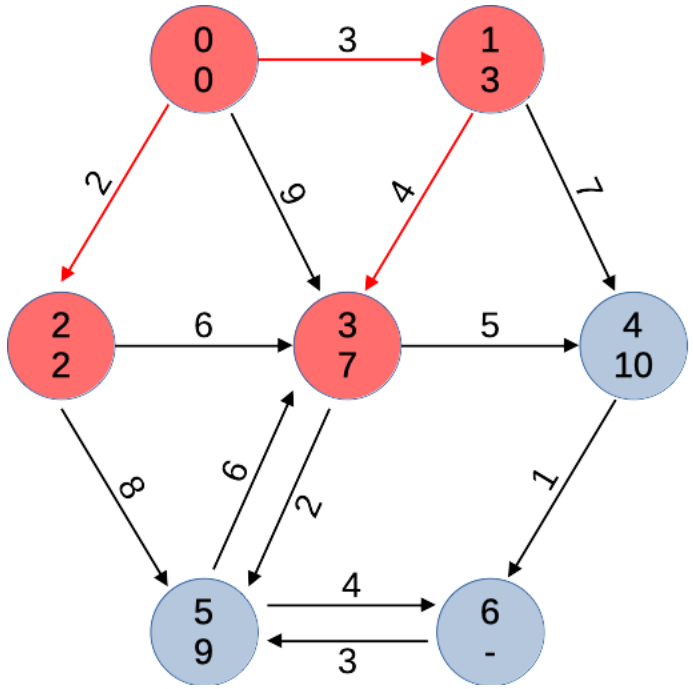


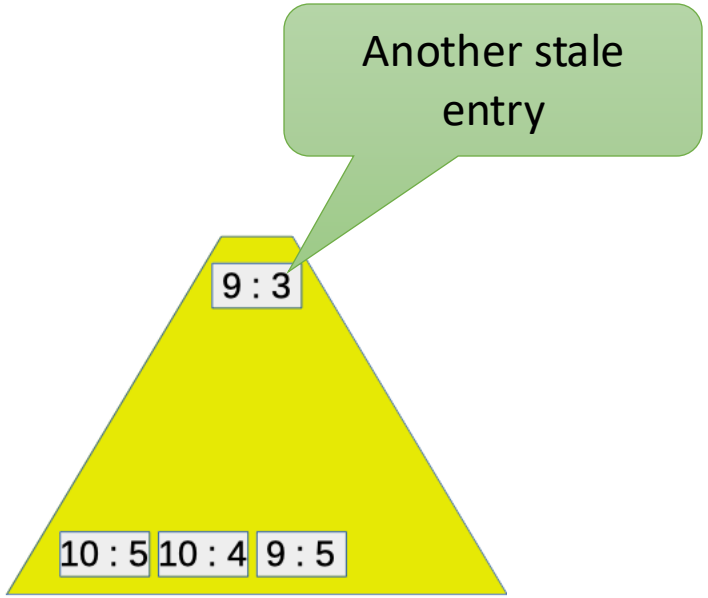
Dijkstra's Algorithm



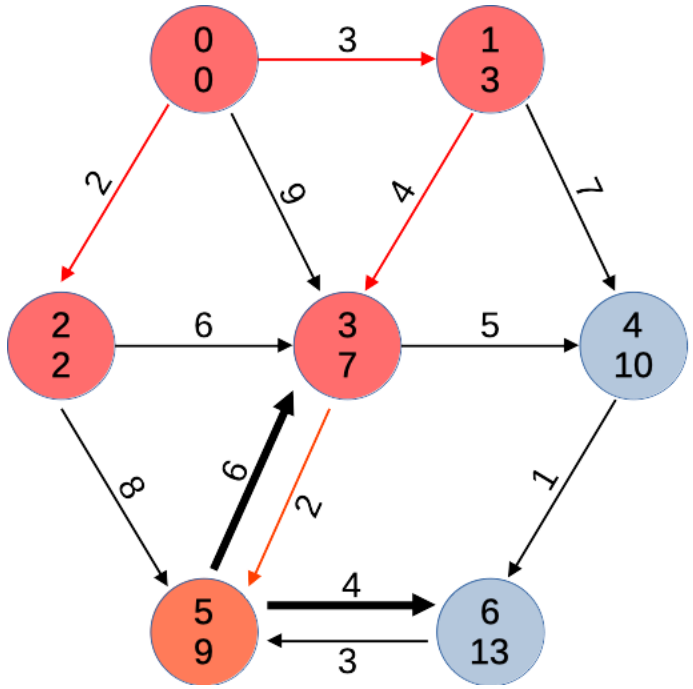


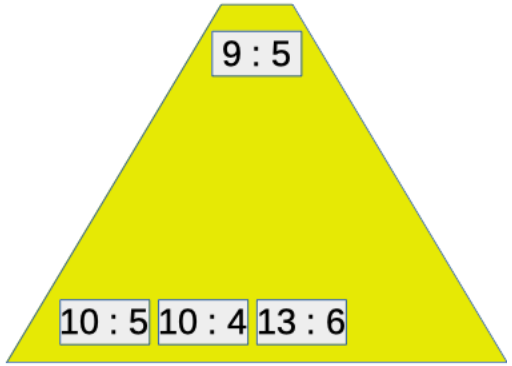
Dijkstra's Algorithm



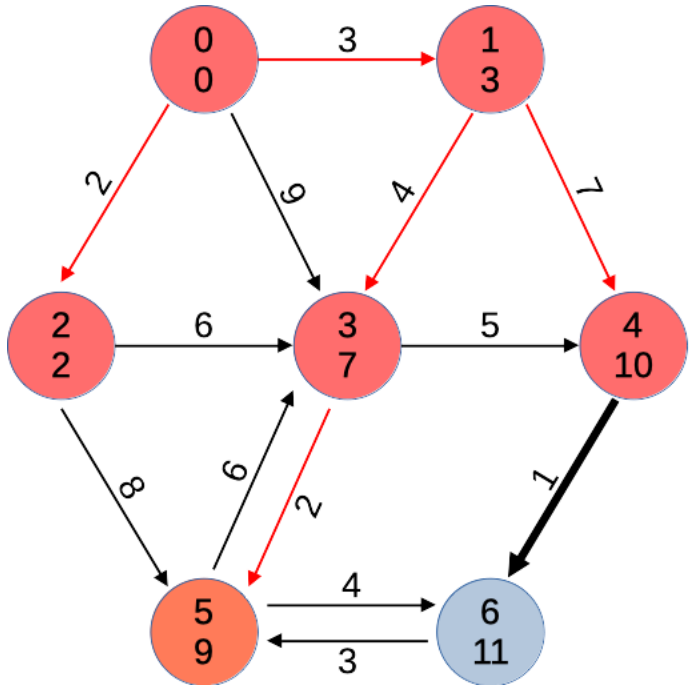


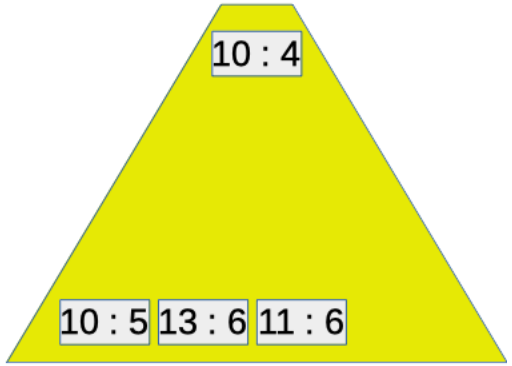
Dijkstra’s Algorithm



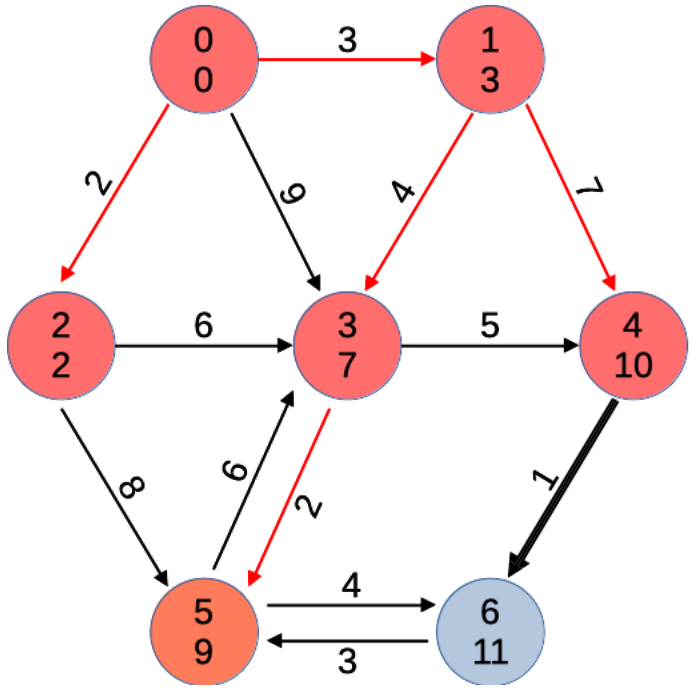


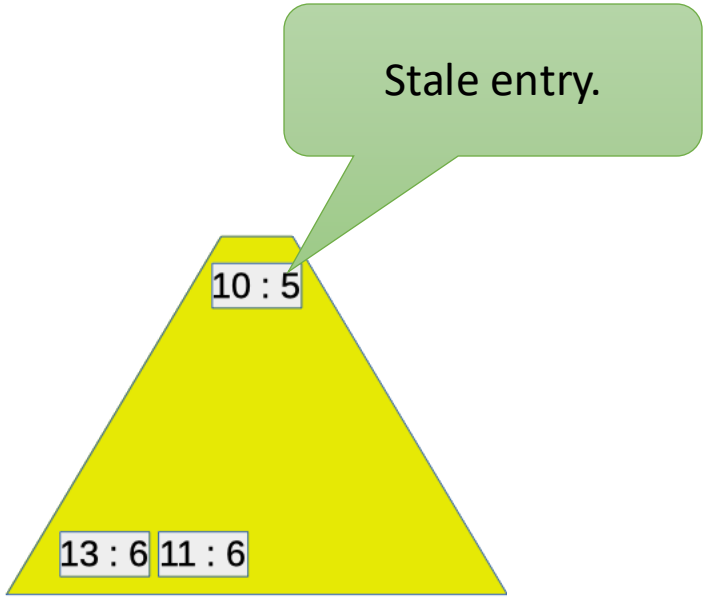
Dijkstra’s Algorithm





Dijkstra’s Algorithm





Dijkstra’s Algorithm