**CS330 – Single Source Shortest Path in a Graph**

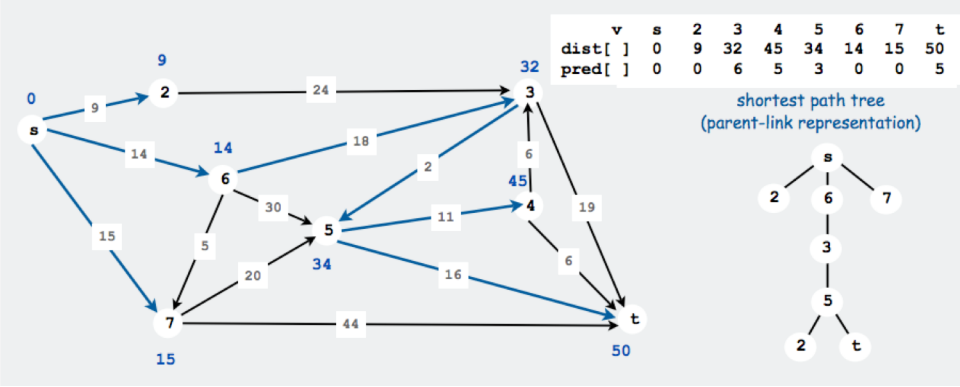
**Purpose:** Given a weighted graph we would like to find the distance and shortest path in the graph from a starting vertex to each of the other vertices in an efficient way.

**Knowledge:** This activity will help you become familiar with the following content knowledge:

- Dijkstra's algorithm and its efficiency

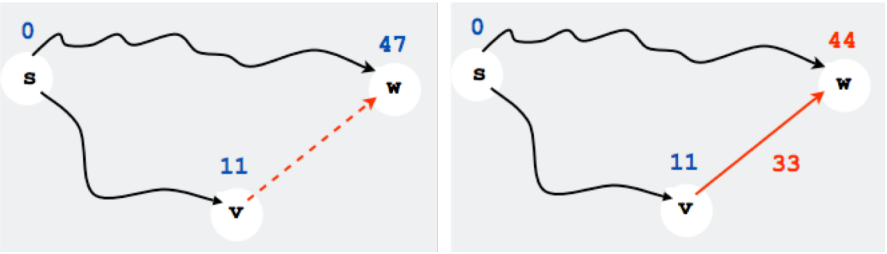
**Activity:** With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 60 minutes.

1. We want to find the distance and shortest path from vertex $s$ to every other vertex in a weighted graph.




   Why must we assume that the graph has no cycles with a negative weight?

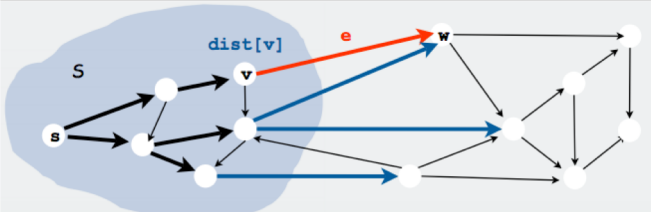
2. For all vertices $v$, let $dist[v]$ be the length of some path from $s$ to $v$. Let $pred[v]$ be the predecessor in that known path from $s$ to $v$. Relaxation along edge $e$ from $v$ to $w$ sets $dist[w]$ to the length of a shorter path from $s$ to $w$ if that edge gives one. In the figure the edge gives a shorter path to $w$ than the path cost of 47 which was previously known.




   Write the code which will change *dist* and *pred* if needed when relaxing the edge from $v$ to $w$. You may assume that `e.weight()` will return the weight of edge $e$.

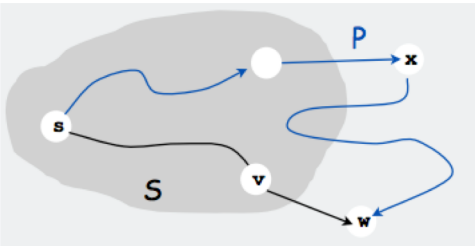
3. An algorithm to solve this problem is Dijkstra's algorithm. In this algorithm we maintain $S$, a set of vertices for which the shortest path length from $s$ is known:

    Initialize $S$ to $s$, $dist[s]$ to 0, and $dist[v]$ to $\infty$ for all other $v$
    Repeat until $S$ contains all vertices connected to $s$:
        find $e$ with $v$ in $S$ and $w$ not in $S$ that minimizes $dist[v] + e.weight()$
        relax along that edge
        add $w$ to $S$




    Why do you think $w$ can be added to $S$ in that step in the algorithm. Remember $S$ is the set of vertices for which the shortest path is known.

4. Dijkstra's algorithm can be proved correct with an induction proof on the size of $S$. Let $w$ be the next vertex added to $S$. Let $P^*$ be the path from $s$ to $w$ through the vertex $v$. Consider any other path $P$ from $s$ to $w$, and let $x$ be the first node on the path outside of $S$.




    Why is the path $P^*$ shorter than the path $P$?

5. When we implement Dijkstra, what data structure should we use to store $dist$?
    Hint: We need to easily extract the minimum value.

    What would be the cost of extracting the minimum edge with that data structure?

    How many times is each edge examined (and relaxed)?

    When we relax an edge we need to adjust the data structure. What the cost of this?

    Putting it all together, what is the order of growth for Dijkstra's algorithm?

6. Finally, just a bit of insight into graph search algorithms. All our graph-search methods are the same algorithm! We maintain a set of explored vertices $S$ and grow $S$ by exploring edges with exactly one endpoint leaving $S$.

   - DFS takes edge from vertex which was discovered most recently
   - BFS takes edge from vertex which was discovered least recently
   - Prim takes edge of minimum weight
   - Dijkstra takes edge to vertex that is closest to $s$.

   Bask in the glory of graph search!