HW11 - Written Questions

Prasun Surana
CS5008, Summer 1

1. The space complexity for our implementation of Dijkstra's Algorithm is O(V). This is because we store each node in the priority queue, so we require as much space as there are nodes in our graph.

2. The time complexity of Dijkstra's Algorithm depends on the implementation. If Dijkstra's Algorithm is implemented using a binary min-heap, the complexity is O((V + E)log(V)), since the search can be more efficiently using a tree. For our array implementation, the time complexity is $O(V^2)$. This is because first we conduct a linear search in the priority queue, which takes O(V) time. We do this O(V) times while the priority queue is not empty, and we also have to check each min vertex's edges, which gives us $O(V^2 + E)$, and this can be simplified to $O(V^2)$.

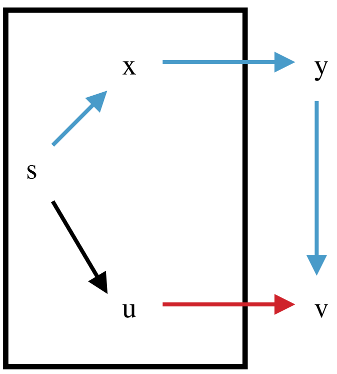
3.

We want to prove that given a set A of nodes that are explored by Dijkstra's Algorithm, for each $u \in A$, the path found to u is the shortest path.

Base Case: When |A| = 1, the only explored node s (source). The distance of the source is 0.

Inductive Hypothesis: Assume the theorem above is true for $|A| \leq k$.

Let v be the $k+1^{th}$ node accessed by edge (u,v).



A

Let v by the first vertex added outside A, where A is the set of already explored nodes, and y be another node outside T that also leads to vertex v.

The paths from s -> x and s -> u are both assumed correct by our inductive hypothesis. The algorithm picks the shortest path for each additional node. The algorithm has chosen so go from (s,u) and then to (u,v). It could have also chosen s->x and then x->y, but since the algorithm picks the minimum cost path, and has already chosen s->u->v, this means that s->x->y is already longer than s->u->v. Hence, s->u->v is the shortest path possible.