Tutorial 7 – Dijkstra

Find all shortest paths from a start vertex s to every other vertex.

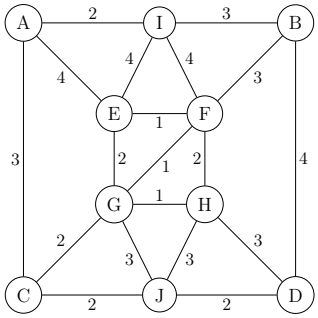
Dijkstra's algorithm finds the shortest paths by something similar to *breadth-first search*, but with a twist:

The *queue* is changed to a *min priority queue*.

The algorithm *grows a distance tree* T one edge at a time.

Priority of vertex v = *least weight path* between v and s so far. (∞ if no such path yet.)

Perform Dijkstra's algorithm on the following graph - choose B as your start vertex:



Make sure you include for each vertex `v` the distance `d[v]` of `v` from B as well as `p[v]` the parent of `v` in the distance tree.

```
dijkstra(G, s)
   PQ := new min-heap()
   PQ.insert(s, 0)
   d[s] := 0
   for each vertex z ≠ s:
      # initialize priority queue
      PQ.insert(z, ∞)
      d[z] := ∞
   while PQ not empty:
      #greedy choice of vertex to grow shortest path tree
      v := Q.extract-min()
      for each u in v's adjacency list:
         #Update priorities of adjacent nodes
         if d[v] + w({v,u}) < d[u]:
            PQ.decrease-priority(u, d[v] + w({v,u}))
            d[u] := d[v] + w({v,u})
            pred[u]:= v
```

Dijkstra Correctness

Fill in the missing steps of the correctness of Dijkstra:

- Let T_s be the *distance tree* constructed by *Dijkstra's Algorithm* starting at s.

- Let O_s be an *optimal distance tree* rooted at s.
- Order the edges $\langle e_1, e_2, \ldots, e_m \rangle$ according to how they are added to T_s.
- Consider the first edge $e_i = (u, v)$ such that ...
- Then $e_1, \ldots, e_{i-1} \in T_s$. Let S be the set of vertices added so far (ie, all endpoints of $\langle e_1 \ldots e_{i-1} \rangle$).
- Each node in S has ...
- Since $(u, v) \notin O_s$ it must be that there exists ...
- Consider the edge $e_j = (x, y), j > i$ on p that has one endpoint in S and one in $V - S$.

Case 1: $y \neq v$, ...

Case 2a: If $y = v$ and $d_O[y] < d_T[v]$ ...

Case 2b: Therefore, $y = v$ and $d_O[y] = d_T[v]$, ...

⋆ One can also prove this by *induction*