Dj's Algorithm Notes

Youtube

find shortest path between any 2 vertices in a graph

generates list of vertexes, shortest distance from index to each one, previous vertex array lists each step taken for shortest path to dest

Packing List: index vertice of where to start destination vertice list of visited vertices list of unvisited vertices (initially all in there) shortest distance from index vertice to N vertex list previous vertex list - keeps track of the path

1. use a visited array and unvisited array. all start in unvisited
2. index to index is 0 and it is its own prev
3. go to each neighbor, A-B is 6 so 6+0 (A-a) = 6, A-D is 2 do 2+0=1 update shortest distancea (if smaller than shortest distance). previous vertice of each neighbor is A
4. then, visit neighbor of A that is smaller, so go to d
5. Ds neighbors are B and E, D-B is 2, d-E is 1. (A-D was 2) D-B would mean 2+2=4, D-E would be 2+1=3, so go to E next and update everything

Wikipedia

shortest path first

common variant fixes single node as source, and finds shortest path from source node to all other nodes in the graph - producing shortest path tree

could also just return shortest path one time

if use min-priority queue for data structure, run in O(Vertices + Edges)log(Vertices)

if use array O(Vertices)^2

How

initial node = starting node distance of node y distance from initial node to Y

1. mark all nodes as unvisited
2. assign every node tentative distance value - 0 for initial, inf for others
3. for current node, calculate all tentative distances for unvisited neighbors through the current node consider newly calculated tentative distance to current assigned value and ex: if current node A is marked with distance 6, and b has 2, distance to B through A will be 6+2=8. if B was previously marked with value >8 change it to 8. otherwise leave it
4. when done considering all unvisited neighbors of current node, mark current node as visited and remove from unvisited set
5. if desintation node has been marked visited OR if smallest tentative distance among nodes in unvisited set is infinity (so no connections) then stop

6. otherwise select unvisited node that is marked with smallest tentative distance, set as new current node and go back to step 3

u and v neighbor nodes

```
create vertex set Q

for each vertex v in Graph:
    dist[v] = INF
    prev[v] = undefined
    add v to Q

while Q is not empty:
    u <- vertex in Q with min dist[u]

remove u from Q

for each neighbor v of u:

//only v that are still in Q
alt <- dist[u] + length(u,v)

if alt < dist[v]:
    dist[v] <- alt
    prev[v] <- u

return dist[], prev[]
```