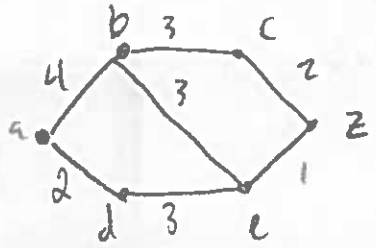
The question of finding a Hamilton cycle is hard, there is a corresponding NP-hard problem: The traveling salesman.

[TSP explanation]

Shortest path problems: Many problems deal with weighted graphs, graphs where each edge has an associated weight. This weight can be anything, cost to travel, travel time, distance etc.

Our goal will be to find an alg to find the shortest weight between vertices, we'll discuss a greedy alg. What is the shortest path between a & z?



$a \to d \to e \to z$

This demonstrates the alg of Dijkstra's Alg, always take shortest available path.

Idea: we have a graph of nodes & weighted edges. Choose starting node label it with weight 0 & every other node with ∞. We use $L_0(a) = 0$ $L_0(v) = \infty$

This is the "length to vertex v on iteration 0".

Let S_k be our set of distinguished vertices after k iterations $S_0 = \emptyset$

$S_k = S_{k-1} \cup \{u\}$ where u has the least distance.

once we update S_k we update $L_k(v)$ for each v

Dijkstra(G): # G = graph with nodes $v_0, \dots, v_n$ lengths $w(v_i, v_j)$
$w(v_i, v_j) = \infty$ if no edge.

S = vertex set = [ ]

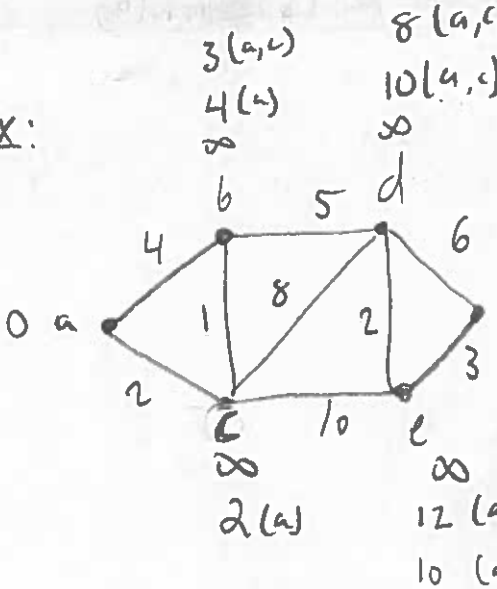
for i = 1 to n:
    $L(v_i) = \infty$ # set all nodes to ∞ away.

$L(a) = 0$

while $z \notin S$
    u = vertex with min dist to a
    S.append(u)
    foreach neighbor of u:
        tmp = $L(u) + w(u,v)$
        if tmp < $L(v)$
            $L(v) = tmp$

return $L(z)$

Ex:



$S = \emptyset$

$S = \{c\}$

$S = \{c, b\}$,

$S = \{c, b, d\}$

$S = \{c, b, d, e\}$

$S = \{c, b, d, e, z\}$

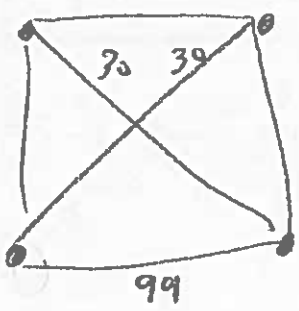
Theorem: Dijkstras finds the length of the shortest path between 2 vertices in weighted connected graph.

Theorem: Dijkstra uses $O(n^2)$ operations.

Notice differences from TSP, just a path, no guarantee of hitting every node.

Dijkstra can be baited into bad choices

Ex:



c
2 (a,b)
30 (a)

d
99 (a)
31 (a,b)
3 (a,b,c)

$S = \emptyset$

$S = \{a\}$

$S = \{a, b\}$

$S = \{a, b, c\}$

$S = \{a, b, c, d\}$

⇒ Path a, b, c, d, a $w(P) = 102$

or Path a, b, d, c, a $w(P) = 62$