0-1 Knapsack Addendum

Inner loop goes backward to prevent using an item more than once.

```
for (item)
    for (i = W; i >= w[item]; i--)
        dp[i] = max(dp[i], dp[i - w[item]] + v[item])
```

this hasn't been updated yet based on item.

What if I want multiple copies of an item

to allow

```
for (item)
    for (i = w[item]; i <= W; i++)
        dp[i] = max(dp[i], dp[i - w[item]] + v[item])
```

this was updated a minute ago! (or less)

Floyd-Warshall's Alg

All Pairs Shortest Path
Neg Edge Weights
Detect Neg Cycles

Input: Directed Weighted Graph

Build up shortest paths (between all pairs of vertices) as we allow intermediate vertices.

(1) Store adjacency matrix of our graph

adj[i][j] = edge weight from i to j.

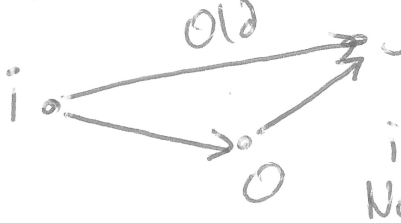
if no edge either store null of large #.
watch out for overflow!

This represents shortest distances when no intermediate vertices are allowed.

Let d[i][j] represent our estimate for the shortest distance from i to j.

Initially d[i][j] = adj[i][j].

Mom says vertex 0 is safe to cross



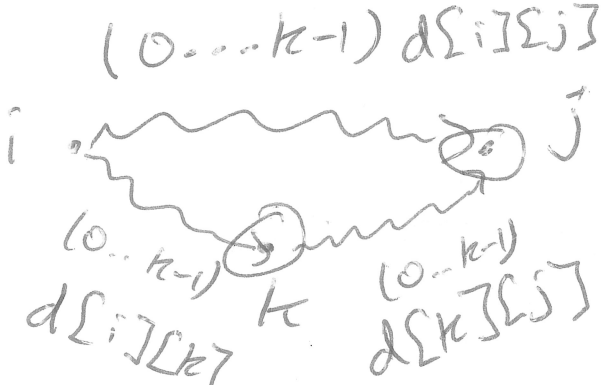
d[i][j] = min(d[i][j], d[i][0] + d[0][j])

i → 0 → j, using intersection 0

After this update for all i, j we'll shortest distances that ONLY go through vertex 0.

d[i][j] = min(d[i][j], d[i][0] + d[0][j])

More generally, once vertex k is deemed to be safe, we went to see if using it helps:



d[i][j] = min (d[i][j], d[i][k] + d[k][j])

// k intermediate vertex

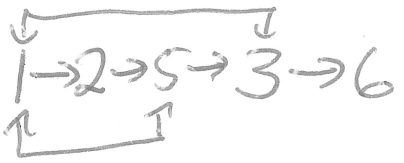
```
for (k=0; k<n; k++)
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            d[i][j] = min (d[i][j], d[i][k]+d[k][j]);
```

i = start
j = end

How to reconstruct the shortest path?

path[i][j] = last vertex on the shortest path from i to j

path[1][6] = 3
path[1][3] = 5
path[1][5] = 2
path[1][2] = 1

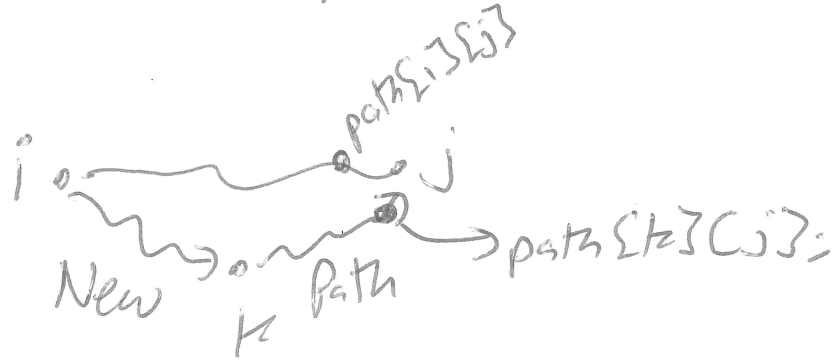


← How to use

How to build?

Initially path[i][j] = i for each edge that exists.
−1 otherwise

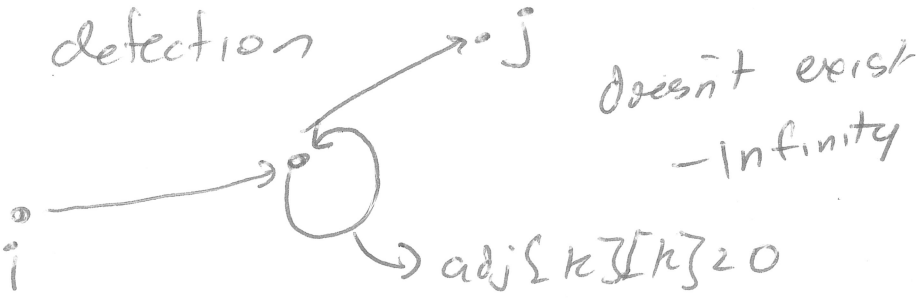
```
if (d[i][k] + d[k][j] < d[i][j]) {
    d[i][j] = d[i][k] + d[k][j];
    path[i][j] = path[k][j]; //NOT k!!!
}
```



Runtime $O(V^3)$, max V in practice ~500. Memory $O(V^2)$.

Neg cycle detection

1000 queues
x 150 k



Doesn't exist
-infinity