Minimum cost walk by dynamic programming

Find minimum cost walk in presence of negative cost edges (but no negative cost cycles).

Note: if there is a negative cost cycle that can be inserted into the walk from start to end, then there is no minimum cost walk - by repeating the cycle, we can obtain walks of cost as small as we want

s = starting vertex, t = ending (target) vertex.

Distances in the graph

Define $d(x,y)$= the cost of the minimum cost walk from x to y, or ∞ if y is inaccessible from x.

Note that there is always a path achieving $d(x,y)$ (if y is accessible from x at all).

Minimum cost walks by length

Define $w_{k,x}$ = the cost of minimum cost walk of length at most k from s to x, or ∞ if no such walk exists.

We have a recurrence relation:

- $w_{0,s}=0$;
- $w_{0,x}=\infty$, for $x\neq s$;
- $w_{k+1,x}=\min(w_{k,x}, \min_{y\in N^{in}(x)}(w_{k,y}+c(y,x)))$;

Based on the recurrence relation above, we can easily compute $w_{k,x}$ for any vertex x and for any natural number k. We compute w row by row (in increasing order of k).

Since the minimum cost is always achieved by a path, $d_{n-1,t}$ gives the minimum cost from s to t.

To retrieve the path, we go back from t, reconstructing how we achieved each value of w.