Dynamic Programming Recipe

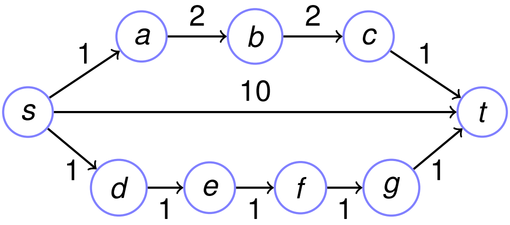
- Define a set of problems, such that
  - base case - easy to solve
  - final case - matches (closely) the final problem we want to solve.
- Write it as a recursion: Solve bigger problem in terms of the smaller problems. (Should be a DAG on the problem instances!)
- Compute the problems on the DAG in the linearized order!

Reliable Shortest Paths.

Shortest reliable path.

Given G, k. Find shortest path that uses at most k edges.

More edges, higher chance of a problem.



Shortest path? $s \to d \to e \to f \to g \to t$ Cost: 5

Shortest path that uses at most 4 edges? $s \to a \to b \to c \to t$ Cost: 6

Shortest path that uses at most 3 edges? $s \to t$ Cost: 10.

Algorithm

Dijkstra's?

Dynamic Program.

Subproblems: shortest path to node using few edges.

dist(v,i) - length of shortest path to v using i or fewer edges.

dist$(v,i) = \min_{(u,v)\in E}(dist(u,i-1)+l(u,v))$

dist(s,i) = 0 , dist$(v,0) = \infty$ for $v \neq s$

$O(nk)$ table entries. $O(|E|)$ time per "iteration".
Number of iterations? $k \implies$ time is $O(|E|k)$.

Is this familiar?

Bellman Ford...Dynamic Program!

All pairs shortest path.

$|V|$ single source shortest paths.

Bellman Ford takes $O(|V||E|)$ time.

$O(|V|(|V||E|)) = O(|V|^2|E|)$ time.

Can we do better?

Find $d(i,j)$ for all i, j.

"Bellman": $d(i,j,h)$ shortest path from i to j using h hops.

$d(i,j,h) = \min_{(j',j)\in E}\{d(i,j',h-1) + l(j',j)\}$.

$O(|V|)$ iterations, $O(|V||E|)$ per iteration. $O(|V|^2|E|)$

Really Bellman-Ford for $|V|$ sources!

Can we do better?

Floyd-Warshall

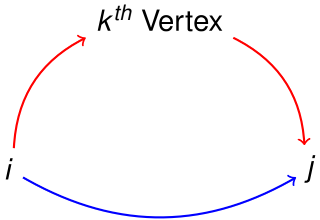
Remember Knapsack without Repetition.

Best knapsack using first i items.

$d(i,j,k)$ - shortest path from i to j using first k nodes on path.

For each edge $(i,j) \in E$, $d(i,j,0) = l(i,j)$.

$d(i,j,0) = \infty$ for $(i,j) \notin E$.



$$d(i,j,k) = \min(d(i,j,k-1), d(i,k,k-1) + d(k,j,k-1))$$

Runtime? ($n = |V|$)

For each edge $(i,j) \in E$, $d(i,j,0) = l(i,j)$.

Initialization time.

(A) $O(n^2)$

(B) $O(|E|)$

B. or A. depends...just doesn't matter!

Fill in table.

$d(i,j,k) = \min(d(i,j,k-1), d(i,k,k-1) + d(k,j,k-1))$

(A) $O(n^3)$

(B) $O(|E|^3)$

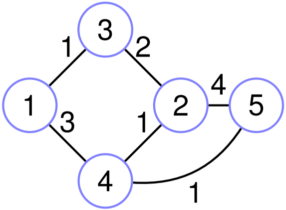
A. $O(n^3)$ table entries. $O(1)$ time per entry.

$O(n^3)$ time. (versus $O(n^2|E|)$ for n Bellman-Fords).

Travelling Salesman Problem.

Travelling Salesman Problem.

Given distances between n cities, find cycle that visits each city once of minimum cost.



try all orders and check. n! times n. Uh oh!

Can we do better?

Much better, but still not polynomial!

TSP Dynamic Program

Subproblem: best tour that visits the first i cities.

Visit cities in order?

W.l.o.g. - start tour at 1.

Subproblem: best tour that visits a subset S of cities...
....and ends at node j.

$C(S,j) = \min_{i \in S-j}\{C(S-j,i) + d_{ij}\}$

For all i we have $C(\{i\},i) = 0$

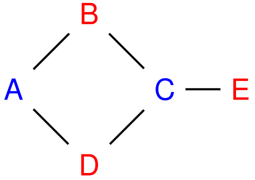
Fill in subsets in order of size.

Table Size: $2^n \times n$.
Fill in each entry: $O(n)$ time.
Time: $O(n^2 2^n)$

Answer? (A) $\min_{S,j}\{C(S,j) + d_{j1}\}$ (B) $\min_j\{C(V,j) + d_{j1}\}$, (C) $\min_j\{C(V,j)\}$

Dominating Set

Given a graph $G = (V, E)$, find the smallest subset, S, where $\forall v \in V, v \in S,$ or $,(u, v) \in E$ and $u \in S$.



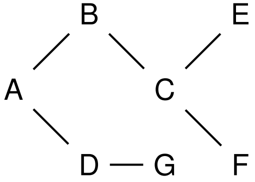
Dominating Set!
Better Dominating Set.

Application?
Place ice cream stand on corners
... and only one block to any ice-cream.

What could be more important than ice-cream!

Vertex Cover on a Tree

Given a tree $T = (V, E)$, find the smallest subset, S, where $\forall v \in V, v \in S,$ or $, (u, v) \in E$ and$u \in S$.



What's the best dominating set?

Root tree at A. Subproblems correspond to subtrees.

Best solution S structure?

for subtree at B.

B could be in S

B could not be in S and be covered by a node in subtree.

B could not be in S and not be covered by a node in subtree.

Best solution S structure?

for subtrees at u.

u could be in S

u could not be in S and be covered by a node in subtree.

u could not be in S and not be covered by a node in subtree.

Subproblem: DS(u,in_cover,covered).

vertex u, booleans in_cover,covered.

Best solution where u is in cover or not

and covered or not in subtree.

DS(u,true,true)

$= 1 + \sum_{\text{subtree}_v} \min\{DS(v, false, false), DS(v, false, true), DS(v, true, true)\}.$

DS(u,false,true) $=$

$\min_{\text{subtree}_x} DS(x, true, true) +$

$\sum_{\text{subtree } v \neq x} \min\{DS(v, false, true), DS(v, true, true)\}.$

DS(u,false,false) $= \sum_{\text{subtree}_v} DS(v, false, true)$

Subproblem: DS(u,in_cover,covered).
vertex u, booleans in_cover,covered.
Best solution where u is in cover or not
and covered or not in subtree.

DS(u,true,true)
$= 1 + \Sigma_{\text{subtree}_v} \min\{DS(v, false, false), DS(v, false, true), DS(v, true, true)\}.$

DS(u,false,true) $=$
$\min_{\text{subtree}_x} DS(x, true, true) +$
$\Sigma_{\text{subtree } v \neq x} \min\{DS(v, false, true), DS(v, true, true)\}.$

DS(u,false,false) $= \Sigma_{\text{subtree}_v} DS(v, false, true)$

For leaf u. DS(u,false,false)? 0
For leaf u. DS(u,true,true)? 1
For leaf u. DS(u,false,true)? ?? Big, actually.

$O(V)$ table entries.
Fill in entry for degree d node, in time $O(d^2)$.
$O(d)$ with a bit of care.
$O(|E|)$ time.