EECS 336: Introduction to Algorithms	Lecture 11
Dynamic Programming (cont)	Bellman-Ford

Reading: 6.4, 6.8

Last time:

- Integer Knapsack
- Interval Pricing
- "finding a first decision"

Today:

- Shortest Paths.

Suggested Approach

I. identify subproblem in english

   OPT(i) = "optimal schedule of $\{i, \ldots, n\}$ (sorted by start time)"

II. specify subproblem recurrence

   $\mathrm{OPT}(i) = \max(\mathrm{OPT}(i+1), v_i + \mathrm{OPT}(\mathrm{next}(i)))$

III. identify base case

   $\mathrm{OPT}(n+1) = 0$

IV. write iterative DP.

   (see last thurs)

Finding Optimal Schedule

"traverse memoization table to find schedule"

Algorithm: schedule

$i = 1$

while $i < n$

  if memo[$i+1$] $<$ v_i+ memo[next(i)]

    schedule i; $i \leftarrow$ next(i).

  else

    $i \leftarrow i + 1$.

  endif

endwhile

Shortest Paths with Negative Weights

"e.g., currency exchange: nodes are currencies, path weights are exchange rates, minimize product of path weights."

Note: to minimize product of path weights, can minimize sum of logs of path weights.

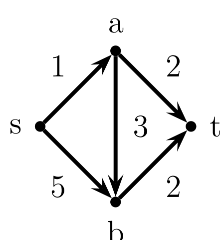
Example: $r_1 r_2 = 2^{\log_2 r_1} 2^{\log_2 r_2} = 2^{\log_2 r_1 + \log_2 r_2}$.

Note: if $r \leq 1$ then $\log r$ is negative.

Recall: Dijkstra's Algorithm

1. initialize known distance from s as ∞, except $d(s) = 0$
2. take closest unknown vertex v
   (a) declare v known.
   (b) update known distances to neighbors of v if closer via v.
3. repeat (2) until t known.

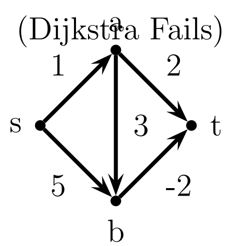
Example:



Shortest Path: d(s-a-t) $= 3$.

Negative Edge Weights

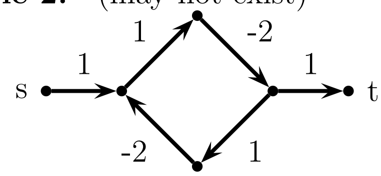
Example 1: (Dijkstra Fails)



Dijkstra's Path: d(s-a-t) $= 3$

Shortest Path: d(s-a-b-t) $= 2$.

Example 2: (may not exist)

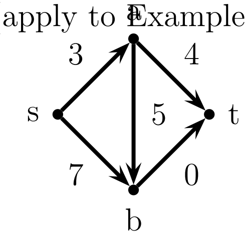


Negative cycle $\Rightarrow$ no shortest path.

First try:

- find most negative edge "$-c$"
- add c to all edges.
- run Dijkstra

Example: (apply to Example 1)



Shortest Paths: s-a-t or s-b-t, not shortest in original problem.

Second Try: Dynamic programming

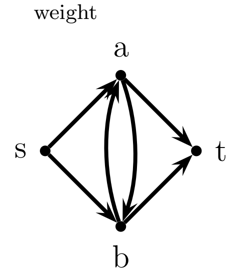
subproblem:

OPT(v)

$= $ shortest path from v to t.

$= \min_{u \in N(v)} [\underbrace{c(v,u)}_{\text{weight}} + \mathrm{OPT}(u)]$.

Example:



Subproblems have cyclic dependencies!

Imposing measure of progress

"parameterize subproblems to keep track of progress"

Lemma: if G has no negative cycles, then minimum cost path is **simple** (i.e., does not repeat nodes); therefore, it has at most $n-1$ edges.

Proof: (contradiction)

- let P be the min-length path with fewest number of edges.
- suppose (for contradiction) that P is not simple.

  $\Rightarrow$ P repeats a vertex v.
- no negative cycle $\Rightarrow$path from v to v non-negative.

  $\Rightarrow$ can "splice out" cycle and not increase length.

  $\Rightarrow$ new path has fewer edges than p.

$\rightarrow\leftarrow$

Idea: if simple path goes $s \rightsquigarrow v \rightarrow u \rightsquigarrow t$ then u-t path has one fewer edge than v-t path.

Part I: identify subproblem in english

$\text{OPT}(v,k)$

= "length of shortest path from v to t with at most k edges."

Part II: write recurrence

$\text{OPT}(v,k)$

$= \min_{u\in N(v)} [c(v,u) + \text{OPT}(u,k-1)]$

Part III: base case

- for all k: $\text{OPT}(t,k) = 0$.
- for all $v \neq t$: $\text{OPT}(v,0) = \infty$.

Part IV: iterative DP

Algorithm: Bellman-Ford

1. initialize

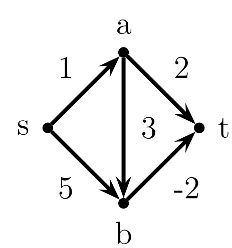
   for all k: $\text{memo}[t,k] = 0$.

   for all $v \neq t$: $\text{memo}[v,0] = \infty$.
2. for $k = 1$ up to $n-1$,

   for all v

   $\text{memo}[v,k] = \min_{u\in N(v)} \text{OPT}(u,k-1)$.
3. return $\text{memo}[s,n-1]$.

Example:



	0	1	2	3
s	∞	∞	3	2
a	∞	2	1	1
b	∞	-2	-2	-2
t	0	0	0	0

Correctness

lemma + induction.

Runtime

$$
\begin{aligned}
T(n,m) &= \overbrace{\text{“size of table”}}^{n^2} \times \overbrace{\text{“cost per entry”}}^{n} \\
&= O(n^3)
\end{aligned}
$$

(better accounting: $T(n,m) = O(n^2 + nm) = O(nm)$)