EECS 336: Introduction to Algorithms	Lecture 5
Dynamic Programming (cont)	Bellman-Ford

Reading: 6.4, 6.8
"guide to dynamic programming" (Canvas)

Last time:

- Dynamic Programming (a framework)
- Integer Knapsack

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

OPT(i) = max(OPT($i + 1$), v_i + OPT(next(i)))

III. solve original problem (from subproblems)

Optimal Interval Schedule = OPT(1)

IV. identify base case

OPT($n + 1$) = 0

V. write iterative DP.

(see last thurs)

VI. analyze runtime.

$O(n \log n)$

VII. (for homework) implement iterative DP.

(any language most students can read. e.g., Python)

Shortest Paths with Negative Weights

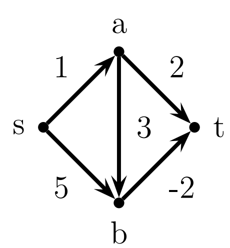
"e.g., currency exchange: nodes are currencies, path weights are exchange rates, minimize product of path weights."

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.

Example:



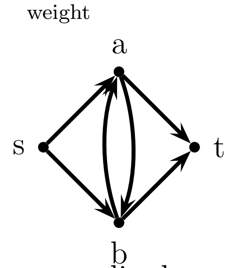
Try Dynamic programming

OPT(v)

$= $ shortest path from v to t.

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

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 $s \leadsto v \rightarrow u \leadsto t$ then u-t path has one fewer edge than v-t path.

Part I: identify subproblem in english

OPT(v, k)

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

Part II: write recurrence

OPT(v, k)

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

Correctness: lemma + induction.

Part III: solve original problem

- minimum cost path = $\mathrm{OPT}(s, n-1)$.

Part IV: base case

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

Part V: iterative DP

Algorithm: Bellman-Ford

1. initialize

   for all k: $\mathrm{OPT}[t, k] = 0$.

   for all $v \neq t$: $\mathrm{OPT}[v, 0] = \infty$.

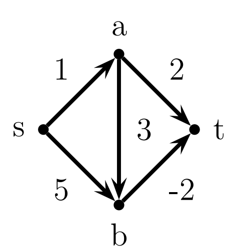
2. for $k = 1$ up to $n - 1$,

   for all v

   $\mathrm{OPT}[v, k] = \min_{u \in N(v)} \mathrm{OPT}(u, k - 1)$.

3. return $\mathrm{OPT}[s, n - 1]$.

Example:



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

Part VI: Runtime

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

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