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

\[ \text{OPT}(i) = \text{“optimal schedule of } \{i, \ldots, n\} \text{ (sorted by start time)”} \]

II. specify subproblem recurrence

\[ \text{OPT}(i) = \max(\text{OPT}(i + 1), v_i + \text{OPT(\text{next}(i))}) \]

III. identify base case

\[ \text{OPT}(n + 1) = 0 \]

IV. write iterative DP.

(see last thurs)

**Finding Optimal Schedule**

“traverse memoization table to find schedule”

**Algorithm:** schedule

\[ i = 1 \]

\[ \text{while } i < n \]

\[ \text{if memo}[i + 1] < v_i + \text{memo}[\text{next}(i)] \]

\[ \text{schedule } i; \ i \leftarrow \text{next}(i). \]

\[ \text{else} \]

\[ i \leftarrow i + 1. \]

\[ \text{endif} \]

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 \( \infty \), 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:

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)

Second Try: Dynamic programming

subproblem:

\[ \text{OPT}(v) = \text{shortest path from } v \text{ to } t. \]

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

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 $p$.

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

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

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

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 $k$: memo$[t, k] = 0$.
   - for all $v \neq t$: memo$[v, 0] = \infty$.

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

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

3. return memo$[s, n - 1]$.

Example:

$$\begin{array}{|c|c|c|c|c|}
\hline
  & 0 & 1 & 2 & 3 \\
\hline
s & \infty & \infty & 3 & 2 \\
a & \infty & 2 & 1 & 1 \\
b & \infty & -2 & -2 & -2 \\
t & 0 & 0 & 0 & 0 \\
\hline
\end{array}$$

Correctness

lemma + induction.
Runtime

\[ T(n, m) = \text{”size of table”} \times \text{”cost per entry”} = O(n^3) \]

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