Shortest Paths III: Special cases

Shortest paths in DAGs
Shortest paths in graphs w/o negative edges
Dijkstra's algorithm

Readings: CLRS 24.2, 24.3

DAGs
Can't have negative cycles because there are no cycles!

1) Topologically sort the DAG, path from $u$ to $v$ implies that $u$ is before $v$ in the linear ordering

2) One pass over vertices in topologically sorted order relaxing each edge that leaves each vertex
$\Theta(V+E)$ time
EXAMPLE

Vertices sorted left to right in topological order

Process r: stays ∞. All vertices to the left of s will be ∞ by definition.

Process s: t: ∞ → 2  x: ∞ → 6

Process t, x, y

preview of dynamic programming
Dijkstra’s Algorithm

For each edge $(u,v) \in E$, assume $w(u,v) \geq 0$

Maintain a set $S$ of vertices whose
final shortest path weights have been determined

Repeatedly select $u \in V - S$ with minimum
shortest path estimate, add $u$ to $S$, relax
all edges out of $u$

Pseudo Code

\begin{verbatim}
Dijkstra (G, w, s)  // uses priority queue Q

  Initialize (G, s)
  $S \leftarrow \emptyset$
  $Q \leftarrow V[G]$  // Insert into Q
  while $Q \neq \emptyset$
    do $u \leftarrow$ EXTRACT-MIN ($Q$)  // deletes $u$ from $Q$
    $S' \leftarrow S \cup \{u\}$
    for each vertex $v \in \text{Adj}[u]$
      do RELAX ($u,v,w$)

  RELAX ($u, v, w$)
  if $d[v] > d[u] + w(u,v)$
    then $d[v] \leftarrow d[u] + w(u,v)$
    $\pi[v] \leftarrow u$
\end{verbatim}
**Example**

![Graph Image]

\[
S = \{\} \quad \{A, B, C, D, E\} = Q
\]
\[
S = \{A\} \quad 0 \quad \infty \quad \infty \quad \infty \quad \infty \quad \text{after relaxing edges from A}
\]
\[
S = \{A, C\} \quad 0 \quad 10 \quad 3 \quad \infty \quad \infty \quad \infty \quad \text{after relaxing edges from C}
\]
\[
S = \{A, C, E\} \quad 0 \quad 7 \quad 3 \quad 11 \quad 5 \quad \text{after relaxing edges from C}
\]
\[
S = \{A, C, E, B\} \quad 0 \quad 7 \quad 3 \quad 9 \quad 5 \quad \text{after relaxing edges from B}
\]

**Strategy:** Dijkstra is a greedy algorithm: choose closest vertex in \( V - S \) to add to set \( S \)

**Correctness:** Each time a vertex \( u \) is added to set \( S' \), we have \( d[u] = S'(u) \).
\( \Theta(V) \) inserts into priority queue

\( \Theta(V) \) \textit{extract-min} operations

\( \Theta(E) \) \textit{decrease-key} operations

Array impl: \( \Theta(V) \) time for extract min
\( \Theta(1) \) for decrease key

Total: \( \Theta(V \cdot V + E \cdot 1) = \Theta(V^2 + E) = \Theta(V^2) \)

Binary \textit{min-heap}:
\( \Theta(lgV) \) for extract min
\( \Theta(lgV) \) for decrease key

Total: \( \Theta(V \cdot lgV + E \cdot lgV) \)

Fibonacci heap:
\( \Theta(lgV) \) for extract min
\( \Theta(1) \) for decrease key

Amortized cost

Total: \( \Theta(V \cdot lgV + E) \)