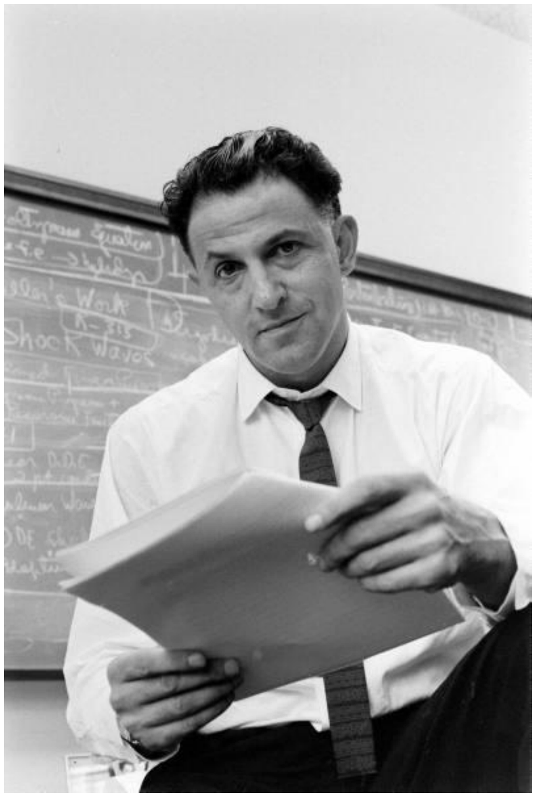
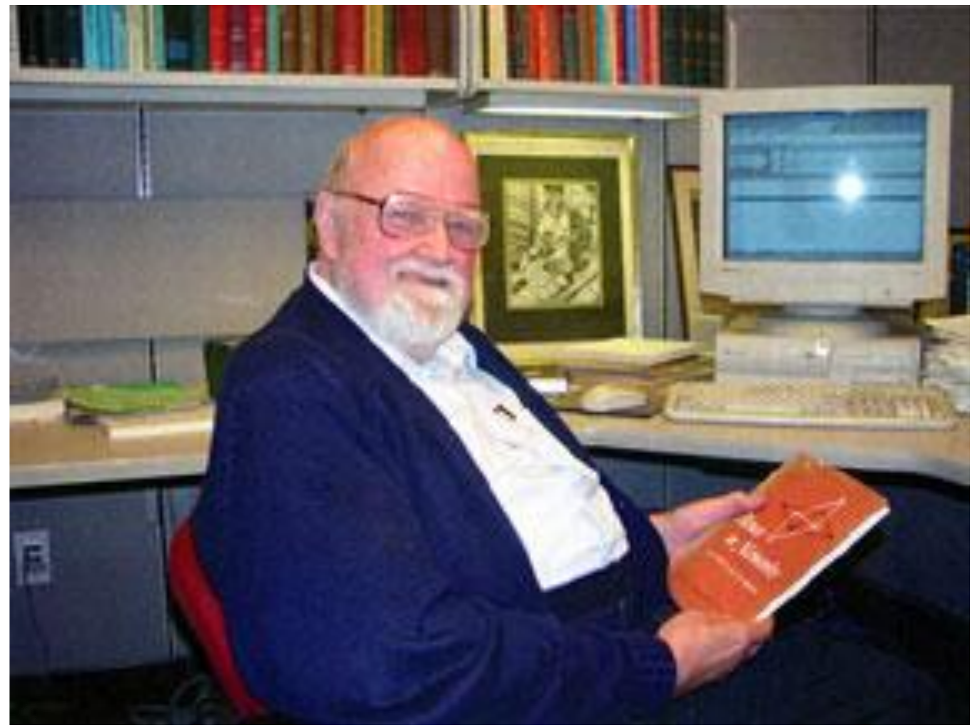
CS380
Artificial Intelligence for Games

Bellman-Ford’s Search

Bellman-Ford's Search



Richard Ernest <u>Bellman</u>
1925-2003



Lester Randolph <u>Ford</u>, Jr
1927-2017

Bellman-Ford's Search

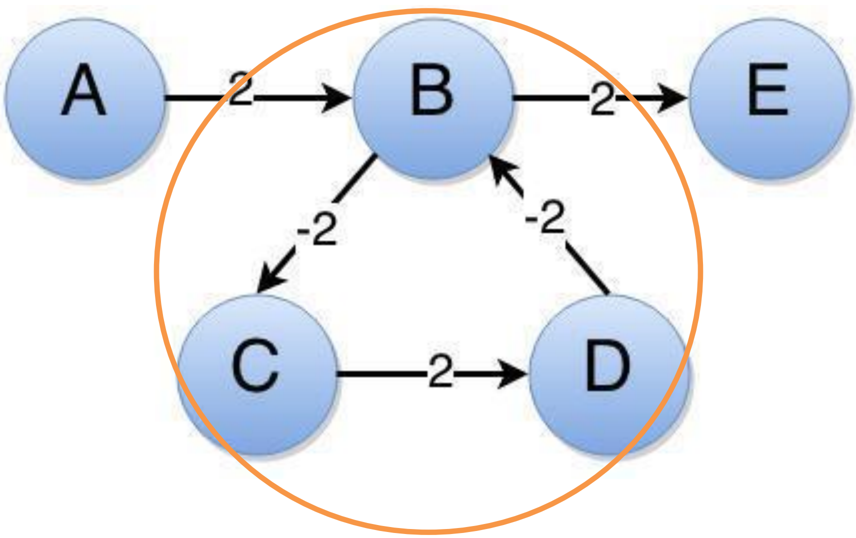
- It finds the shortest path from a source node to all other nodes in a graph for <u>any positive or negative</u> edge cost values in a digraph or tells whether the graph contains **negative cycles**.

- It may consider nodes that are already expanded as long as they are reachable with a less cost.

- **Dynamic programming** algorithm
  - Optimal Substructure

Bellman-Ford's Search

- Cycles with negative total weights



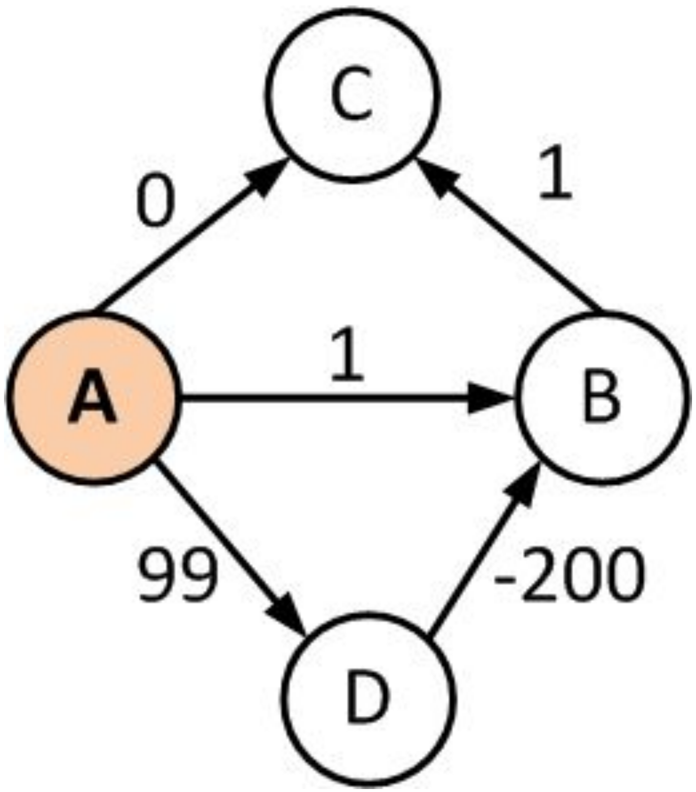
- Negative edges make the shortest path problem harder.
- Negative cycles make the path problem non-traceable.

Bellman-Ford's Search

- Claim: If G has **no** negative cycles, then there is a shortest path from s to t that is simple (i.e. does not repeat nodes), and hence has at most $n - 1$ edges.

```
for (each vertex v in Graph){
  dist[v] = +∞;
  previous[v] = null;
}
dist[source] = 0;
counter = 0;
for (i=0; i < numberOfVertices; i++) {
   for (each edge (u, v) with cost w)      // A
         if (dist[u] + w < dist[v]) {
             dist[v] = dist[u] + w;
             previous[v] = u;
             counter++;
         }
     if (counter==0) break;
 }
 if (counter==0) then there is no negative cycles
 else {
   run loop A again
   if (counter==0) then there is no negative cycles
   else there is a negative cycle
 }
```

Bellman-Ford's Search



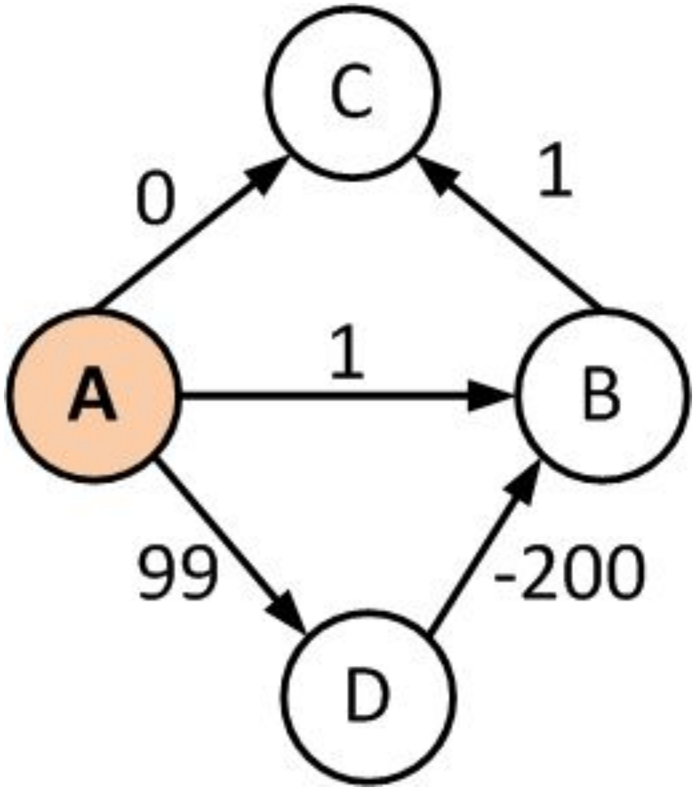
dist[A] = 0
dist[B] = +∞
dist[C] = +∞
dist[D] = +∞

previous[A] = NA
previous[B] = NA
previous[C] = NA
previous[D] = NA

i=0

counter = 0

Bellman-Ford's Search



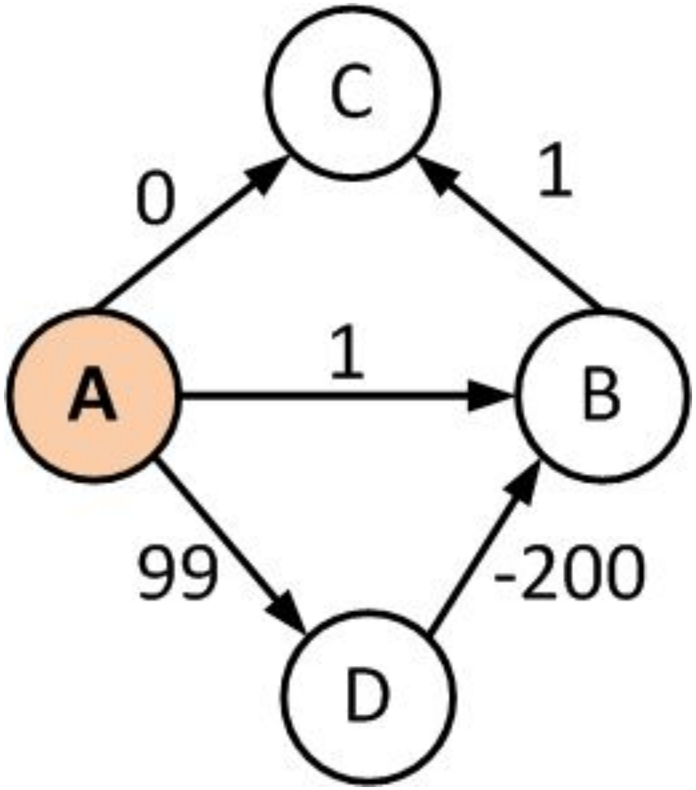
dist[A] = 0
dist[B] = +∞
dist[C] = **0**
dist[D] = +∞

previous[A] = NA
previous[B] = NA
previous[C] = **A**
previous[D] = NA

i=1

counter = **1**

Bellman-Ford's Search



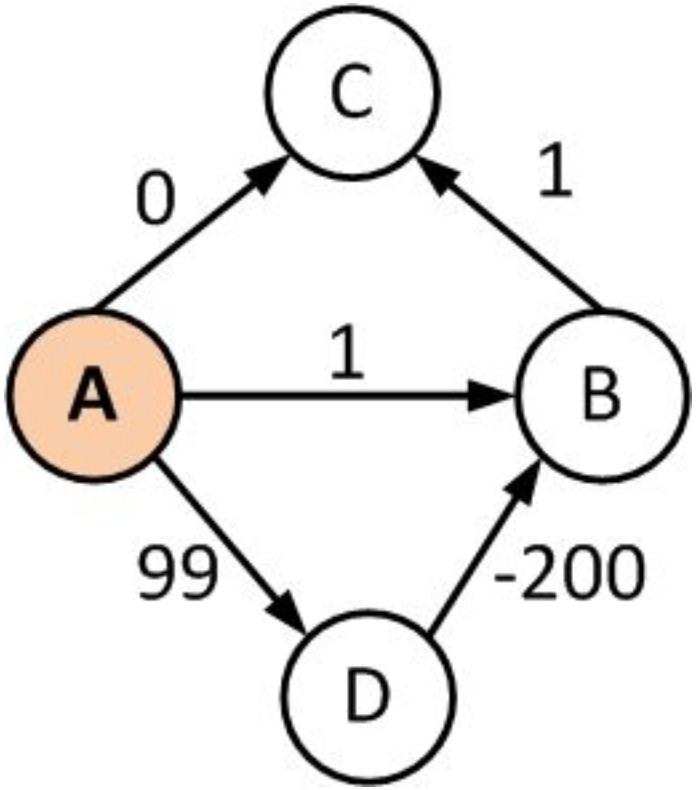

```
dist[A] = 0
dist[B] = +∞
dist[C] = 0
dist[D] = 99

previous[A] = NA
previous[B] = NA
previous[C] = A
previous[D] = A

i=1

counter = 2
```

Bellman-Ford's Search



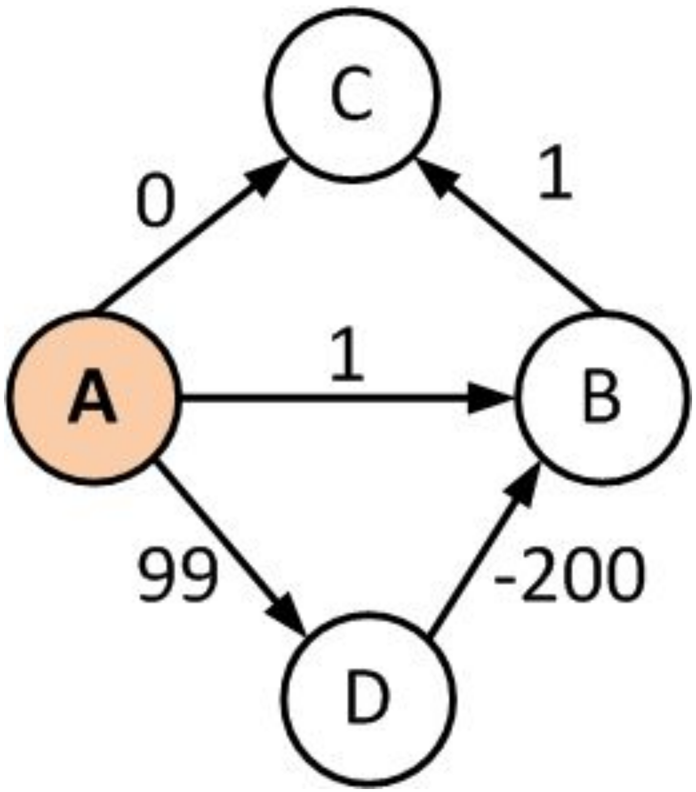

```
dist[A] = 0
dist[B] = 1
dist[C] = 0
dist[D] = 99

previous[A] = NA
previous[B] = A
previous[C] = A
previous[D] = A
```

i=1

counter = 3

Bellman-Ford's Search



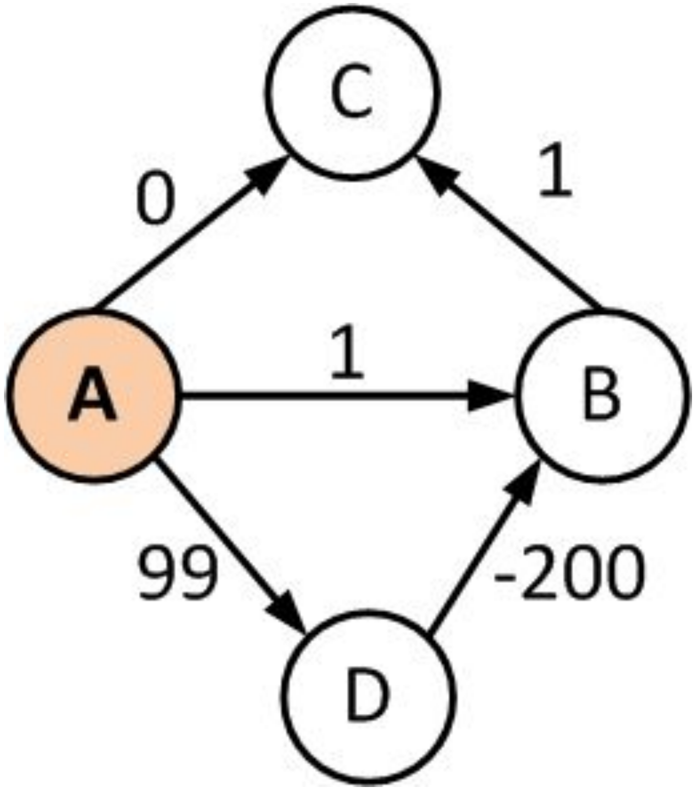

```
dist[A] = 0
dist[B] = 1
dist[C] = 0
dist[D] = 99

previous[A] = NA
previous[B] = A
previous[C] = A
previous[D] = A
```

i=2

counter = **0**

Bellman-Ford's Search



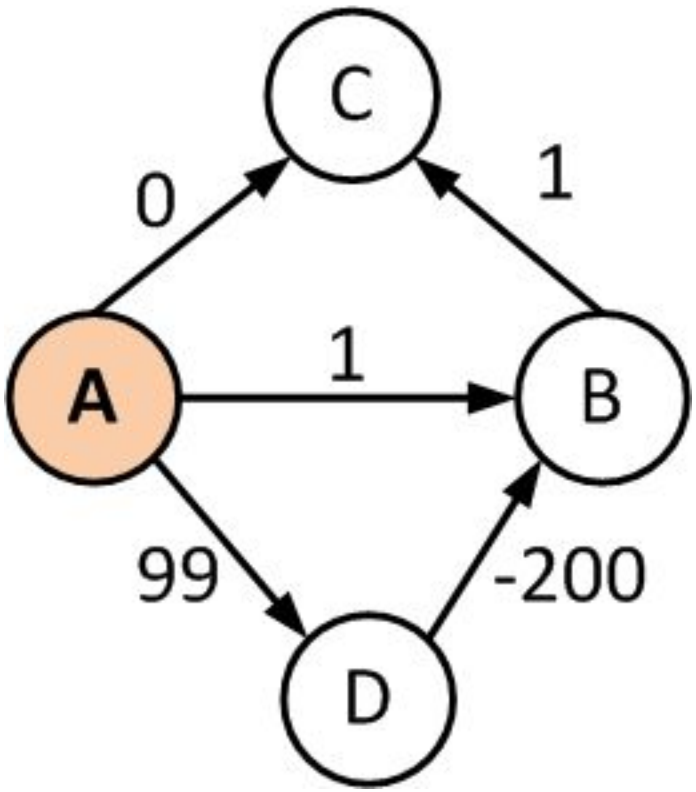

```
dist[A] = 0
dist[B] = -101
dist[C] = 0
dist[D] = 99

previous[A] = NA
previous[B] = D
previous[C] = A
previous[D] = A
```

i=2

counter = **1**

Bellman-Ford's Search



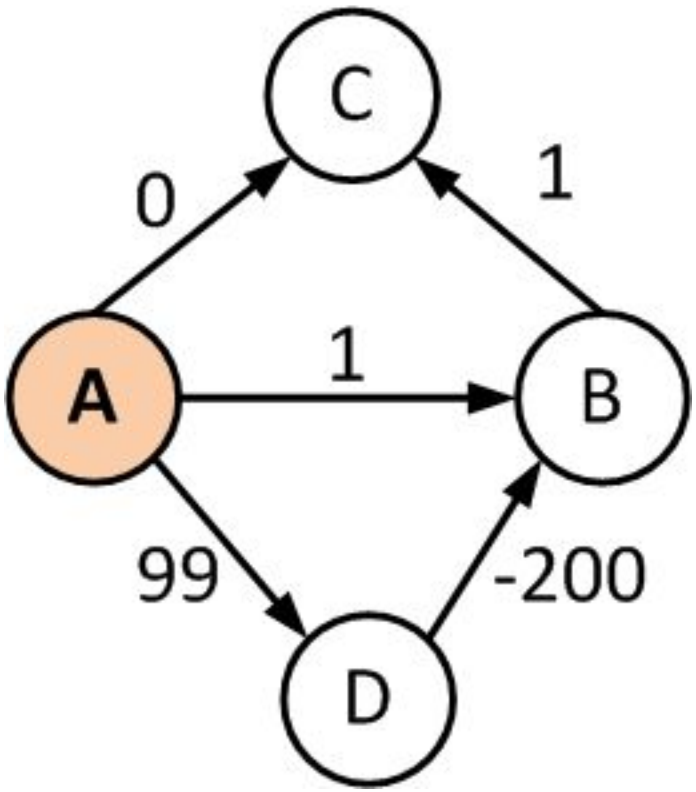

```
dist[A] = 0
dist[B] = -101
dist[C] = 0
dist[D] = 99

previous[A] = NA
previous[B] = D
previous[C] = A
previous[D] = A
```

```
i=3
```

```
counter = 0
```

Bellman-Ford's Search



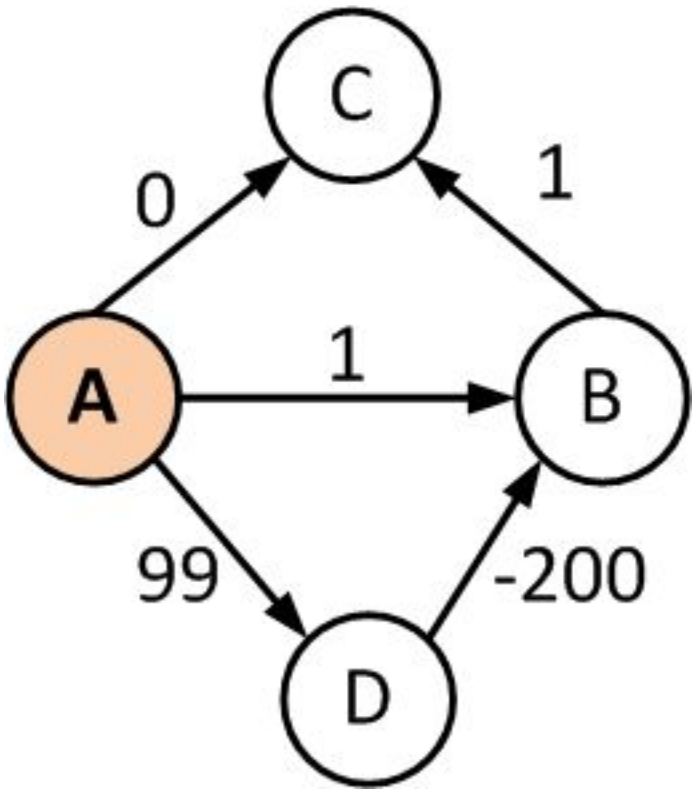
dist[A] = 0
dist[B] = -101
dist[C] = **-100**
dist[D] = 99

previous[A] = NA
previous[B] = D
previous[C] = **B**
previous[D] = A

i=3

counter = **1**

Bellman-Ford's Search



dist[A] = 0
dist[B] = -101
dist[C] = -100
dist[D] = 99

previous[A] = NA
previous[B] = D
previous[C] = B
previous[D] = A

i=4

counter = **0**

No negative cycles

Bellman-Ford's Search

- **Complete?**
  - Yes.
- **Optimal?**
  - Yes, if the state space doesn't contain cycles with negative total values.
- **Complexity** in terms of number of vertices: n and number of edges: m
  - Time complexity: $O(m * n) = O(n^3)$, note: $m = O(n^2)$.
  - Space complexity: $O(n)$

Self-Test

Question: How many invocations of a single source shortest-path subroutine are needed to solve the all-pairs shortest path problem? [n = # of vertices]

a) 1
b) n-1
c) n
d) n^2

Self-Test

Question: How many invocations of a single source shortest-path subroutine are needed to solve the all-pairs shortest path problem? [n = # of vertices]

a) 1
b) n-1
c) <u>n</u>
d) n^2