

Lesson 3: Shortest-path Problems

Notes

Book acknowledgment:

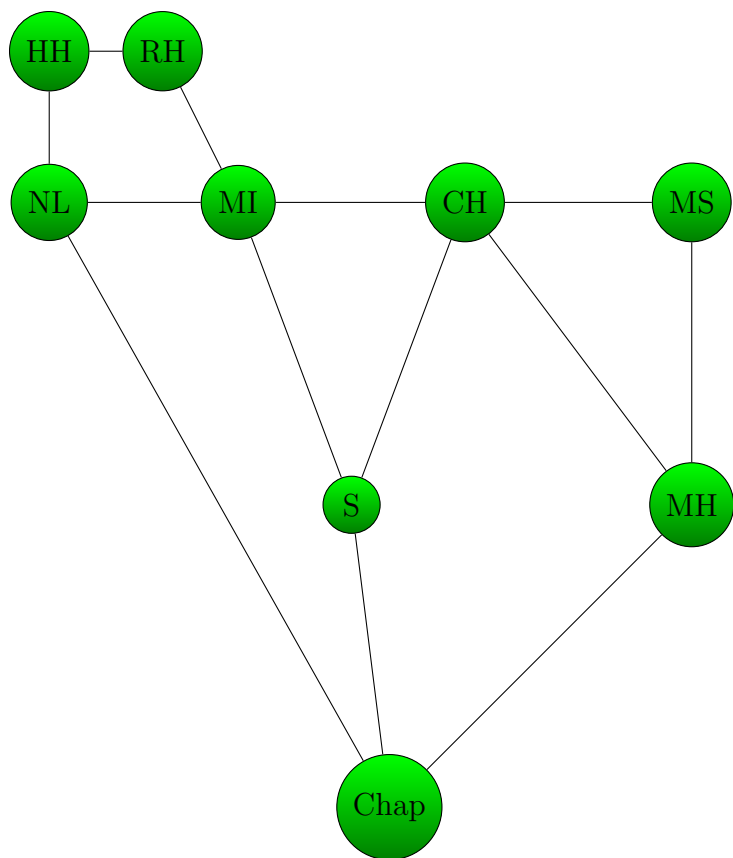
Goals

- Learn the Shortest Path Problem
- Learn and implement the following algorithms:
 - Dijkstra's Algorithm
 - Bellman-Ford Algorithm

Example:

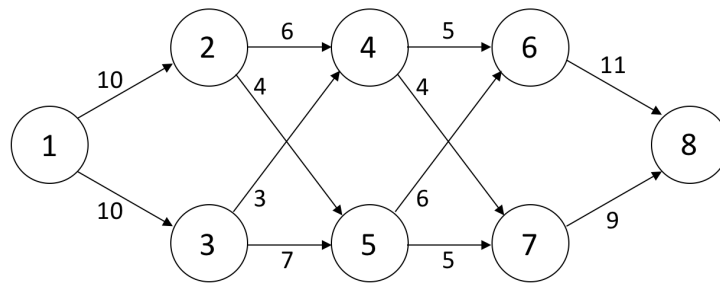
Below is a network representation of some of the buildings on the Yard.

- HH is Hopper Hall
- RH is Rickover Hall
- NL is Nimitz Library
- MI is Michelson
- CH is Chauvenet
- S is the middle of Stribling
- MS is the Mid-Store
- MH is Memorial Hall
- Chap is the Chapel



Label each arc with *about* how long you think it might take to walk from one building to the other.

1 More General Example



a. Determine a shortest path from 1 to 8.

b. Determine a shortest path from 1 to every other node.

2 Problem Definition

- Given a directed graph $G = (V, A)$, let s be the source (origin) node and t be the terminal (sink) node.
- Let c_{ij} be the cost of traversing arc $(i, j) \in A$.

Shortest path problem:

Determine the minimum cumulative cost path from s to t .

How would you go about solving this on your own? (Given what you already know from previous courses.)

3 Dijkstra's Algorithm

- Dijkstra's algorithm is very similar to Prim's Algorithm in that we will keep track of which nodes are and are not included in a shortest path from s to t , so far.
- In doing so, we will actually build a **shortest-path tree** rooted at the source node s .
- At each step of the algorithm, we will search for a node not yet in the shortest-path tree having a minimum distance from the source.

Written Steps:

1. Initialize set $S = \emptyset$ that contains all the nodes currently in the shortest-path tree. Assign a minimum distance value v_i to each node $i \in V$ in the graph. Initialize $v_i = \infty$, $\forall i \in V$, and set $v_s = 0$. Proceed to Step 2.
2. If $S = V$, then proceed to Step 4. Otherwise, pick a node $u \in V \setminus S$ having a minimum distance value $v_u = \min\{v_i : i \in V \setminus S\}$, add u to S , and proceed to Step 3.
3. Update the distance value for each adjacent node of u . For all $(u, w) \in A$, if $v_w > v_u + c_{uw}$, then set $v_w = v_u + c_{uw}$ and set the parent of w to be u . Return to Step 2.
4. Terminate with the shortest-path tree S , distance value vector v , and parent vector.

Pseudocode:

Algorithm 1 Determine a shortest-path tree rooted at source node s

Let $S = \emptyset$ be the set of shortest-path tree nodes.

Set $v_i = \infty$, $\forall i \in V$, set $v_s = 0$, and set $parent(i) = -1$, $\forall i \in V$

while $S \subset V$: **do**

 Pick a node $u \in V \setminus S$ for which $v_u = \min\{v_i : i \in V \setminus S\}$.

 Add u to S .

for $(u, w) \in A$ **do**

if $v_w > v_u + c_{uw}$, **then**

 Set $v_w = v_u + c_{uw}$

 Set $parent(w) = u$

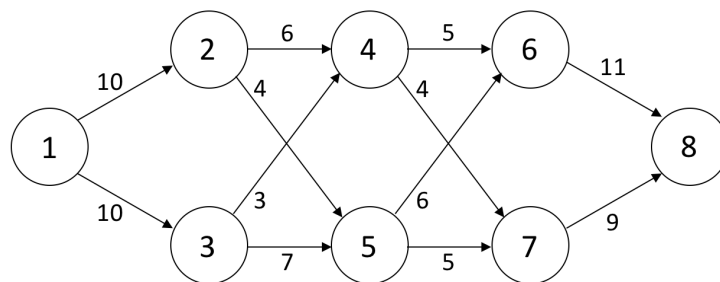
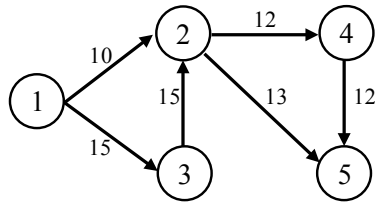
end if

end for

end while

Let's practice!

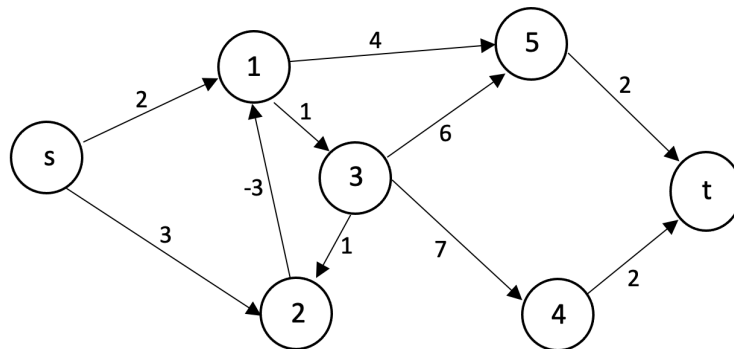
Use Dijkstra's Algorithm to solve the two examples from earlier.



4 What could go wrong?

- The two examples from earlier all have positive arc costs. What could go wrong if some arcs have a negative cost?

Use Dijkstra's Algorithm to determine the shortest path from s to t .



5 Bellman-Ford Algorithm

In the worst-case, the Bellman-Ford Algorithm is slightly slower than Dijkstra's Algorithm. It is more versatile because it is capable of handling graphs having negative arc costs. If a graph/network has a negative-cost cycle, then the Bellman-Ford Algorithm is able to detect and report it.

Like Dijkstra's Algorithm, the Bellman-Ford iteratively updates approximations to the minimum computed distance to each node in the network. Whereas Dijkstra's uses a greedy approach that selects the *closest* node that has not yet been added to the shortest-path tree set S , Bellman-Ford updates all edges at most $|V| - 1$ times.

Written Steps:

1. Set $k = 0$, $v_s = 0$, and $v_i = \infty, \forall i \in V \setminus \{s\}$
2. If $k > |V|$, then proceed to Step 4. Otherwise, set $k = k + 1$ and proceed to Step 3.
3. For all $(i, j) \in A$, if $\infty > v_j > v_i + c_{ij}$, then set $v_j = v_i + c_{ij}$ and set the parent of j to be i . Return to Step 2.
4. Terminate the Bellman-Ford Algorithm with the shortest-path tree corresponding to the parent vector.

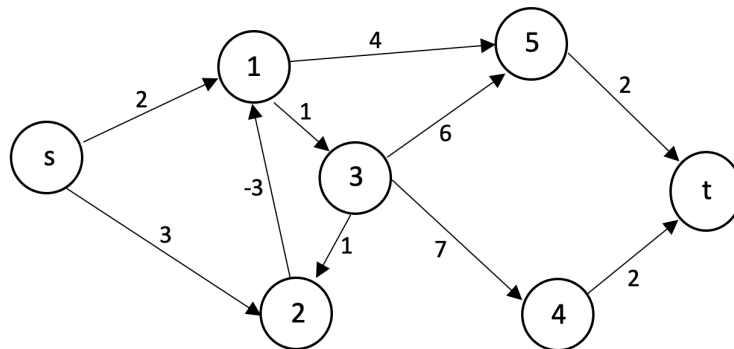
Pseudocode:

Algorithm 2 Determine a shortest-path tree rooted at source node s

Set $k = 0$, $v_s = 0$, $v_i = \infty, \forall i \in V \setminus \{s\}$, and $parent(i) = -1, \forall i \in V \setminus \{s\}$.

```
while  $k < |V|$ : do
  for  $(i, j) \in A$  do
    if  $\infty > v_j > v_i + c_{ij}$  then
      Set  $v_j = v_i + c_{ij}$ .
      Set  $parent(j) = i$ .
    end if
  end for
  Set  $k = k + 1$ .
end while
```

Use the Bellman-Ford Algorithm to determine the shortest path from s to every other node in $V \setminus \{s\}$.



6 Now, let's code!!