Shortest paths and Minimum Spanning Tree

CS 124 Section 2
Zuzanna Skoczylas

Breadth-first search (BFS)

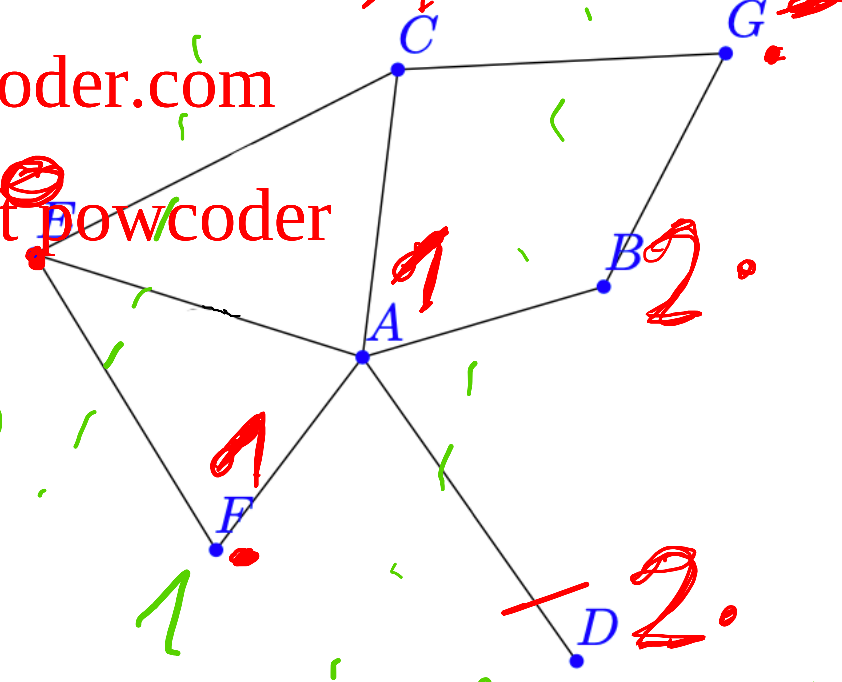
- We use a queue (first-in first-out) to keep track of which vertices have already been visited. Unlike Depth-first search, we visit all of our closest neighbors before moving on to further ones.

- Can be used to find the shortest path when <u>all the edge weights are equal to 1</u>.
- Run-time: $O(|V| + |E|)$

BFS

BFS($G(V, E), s \in V$)

for v in $V - \{s\}$ **do**
  DIST[v] $\leftarrow \infty$
  PREV[v] $\leftarrow$ NIL
  PLACED[v] $\leftarrow$ False
end for
DIST[s] $\leftarrow 0$
PREV[s] $\leftarrow$ *nil*
PLACED[s] $\leftarrow 1$
QUEUE q
ENQUEUE(q, s)
while $q \neq \emptyset$ **do**
  u = DEQUEUE(q)
  for $(u, v) \in E$ **do**
    if not PLACED[v] **then**
      DIST[v] $\leftarrow$ DIST[u] + 1
      PREV[v] $\leftarrow u$
      PLACED[v] $\leftarrow$ True
      ENQUEUE(q, v)
    end if
  end for
end while

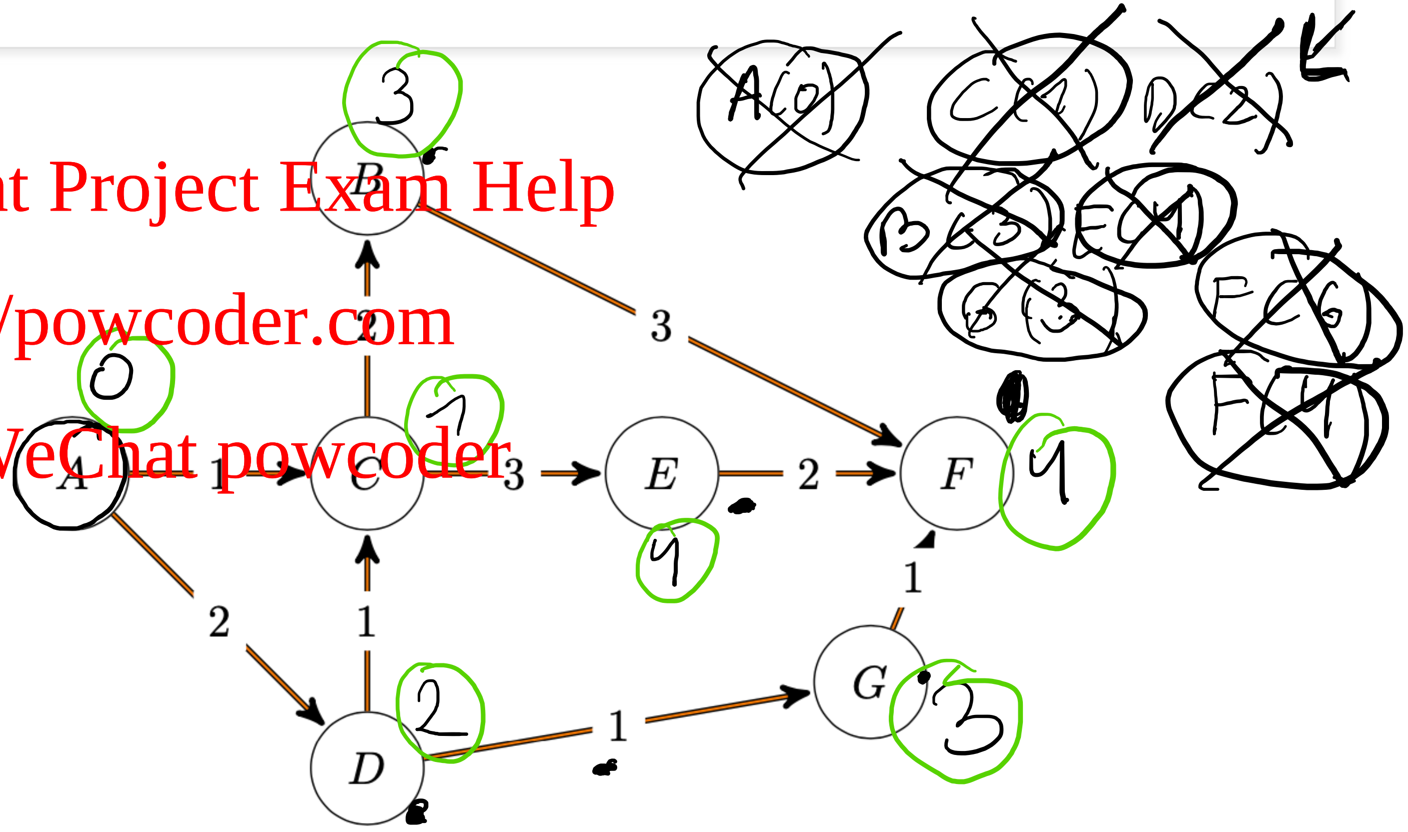


Dijkstra's Algorithm

- Find shortest paths in graphs with any positive edge weights from given point.
- When a node is popped off the heap, that means we have our final answer for that node's distance and path from the source.
- Note that Dijkstra's algorithm does not work for negative edge weights!
- Run-time: O(|V | · deleteMin + |E| · insert).
- The running time of Dijkstra's algorithm depends on the implementation of the heap H. For each vertex, we perform a delete min, while for each edge we perform an insertion.

Dijkstra's Algorithm

```
Dijkstra(G(V, E, ω), s ∈ V)
for v in V do
    DIST[v] ← ∞
    PREV[v] ← NIL
end for
DIST[s] = 0
H ← {(s, 0)}
while H ≠ ∅ do
    v ← DELETEMIN(H)
    for (v, w) ∈ E do
        if DIST[w] > DIST[v] + ω(v, w) then
            DIST[w] ← DIST[v] + ω(v, w)
            PREV[w] ← v
            INSERT((w, DIST[w]), H)
        end if
    end for
end while
```



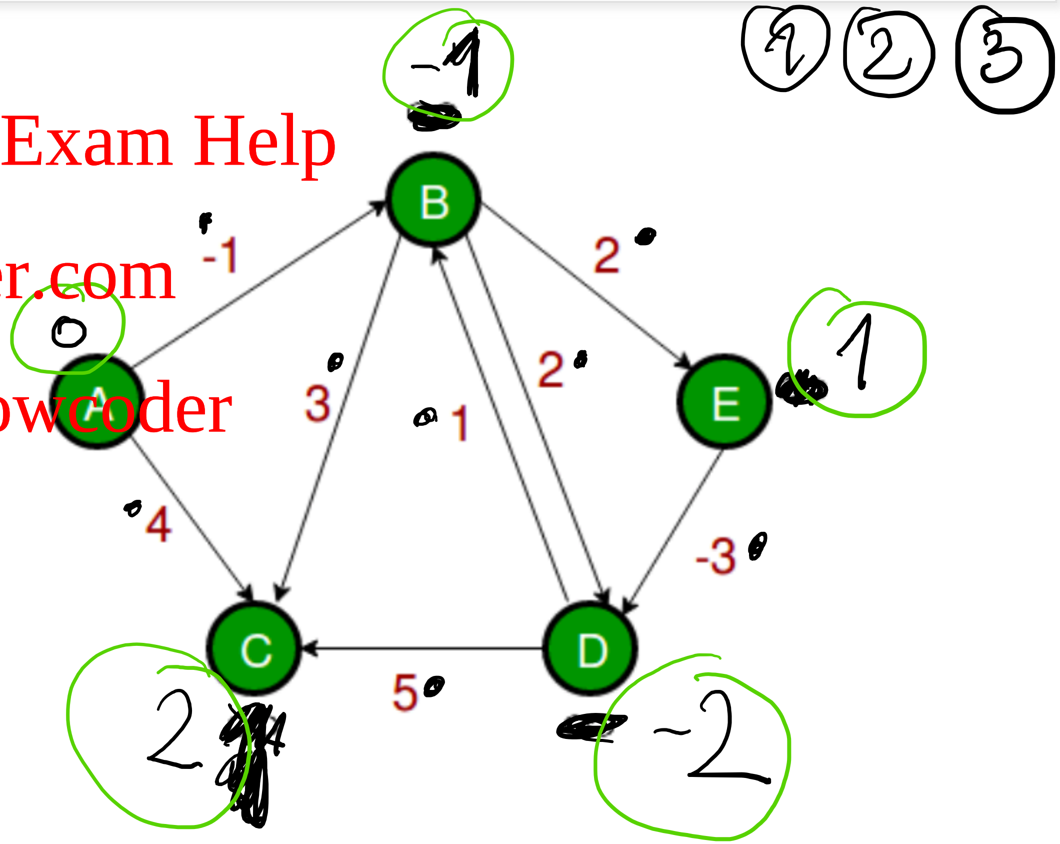
Bellman Ford Algorithm

- Find shortest paths in graphs with any positive or negative edge weights from given point.
- Can be used to find negative cycles in a graph.
- Run-time: $O(|V||E|)$

Bellman Ford

General-SP$(G(V,E,\omega), s \in V)$

```
for v in V − {s} do
    DIST[v] ← ∞
    PREV[v] ← NIL
end for
DIST[s] = 0
for i = 1, 2, ... |V| − 1 do
    for (v, w) ∈ E do
        if DIST[w] > DIST[v] + ω(v, w) then
            DIST[w] ← DIST[v] + ω(v, w)
            PREV[w] ← v
        end if
    end for
end for
```

Minimum Spanning Trees

A **tree** is an undirected graph T = (V, E) satisfying all of the following conditions:

- T is connected,
- T is acyclic,
- |E|=|V|−1.

Any two conditions above imply the third.

A **spanning tree** of an undirected graph G = (V,E) is a subgraph which is a tree and which connects all the vertices. (If G is not connected, G has no spanning trees.)

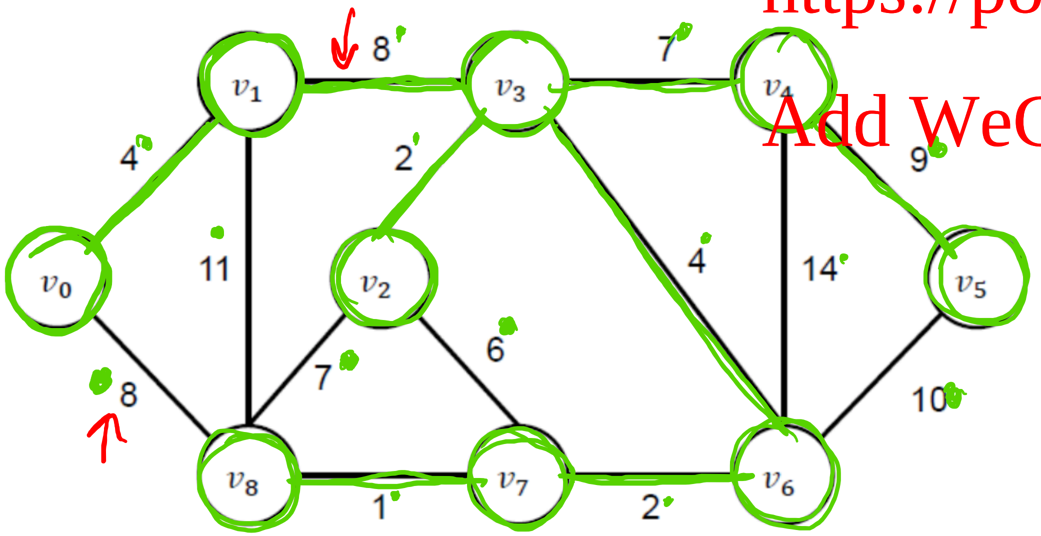
A **minimum spanning tree** is a spanning tree whose total sum of edge costs is minimum.

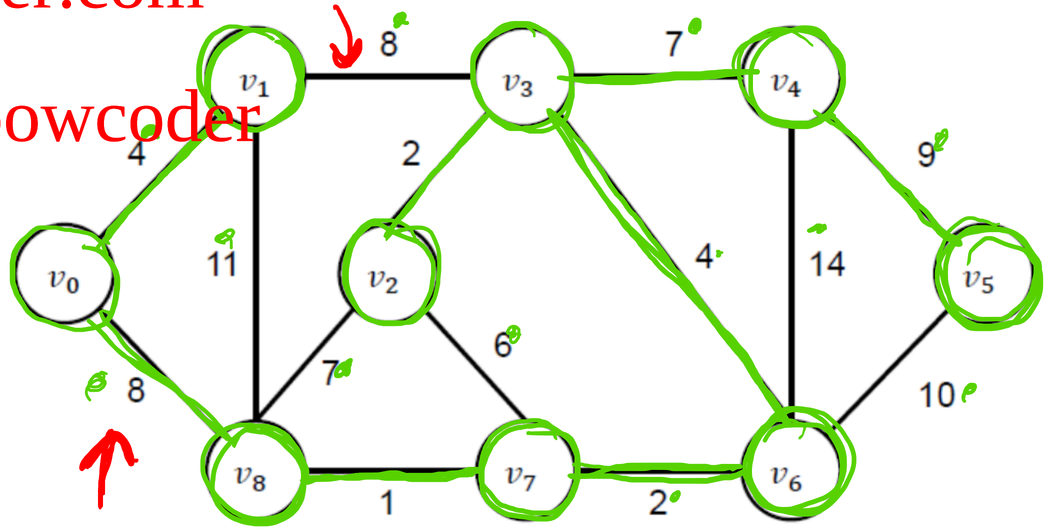
Minimum Spanning Tree - Algorithms

~~1 2 2 4 4~~ 6 ~~7~~ 7 ~~8~~ 8 ~~8~~ 10 11 14

Prim's Algorithm

Kruskal's Algorithm

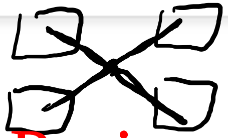


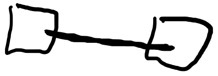


Exercise

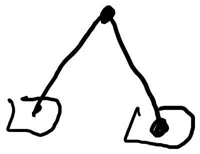
(1) 

3

(2) 
$2\sqrt{2}$



1. (1) You are given a map of a city with houses and a task of planning telephone lines. Each house must be connected to each other house via telephone lines. Each segment of telephone line is between two houses. You want to use as little length of telephone line as you can.

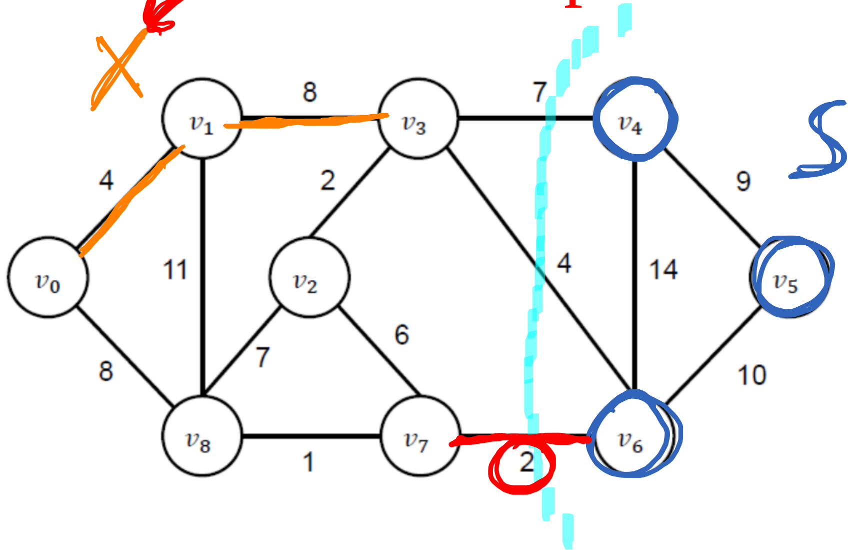


2. (2) You are given the same problem, but this time the endpoints of the line can be located not only in houses.

Which of these problems is a Minimum Spanning Tree?

Cut property

Let X be a subgraph of $G = (V, E)$ such that X is contained in some MST of G. Let $S \subset V$ be a cut such that no edge of X has one endpoint in S. Suppose e has minimum weight among the edges crossing the cut: then $X \cup \{e\}$ is contained in some MST of G. This means that it's safe to add e to our partial MST and still be on-track to finding an MST.

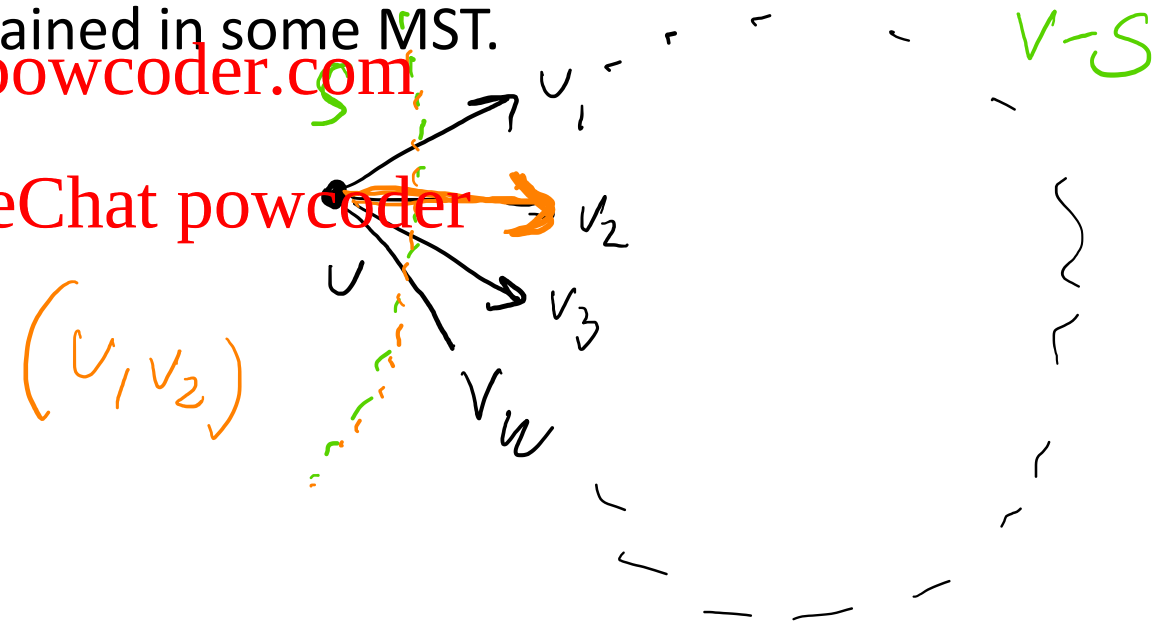


Exercise

Use the cut property to show that given any vertex, the edge extending out of that $\boxed{\text{vertex}}$ with the lightest weight is contained in some MST.

$X = \emptyset$

$S = \{u\}$



Solution

Let u be the vertex we are considering, and let $v\_1, v\_2 \ldots v\_k$ be the neighbors of u. Let $X = \emptyset$ and $S = \{u\}$ in the cut property above. Clearly, X is a subgraph of some MST because the empty graph is a subgraph of all graphs. Our cut is between the sets S and $V - S$, and clearly no edges in our empty X crosses that cut. The only edges that cross this cut are (u, v_1), (u, v_2), . . . , (u, v_k) and thus by the cut property, the lightest one of them must be contained in some MST.

Problem 1

In a city there are N houses, each of which is in need of a water supply. It costs W_i dollars to build a well at house i, and it costs C_{ij} to build a pipe in between houses i and j. A house can receive water if either there is a well built there or there is some path of pipes to a house with a well. Give an algorithm to find the minimum amount of money needed to supply every house with water.

Problem 2

There are n students standing in a playground trying to split into two teams to play kickball. No one cares that the teams have equal, or even nearly equal sizes, but the students do care that they are not on the same team as any of their mortal enemies. You are given a set of m enemy links, which are mutual. If person A has an enemy link with B, they absolutely cannot be on the same team. Give an algorithm to partition the students into two teams such that no enemies are on the same team. If this is not possible, return "Impossible".

Problem 3

(CLRS 21.3-6) Show that a graph has a unique minimum spanning tree if, for every cut of the graph, there is a unique light edge (one of minimum weight) crossing the cut. Conclude that this implies that if a graph has unique edge weights, then it has a unique minimum spanning tree.