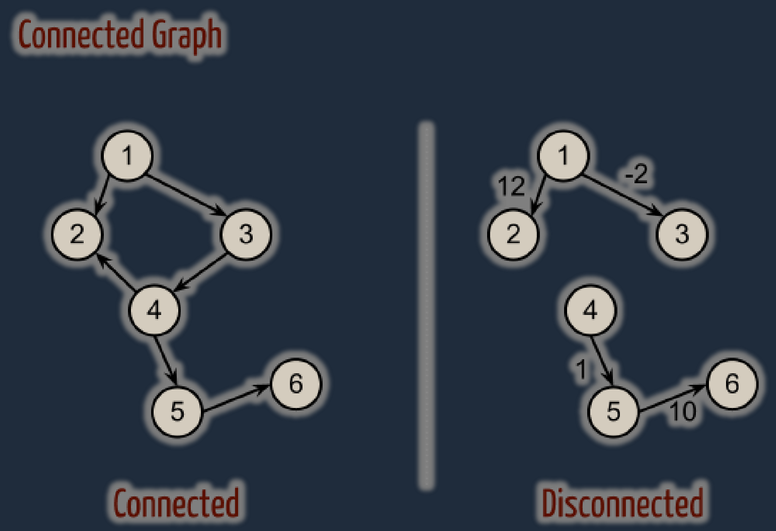
Prim’s Algorithm

A Cursory Introduction to Graph Problems in Computer Science

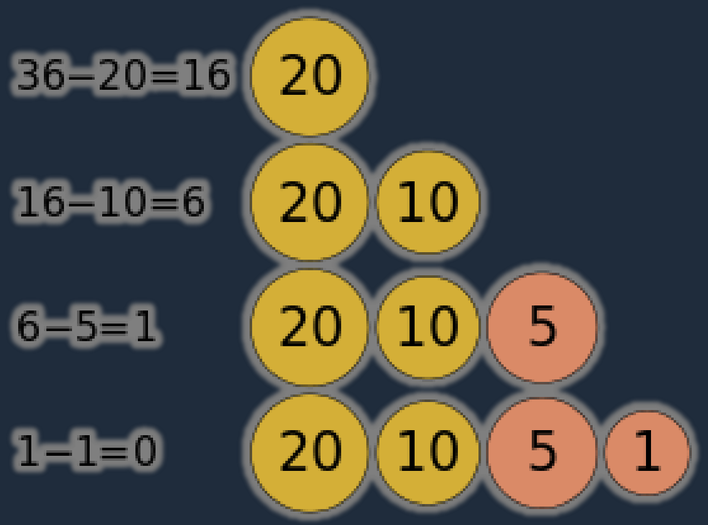
PRESENTED BY: **William Granados**

Terminology



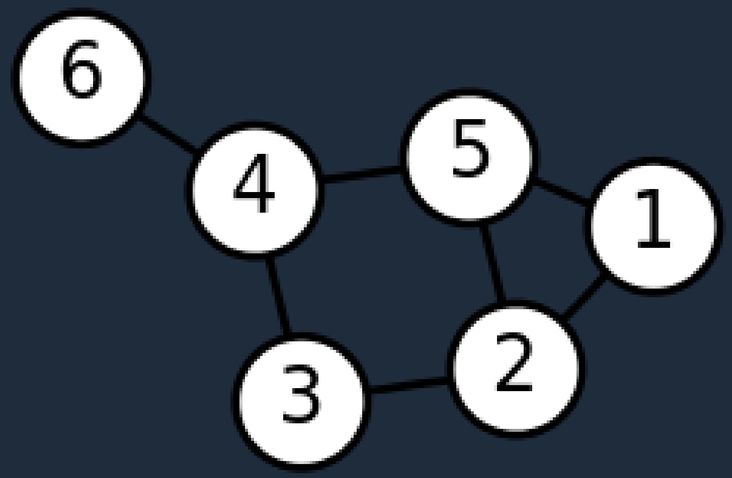
- Given a graph the **spanning tree** is a subgraph that **connects all vertices**. A spanning tree also has **V-1 edges.**
- The **minimum spanning tree** is a subgraph where the edge weights are minimized
- A graph is **connected** if there is a path between all edges in the graph

Terminology Continued



- An algorithm is considered **greedy** if it makes the locally optimal choice which may or may not find the most optimal solution to a problem
- An example, try to make change for $36 using the standard Canadian coin denominations.
- Note that **greedy** algorithms may not always work, can you think of a twist to the coin change problem which will break your strategy? (Hint: use different denominations)

What does Prim's algorithm do?

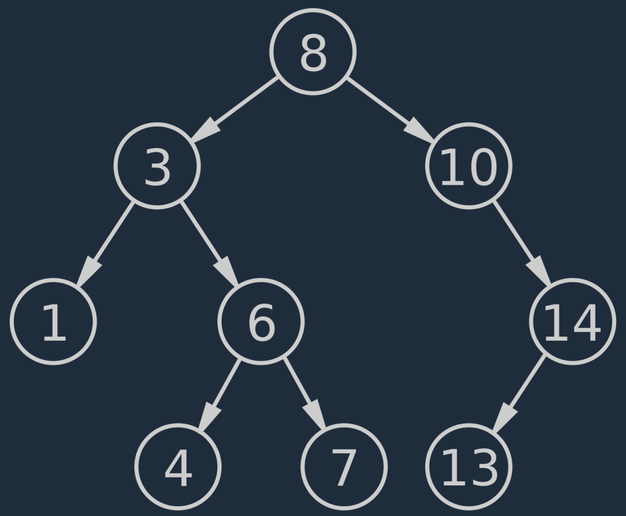


Prim's Algorithm is a **greedy algorithm** that finds the **minimum spanning tree** for a **connected** weighted undirected graph(Wiki). The implementation in this presentation will do so in O(|E| log |V|) making use of a **priority queue** and an **adjacency list**.

Important: Prim's algorithm does not work on a **disconnected** graph. The algorithm that does work is known as **Kruskal's** algorithm, and it instead creates a **minimum spanning forest**

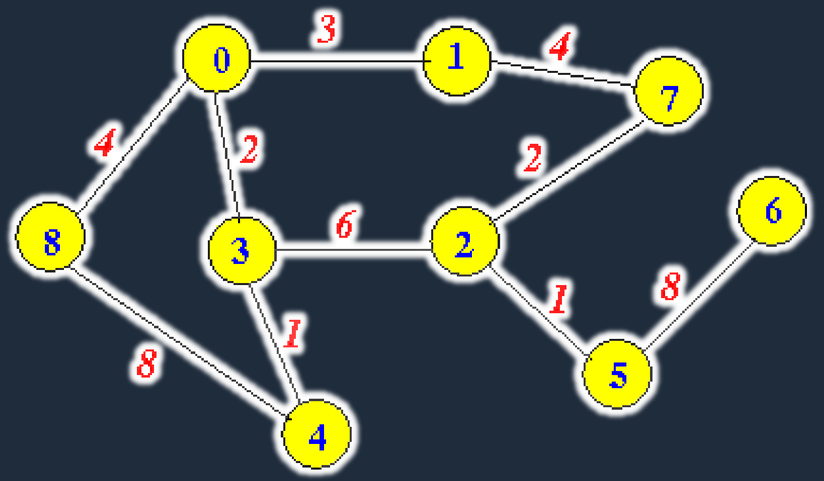
Main Idea for Prim's Algorithm

Similar to a Tree..?



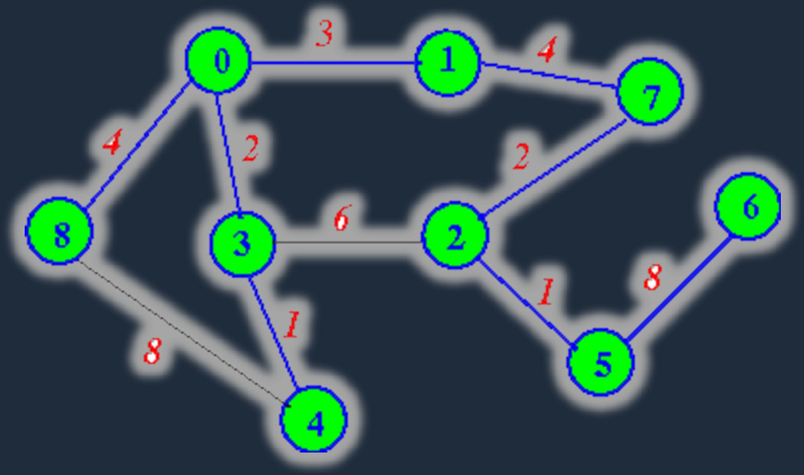
1. Start at any vertex and mark this vertex as visited. Then mark all other vertices as unvisited
2. Find the smallest edge within the graph that connects the current vertex to a new vertex that is not already in the minimum spanning tree
3. Once this edge is found mark the new vertex as visited and add the edge's weight to the current weight for the minimum spanning tree
4. Repeat steps 2 and 3 until there are V-1 edges in the minimum spanning tree

Sample Graph



Try to find the minimum spanning tree for the following graph. Hint: Think greedy

Sample Graph Solution



Edges that were selected are highlighted in dark blue were added to our MST

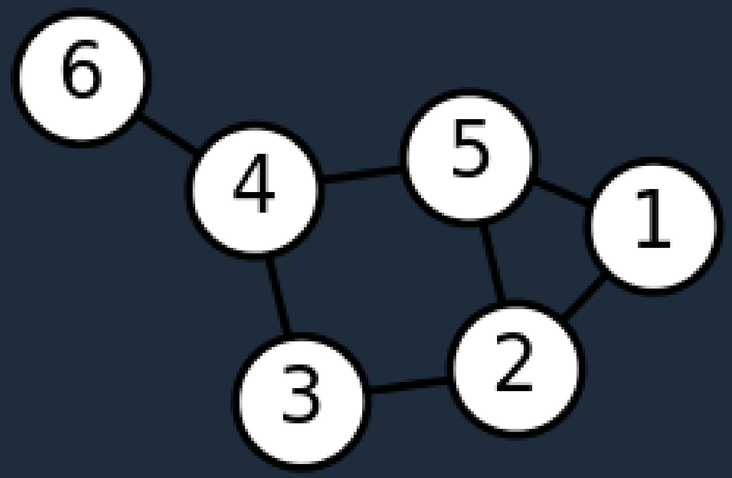
Pseudocode

```
int prim(){
        create empty min heap of edges //the edge with the smallest weight is always at the top
        set number of edges to 0
        set total weight to 0
        for each edge connected to the 0th node
                push current edge to heap

        set all nodes to not visited
        set first node to visited

        while (number of edges < number of nodes-1 and priority queue not empty){
                take top edge from min heap
                if (the start node has not been visited or the end node has been visited)
                        continue
                set current node to visited
                increment number of edges
                add current edge's weight to total weight
                for each of the end nodes edges
                        push edge to heap
        }
        return total weight
}
```

C++ Code

```
int prim(){
	priority_queue<edge>pq; // (edge){weight,start node,end node}
	int MSTEdges = 0, totalWeight = 0;
	for(int i = 0; i < adj[0].size();i++){// arbitrarily choose first node to start prim's algorithm from, 0 is the first node;
		pq.push((edge){adj[0][i].second,0,adj[0][i].first});// adjacency list edge {end node,cost}
	}
	memset(visited,false,sizeof visited);// set all nodes to not visited
	visited[0] = true;// set first node to visisted
	while(MSTEdges < N-1 && !pq.empty()){// Minimum spanning trees have N-1 edges
		int w = pq.top().w;
		int sn = pq.top().sn;
		int en = pq.top().en;
		pq.pop();
		if(visited[sn] && !visited[en]){
			visited[en] = true;
			MSTEdges++;
			totalWeight+=w;
			for(int i = 0; i < adj[en].size();i++)
				pq.push((edge){adj[en][i].second,en,adj[en][i].first});// adjacency list edge {end node,cost}
		}
	}
	return (MSTEdges == N-1) ? totalWeight:-1;// if we have less than N-1 edges then that means the graph is disconnected
}
```

Practice Problems, try them out!



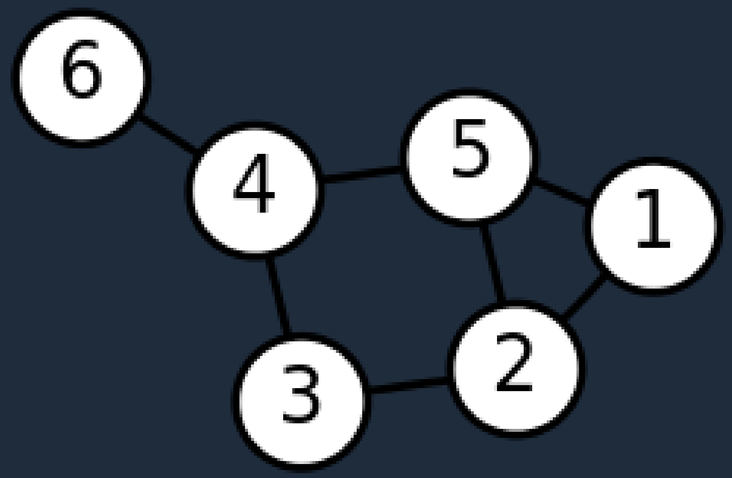
Time to get Medieval
http://wcipeg.com/problems/desc/wc01p8

Trucking Troubles
http://wcipeg.com/problems/desc/ccc03s5

Cable TV
http://wcipeg.com/problems/desc/graph3p3

Animal Farm
http://wcipeg.com/problems/desc/ccc10s4

Sample Solution for Time to Get Medieval



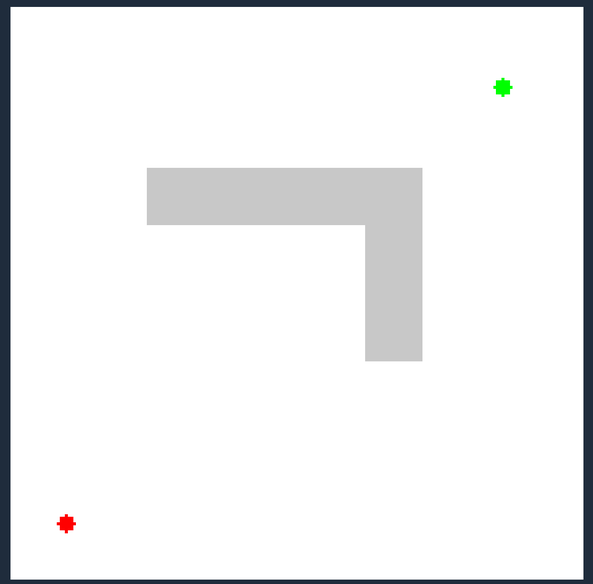
Python

https://github.com/csecutsc/ProblemSetSolution/blob/master/medieval.py

C++

https://github.com/csecutsc/ProblemSetSolution/blob/master/medieval.cpp

Sources



(n.d.). Retrieved February 12, 2015, from http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/11-Graph/prim2.html

Algorithm of the Week: Graphs and their Representation. (2012, April 9). Retrieved February 12, 2015, from http://java.dzone.com/articles/algorithm-week-graphs-and

Prim's algorithm. (n.d.). Retrieved February 12, 2015, from http://en.wikipedia.org/wiki/Prim's_algorithm