Prim's algorithm

Equipe: Daniel Diniz, Paulo Bernardo e Ricardo Alves

https://github.com/DataStructureProject2018

Problem

Let's suppose we need to connect 10 cities with wires for electricity, but we don't want to spend any unnecessary material along the way.

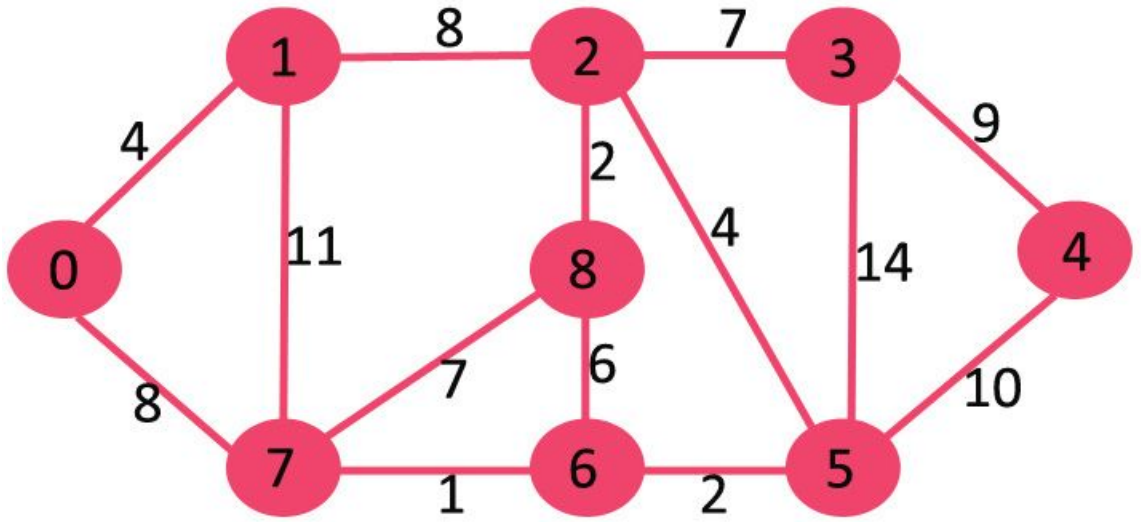
Instead of wiring every city with the source of energy, we put wires in the shorter path that passes through all cities.

Doing so we avoid spending any wire that's more than necessary.

We can do that with Prim's algorithm!

Prim's Algorithm

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph.



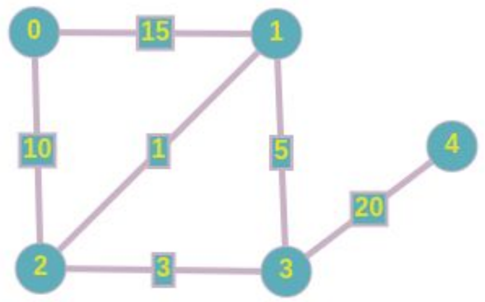
Definitions:

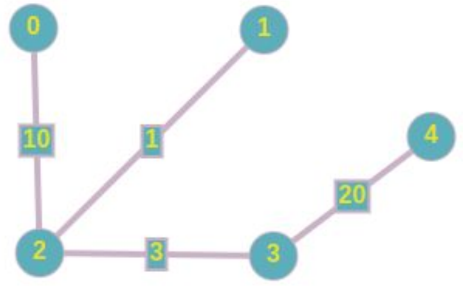
A greedy algorithm: it's a project technique of algorithms that tries to solve a problem by doing great local choices in each stage with hope of finding a great global optimum.

Minimum spanning tree (MST): is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. That is, it is a spanning tree whose sum of edge weights is as small as possible.

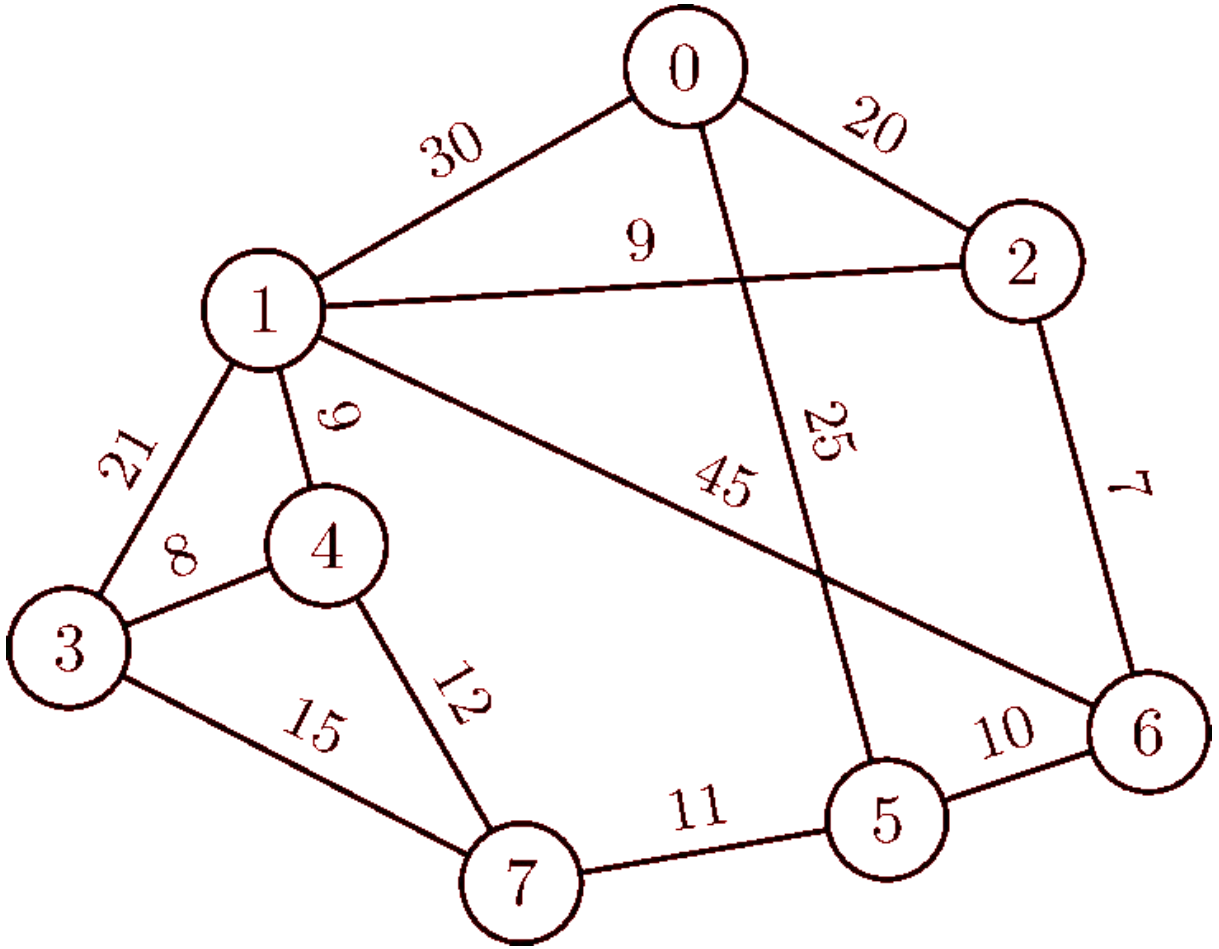
Weighted Undirected Graph: A Graph with no orientation on its edges and with a weight attached to each edge. The weights are often numbers.

Prim’s Code





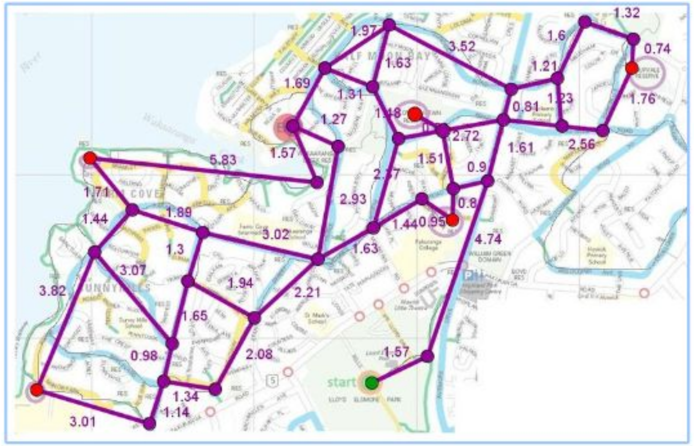
```
int primMST(graph *graph, int V){
      int parent[V];     // Array para guardar a MST
      int key[V]; // Valores chaves para usar as arestas de menor peso
      int i, count, cost = 0;
      for (i = 0; i < V; i++){
            key[i] = INT_MAX;
      }
      key[0] = 0; // Faz o vértice 0 ser o primeiro escolhido
      parent[0] = 0; // Faz o vértice 0 estar sempre ligado com ele mesmo

      priority_queue *queue = create_queue();
      enqueue(queue, 0, 0);

      while (queue->head){
            node *u = dequeue(queue);
            adj_list *current = graph->vertex[u->vertex].head;

            if (graph->visited[u->vertex] == false){
                  while(current){
                        if(graph->visited[current->vertex] == false &&
                        current->weight < key[current->vertex]){
                              parent[current->vertex] = u->vertex;
                              key[current->vertex] = current->weight;
                        }
                        enqueue(queue, current->vertex, current->weight);
                        current = current->next;
                  }
                  graph->visited[u->vertex] = true; // Marca o vértice como visitado
                  cost += u->weight; // Soma o custo da aresta que foi para u
            }
      }
      return cost;
}
```


0
1
2
3
4
5
6
7
30
20
9
25
21
6
45
7
8
12
15
10
11

That's it!

How did prim's algorithm fixed our previous problem?

Using prim's we've passed in every city going always to the nearest unvisited city until we wired them all, not wasting more material than necessary.



Applications

- With Prim´s we can find minimum spanning trees (MST).
- Travelling salesman problem (approximately result).

And with MST we have direct applications such as:

- Design of networks (computer networks, transportation networks, water supply networks, and electrical grids).
- Taxonomy.
- Circuit design (implementing efficient multiple constant multiplications).
- Minimum spanning trees can also be used to describe financial markets.
- And many more...