

Lecture 29: Graph Algorithms

Agenda:

- Prim's MST algorithm
- Kruskal's MST algorithm

Reading:

- Textbook pages 561 – 579

Prim's algorithm for the MST problem (recall):

- Input: an edge-weighted (simple, undirected, connected) graph (positive weights)
- Output: an MST
- Idea:
 - suppose we have already an MST T' spanning subset V' of vertices (T' is initialized empty and V' is initialized to contain any one vertex)
 - grow T' to span one more vertex $v \in V - V'$
 - v is selected such that there is a vertex $u \in V'$, edge (u, v) is the minimum weighted over all edges of form (u', v') where $u' \in V'$ and $v' \in V - V'$
 - when V' becomes V , terminate
- One simplest implementation:

```

procedure primMST( $G, w, r$ )           ** $G = (V, E)$ 

 $S \leftarrow \{r\}$                        ** $S$ : spanned vertex subset
 $T \leftarrow \emptyset$                    ** $T$ : MST spanning  $S$ 
while  $|S| < |V|$  do
    find a minimum weight edge  $(u, v)$ :  $u \in S$  and  $v \in V - S$ 
     $S \leftarrow S \cup v$ 
     $T \leftarrow T \cup (u, v)$ 
return  $T$ 
    
```

Running time analysis:

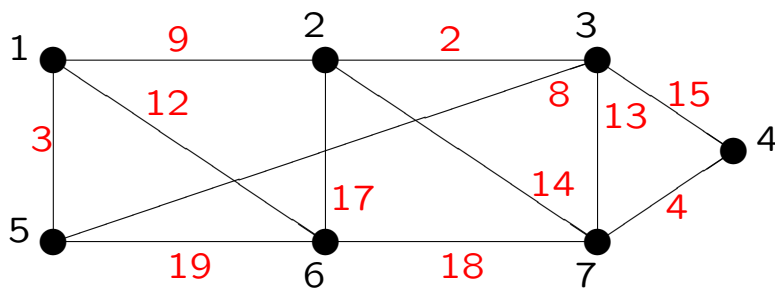
1. finding such an edge in $O(n^2)$ (or $O(m)$) time
2. there are $n - 1$ edges in the output MST
3. therefore, in total $O(n^3)$ (or $O(nm)$) time

Prim's algorithm for the MST problem — correctness:

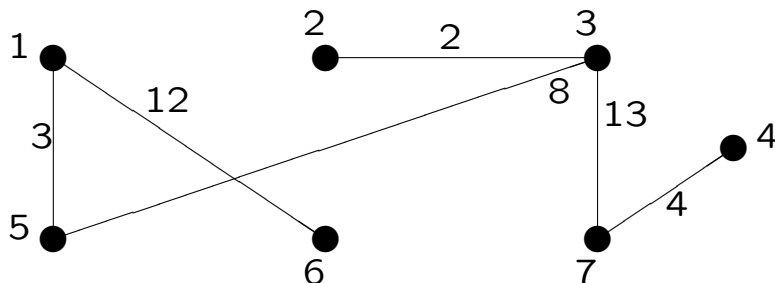
- Input graph $G = (V, E)$: $E = \{e_1, e_2, \dots, e_m\}$
- Suppose edges in the output tree T are $e_{i_1}, e_{i_2}, \dots, e_{i_{n-1}}$ (in the order picked by Prim's algorithm)
- Want to prove: T is an MST
- Suppose T' is an MST and it contains edges $e_{j_1}, e_{j_2}, \dots, e_{j_{n-1}}$ (sorted in the way that it maps the edge order in T as much as possible). If $T \neq T'$ (otherwise we are done), then
 - there is a minimum index k , such that $e_{j_k} \neq e_{i_k}$
 - let T_0 denote the tree formed by $\{e_{i_1}, e_{i_2}, \dots, e_{i_{k-1}}\}$
 - let $V_0 = V(T_0)$ and $V_1 = V - V_0$
 - adding e_{i_k} to T' creates a cycle which contains some edge, say e_{j_p} , that has one ending vertex in V_0 and the other in V_1
 - $T'' = T' + e_{i_k} - e_{j_p}$ is another spanning tree
 - T'' is another MST (why ?) sharing one more edge with T
 - repeat this argument to claim that T is also an MST
- Note: this is a proof using 'contradiction' + 'graph theory'.
- Proof can also be done by
(while) Loop Invariant: T is an MST on S .
Exercise !

Prim's algorithm for the MST problem — improvement:

- Where to improve: finding the minimum weight edge (u, v)
- Initially we need to scan all the edges, $\Theta(m)$ (worst case)
- Example: input graph G :

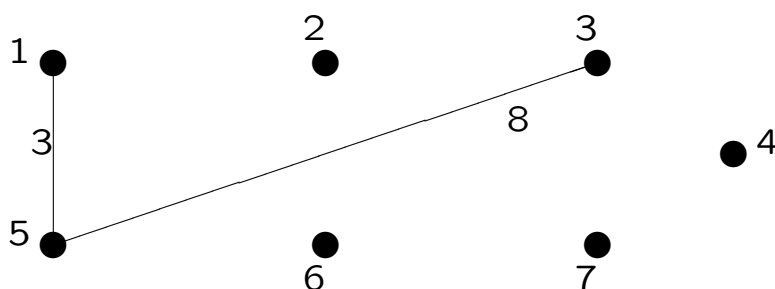


- $\text{primMST}(G, w, 1)$ returns:



- $\text{primMST}(G, w, 1)$: an intermediate tree

What are the candidate edges ?



Prim's algorithm for the MST problem — improvement:

- Ideas:

- for each non-tree vertex v , store its minimum-weight tree neighbor $p[v]$
- store edges of type $(p[v], v)$ in a min-priority queue Q
- therefore, every time the target edge can be extracted $\text{ExtractMin}(Q)$
note: need to update the neighbor information for non-tree vertices after the extraction

- Pseudocode:

```

procedure primMST( $G, w, r$ )                                ** $G = (V, E)$ 

  for each  $v \in V(G)$  do
     $key[v] \leftarrow \infty$ 
     $p[v] \leftarrow \text{NIL}$ 
   $key[r] \leftarrow 0$ 
   $Q \leftarrow V(G)$ 
  while  $Q \neq \emptyset$  do
     $u \leftarrow \text{ExtractMin}(Q)$                                 ** $r$  dequeued first
    for each  $v \in \text{Adj}[u]$  do
      if  $v \in Q$  &&  $w(u, v) < key[v]$  then                    **update  $v$ 
         $p[v] \leftarrow u$ 
        decrease-key( $v, w(u, v)$ )                            ** $key[v] \leftarrow w(u, v)$ 

```

- Analysis:

- correctness (almost done — need to prove that $\text{ExtractMin}(Q)$ does extract the minimum weight edge)

- running time: $\Theta\left(n \log n + \sum_{u \in V} \left(\text{degree}(u) \times \log n\right)\right)$
so: $\Theta(m \log n)$ — adjacency list graph representation

Kruskal's algorithm for the MST problem:

- Input: an edge-weighted (simple, undirected, connected) graph (positive weights)
- Output: an MST
- Idea:
 - suppose we have already an acyclic subgraph T
 - grow T to by including one more edge e
 - edge e is selected such that
 1. maintaining 'acyclic'
 2. of minimum weight
 - when T contains $n - 1$ edges (and thus becomes a spanning tree), terminate
- One implementation using DSUF:

```

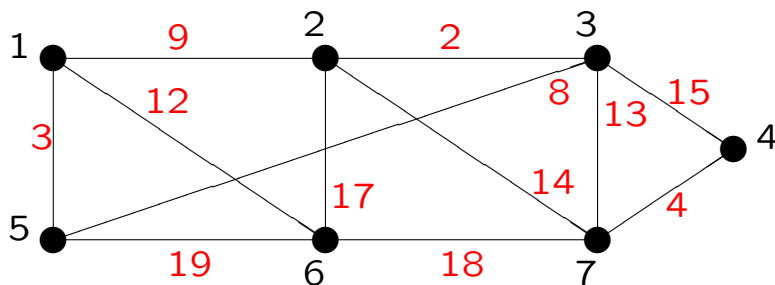
procedure kruskalMST( $G, w$ )           ** $G = (V, E)$ 

 $T \leftarrow \emptyset$                   ** $T: (V, T)$  acyclic
for each  $v \in V(G)$  do
    MakeSet( $v$ )
sort edges in  $E(G)$  into non-decreasing weight order
for each  $(u, v) \in E(G)$  do           **taken in order
    if cFind( $u$ )  $\neq$  cFind( $v$ ) then **( $u, v$ ) is 'safe'
         $T \leftarrow T + (u, v)$ 
        rUnion( $u, v$ )
return  $(V, T)$ 
    
```

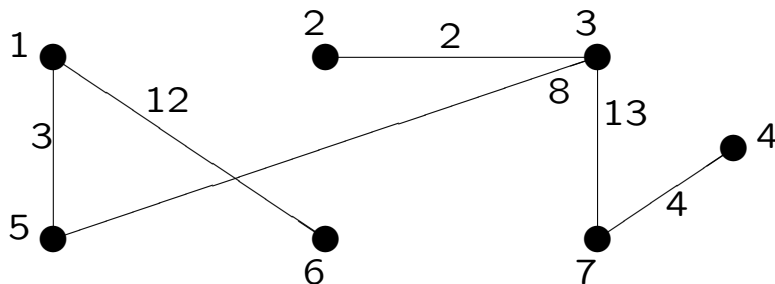
Kruskal's algorithm for the MST problem — analysis:

- Correctness:
 - by Loop Invariant: what is the loop invariant ???
 - by 'contradiction' + 'graph theory'
- Running time analysis (adjacency list graph representation):
 1. sorting edges $\Theta(m \log n)$
 2. $\Theta(m)$ rUnion/cFind: $O(m\alpha(n))$
 3. therefore, in total $\Theta(m \log n)$ time

- An example:



- `kruskalMST( $G, w$ )` returns:



Lecture 29: Graph Algorithms

Have you understood the lecture contents?

well	ok	not-at-all	topic
☐	☐	☐	Prim's algorithm: correctness
☐	☐	☐	improved implementation and analysis
☐	☐	☐	Kruskal's algorithm: idea
☐	☐	☐	correctness
☐	☐	☐	DSUF implementation, execution, & analysis