CISC 320 Introduction to Algorithms Fall 2005

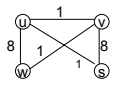
Lecture 14
Minimum Spanning Tree
and
Greedy Algorithms

Definitions

- Spanning tree: given a connected, undirected graph G=(V,E), a spanning tree is a subgraph of G that is an undirected tree and contains all the vertices of G.
- Minimum spanning tree (MST): is a spanning tree with minimum weight. The weight of a subgraph is the sum of the weights of the edges in the subgraph.

How to find MST?

- Using depth-first or breadth-first search to traverse the graph will yield a spanning tree, but the found spanning tree is not guaranteed to be a MST.



- Need a different scheme to traverse the graph

Optimization problem & Greedy approach

- Pick a starting point randomly
- "grow" step by step, each step is the best among all possible choices
- Stop when a stopping criterion is satisfied

Note: being greedy in a short term may NOT lead to the overall best solution

Growing a MST

Definitions

- Tree vertices: vertices that are in the tree constructed so far
- Fringe vertices: not in the tree, but adjacent to some tree vertices
- Unseen vertices: all others.

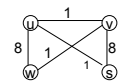
MST-Prim(G,w)

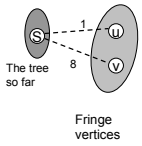
1. Initialize all vertices as unseen
2. Select an arbitrary vertex ***s*** to start the tree, mark ***s*** as tree vertex
3. Mark all vertices adjacent to ***s*** as fringe
4. While there are fringe vertices
5. select an edge of minimum weight between a tree vertex ***t*** and a fringe vertex ***v***
6. mark ***v*** as tree and add edge (***tv***) to the tree
7. mark all unseen vertices adjacent to ***v*** as fringe

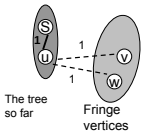
Managing the Fringe with a priority queue

```
MST-Prim(G, w, r)
1.  for each u ∈ V[G]                          } O(V) to build a binary heap
2.      key[u] ← ∞                             }
3.      P[u] ← nil                             }
4.  key[r] ← 0                                 }
5.  Q ← V[G]
6.  while Q is not empty                       // |v| times
7.      u ← Extract-Min(Q)                     // O(log(v))
8.      for each v ∈ Adj[u]                    // O(E), combined with line 6
9.          if v ∈ Q and w(u,v) < key[v]
10.             then p[v] ← u
11.                 key[v] ← w(u.v)            // O(log(v)) time to prioritize v in Q
```

Total running time: O(V lg V + E lg V)







Definitions

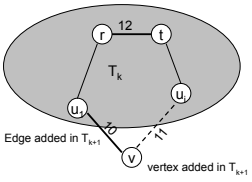
- Minimum spanning tree property: Let T be any spanning tree of a connected, weighted graph G. For any edge uv of G that is not in T, if uv is added to T it creates a cycle and uv is a maximum-weight edge on that cycle, then T has the *minimum spanning tree property*.

Theorem In a connected, weighted graph G = (V,E,W), a spanning tree T is a MST ***iff*** T has the MST property.

Correctness of Prim's MST algorithm

- Each time an edge from a tree vertex to a fringe vertex is added into the tree, so never is a cycle created.
- All vertices will be added to the tree eventually.
- Therefore the final tree is a spanning tree.
- It is also a minimum spanning tree, because
  - At each step, the so far constructed tree has the MST property in its induced graph of G.

- Proof by induction:
  - Let T_k be the tree in k-th step. Let G_k be the subgraph of G induced by T_k (i.e., uv is an edge in G_k if it is an edge in G and both u and v are in T_k)
  - T_1 has MST property of G_1.
  - Let assume it is true up to arbitrary k>0.
  - At step k+1, vertex v is added, and v has edge with some vertices u_1, …, u_d in T_k. For definiteness, assume vu_1 is the edge of minimum weight among all possibilities. Now $T_{k+1} = T_k + vu_1$, and $G_{k+1} = G_k + vu_1 + \ldots + vu_d$.
    To prove T_{k+1} is MST of G_{k+1}, we need to prove, for any edge xy in G_{k+1} but not in T_{k+1}, if add xy to T_{k+1}, xy will be the maximum weight edge in the cycle thus created.
    See next slide for an example.
  - At step |V|, tree $T_{|V|}$ contains all vertices in G, and G itself is the induced graph by $T_{|V|}$. Therefore, Spanning tree $T_{|V|}$ is the MST of G.



It is impossible to have edge rt such that W(rt) > W(vu_i)

Suppose t is added in later than r, then Prim algorithm will add edge (u_x,v) or (u_y,v) instead of (rt), since v is in the fringe and has an edge of smaller weight.

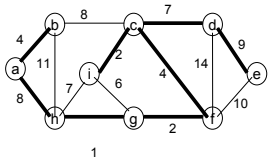
Kruskal MST Algorithm

MST-Kruska(G, w)

1. Initialize set A as empty // to store a forest of trees
2. Build a min priority queue Q of edges of G, prioritized by weight w
3. Initialize a union-find structure, sets, in which each vertex of G is in its own set.
4. while Q is not empty
5. (u,v) = Extract-Min(Q)
6. if (FIND-Set(u) ≠ FIND-Set(v)) // sure v and w not in the same tree
7. A ← A ∪ (u,v) // add edge (u,v) to A
8. Union(u,v)
9. return A // this is a MST

NOTE: if there are degeneracy in edge weights, MST will not be unique, depending on how ties are resolved in the priority queue.

Kruskal algorithm: example



w	Edge	w	Edge
1	(gh)	7	(cd) (hi)
2	(ci) (fg)	8	(ah) (bc)
4	(ab)	9	(de)
5	(cf)	10	(ef)
6	(gi)	11	(bh)

Edge	Action	Sets
(gh)	add	{a}, {b}, {c}, {d}, {e}, {f}, {g,h}, {i}
(ci)	add	{a}, {b}, {c,i}, {d}, {e}, {f}, {g,h}
(fg)	add	{a}, {b}, {c,i}, {d}, {e}, {f,g,h}
(ab)	add	{a,b}, {c,i}, {d}, {e}, {f, g, h}
(cf)	add	{a,b}, {c,f,g,h,i}, {d}, {e}
(gi)	reject	
(cd)	add	{a,b}, {c,d,f,g,h,i}, {e}
(hi)	reject	
(ah)	add	{a,b,c,d,f,g,h,i}, {e}
(de)	add	{a,b,c,d,e,f,g,h,i}

Kruskal algorithm

Time analysis

- Initialization: O(V) Make-Set for each vertex
- Deleting all edges from the queue: Θ(E log E)
- Find-Set called 2|E| times, Union called |V| times. The cost is O((E + V) α(V)), where α(V) is a very slowly growing function defined in Section 21.4.

  Since the graph is assumed to be connected, E≥ V-1. Therefore the cost is O(E α(V)).

Total worst-case time: Θ(E log E)