CS F364
Design & Analysis of Algorithms

Algorithm Design: Greedy Technique

Minimum Spanning Trees

Properties and A Greedy Algorithm

CSIS, BITS, Pilani   Sundar B.
3/3/2016

Minimum Spanning Trees

- Theorem:
  - Let $G = (V,E,w)$ be a connected graph. Let V_1 and V_2 form a partition of V i.e. $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \{ \}$
  - If ***e*** is the edge with minimum weight among those with one end in V_1 and the other in V_2 ,
    - then there is a minimum spanning tree with ***e*** as one of its edges.
- Question:
  - What is the implication of the theorem?

MINIMUM SPANNING TREES

- Theorem:
  - Let $G = (V,E,w)$ be a connected graph. Let V_1 and V_2 form a partition of V i.e. $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \{ \}$
  - If ***e*** is the edge with minimum weight among those with one end in V_1 and the other in V_2 ,
    - then there is a minimum spanning tree with ***e*** as one of its edges.
- Proof (by contradiction):
  - Let T be an MST without ***e***, the min. edge bet. V_1 and V_2
    - Addition of ***e*** to T would create a cycle i.e.

      $\exists$ edge ***f*** in T with one end in V_1 and the other in V_2
    - But $w(e) <= w(f)$
    - If we remove ***f*** from T U { ***e*** } we get a spanning tree T' with total weight no more than that of T.
      - Contradiction unless T' is also an MST.

MINIMUM SPANNING TREES

- Corollary:
  - Minimum Spanning Tree problem satisfies optimal sub-structure property.
    - i.e. if $G = (V,E,w)$ is partitioned as in the Theorem,
    - then the MST for G would include the MSTs for G_1 and G_2 induced by V_1 and V_2 respectively, and the minimum edge between V_1 and V_2 .

Minimum Spanning Trees

- Greedy Choice:
  - Given minimum spanning trees for two sub-graphs, (locally) choosing a minimum edge between the sub-graphs
    - will allow the combination of minimum sub-spanning trees into a minimum spanning tree for the whole graph.

Minimum Spanning Trees

- Kruskal's algorithm:
  - Uses a greedy approach based on the Corollary (last slide)
  - Build the spanning tree in clusters.
    - Initially each vertex is in its own cluster
    - Consider each edge, in increasing order of weight:
      - If the edge ***e*** connects two different clusters,
      - then add ***e*** to the spanning tree and merge the clusters
      - else discard ***e***
    - Algorithm terminates when there are sufficient edges (i.e. the tree spans the graph)

Minimum Spanning Trees – Kruskal's Algorithm

- Input: simple, connected, weighted graph G = (V,E)

```
for each u in V   define  cluster C[u] = { u }
Let Q be a priority queue with all edges in E in increasing
   order of weights.
T = { }   // tree represented as a set of edges
while  (|T| < n-1)  {
     (u,v) = min(Q);     Q = deleteMin(Q);
     Let C[u] be the cluster containing  u and
           C[v] be the cluster containing  v
      if (C[u] != C[v]) then {
          T = T U { (u,v) }
           C[u] = C[v] =  C[u] U C[v]
       }
}
return T
```