CS F364
Design & Analysis of Algorithms

Algorithm Design: Greedy Technique

Minimum Spanning Trees

Kruskal's Algorithm – Implementation Issues and Analysis

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

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
```

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 e in E in increasing order of
    weights.
T = { }
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
```

Time Complexity:

- O(m) for heap construction;
- O(m*logm+L(m,n))) for the loop
- L(m,n) – cost of all clustering operations

m = |E|

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 e in E in increasing order of weights.

T = { }

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

Time Complexity:
O(m*logm+L(m,n)))
L(m,n) – cost of all cluster operations

- **Cluster – unordered linked list of vertices;**
    - **each vertex has a reference to the cluster;**
- **Merging (i.e. union) of clusters:**
    - **add elements of smaller cluster to larger one.**

Minimum Spanning Trees – Kruskal's Algorithm

- Time Complexity:
  - O(m*logm + L(m,n)) where L is the cost of all clustering operations
  - L(m,n)
    - Represent clusters as unordered linked lists of vertices
      - Each vertex holds a pointer to the head of the list
    - m comparisons – each of O(1) cost
    - n merge operations
      - Each costs O(min(|C[u]|,|C[v]|) if smaller list is appended to the larger list.
      - Total cost = nlogn
        - Why?
    - Total cost is L(m,n) = m+n*logn
  - Total complexity: O(m*logm + n*logn)