G205 Fundamentals of Computer Engineering

CLASS 15, Mon. Nov. 1 2004

Stefano Basagni

Fall 2004

M-W, 1:30pm-3:10pm

A Problem on Graphs

- A town has a set of houses and a set of roads
- A road connects 2 and only 2 houses
- A road connecting houses u and v has a repair cost w(u,v)
- **Goal:** Repair enough (and no more) roads such that
  1. everyone stays connected: can reach every house from all other houses, and
  2. total repair cost is minimum

Model as a Graph

- Undirected graph $G = (V, E)$
- **Weight** $w(u,v)$ on each edge $(u,v) \in E$
- Find $T \subseteq E$ such that
  1. T connects all vertices (T is a **spanning tree**)
  2. $w(T) = \text{SUM}_{((u,v) \in T)} w(u,v)$ is minimized
- A spanning whose weight is minimum over all spanning trees is called a **minimum(-weight) spanning tree**, or **MST**

Growing a MST

- Some properties of a MST
  - It has |V| - 1 = n – 1 edges
  - It has no cycle
  - It might not be unique

Building Up a Solution

- We will build a set A of edges
- Initially, A has no edges
- As we add edges to A, maintain a loop invariant:

  Loop invariant: A is a subset of some MST
- Add only edges that maintain the invariant
- If A is a subset of some MST, an edge (u,v) is **safe** for A if and only if $A \cup \{(u, v)\}$ is also a subset of some MST (add only safe edges)

Generic MST algorithm

GENERIC-MST(G,w)

A = 0

while A is not a spanning tree **do**

find an edge (u, v) that is safe for A

A = A ∪ {(u, v)}

return A

Correctness

- We use the loop invariant
- **Initialization:** The empty set trivially satisfies the loop invariant
- **Maintenance:** Since we add only safe edges, A remains a subset of some MST
- **Termination:** All edges added to A are in an MST, so when we stop, A is a spanning tree that is also an MST

Finding a Safe Edge

- A cut $(S, V \setminus S)$ of an undirected graph G is a partition of V
- An edge (u,v) crosses the cut $(S, V \setminus S)$ if one of its endpoints is in S and the other in $V \setminus S$
- A cut respects a set of edges A if no edge in A crosses the cut
- An edge is a light edge crossing the cut if its weight is the minimum among all those that cross the cut

Recognizing Safe Edges

- Theorem: Let $G=(V,E)$ be a connected, undirected graph, and $w:E \to \mathbf{R}$.

  Let $A \subseteq E$ included in some MST of G.
  Let $(S, V \setminus S)$ any cut of G that respects A and let (u,v) be a light edge crossing $(S, V \setminus S)$.

  Then edge (u,v) is safe for A

Analysis of GENERIC-MST

- A is a forest containing connected components. Initially, each component is a single vertex
- Any safe edge merges two of these components into one. Each component is a tree.
- Since an MST has exactly |V|-1 edges, the **for** loop iterates |V|-1 times. Equivalently, after adding |V|-1 safe edges, we are down to just one component

Kruskal's Algorithm for MST

- G = (V, E) is a connected, undirected, weighted graph. $w : E \to \mathbf{R}$
  - Starts with each vertex being its own component
  - Repeatedly merges two components into one by choosing the light edge that connects them (i.e., the light edge crossing the cut between them)
  - Scans the set of edges in monotonically increasing order by weight
  - Uses a disjoint-set data structure to determine whether an edge connects vertices in different components

The Algorithm

```
KRUSKAL(V,E,w)
  A=0
  for each vertex v ∈ V do MAKE-SET(v)
  sort E into non-decreasing order by weight w
  for each (u,v) taken from the sorted list do
    if FIND-SET(u) ≠ FIND-SET(v)
      then A=A∪{(u, v)}
                UNION(u,v)
  return A
```

Analysis of Kruscal, 1

- Running time of Kruscal depends on implementation of disjoint-set data structure
- Main operations:
  - Initialize A: O(1)
  - First **for** loop: |V| MAKE-SETs
  - Sort E: O(E log E)
  - Second **for** loop: O(E) FIND-SETs and UNIONs

Analysis of Kruskal, 2

- The |V| MAKE-SETs and the FIND/UNIONs takes $O(V+E\ \alpha(V))$
- $\alpha(V)$ grows very slowly
- G connected → $|E| \geq |V|-1$ → disjoint-set operations take $O(E\ \alpha(V))$
- $\alpha(|V|)=O(\log V)=O(\log E)$ → Kruskal is in $O(E \log E)$ which is $O(E \log V)$

Prim's Algorithm for MST

- Builds one tree, so A is always a tree
- Starts from an arbitrary "root" r
- At each step, find a light edge crossing cut $(V_A, V \setminus V_A)$, where V_A = vertices that A is incident on
- Add this edge to A

Selecting Edges Efficiently

- Use a priority queue Q:
  - Each object is a vertex in $V \setminus V_A$
  - Key of v is minimum weight of any edge (u,v), where $u \in V_A$
  - The vertex returned by EXTRACT-MIN is v such that there exists $u \in V_A$ and (u,v) is a light edge crossing $(V_A, V \setminus V_A)$
  - Key of v is ∞ if v is not adjacent to any vertices in V_A

Prim's MST

- The edges of A will form a rooted tree with root r:
  - r is given as an input to the algorithm, but it can be any vertex
  - Each vertex knows its parent in the tree by the attribute $\pi[v]$ = parent of v. $\pi[v]$ = NIL if v = r or v has no parent
  - As algorithm progresses, $A = \{(v, \pi[v]) : v \in V \setminus \{r\} \setminus Q\}$
  - At termination, $V_A = V \Rightarrow Q = 0$, so MST is $A = \{(v, \pi[v]) : v \in V \setminus \{r\}\}$

Prim, the Algorithm

```
PRIM(G,w,r)
  for each u ∈ V do key[u]=∞; π[u]=NIL
  key[r]=0; Q=V
  while Q≠0 do
    u=EXTRACT-MIN(Q)
    for each v ∈ Adj[u] do
      if v ∈ Q and w(u,v) < key[v]
        then π[v]=u
             key[v]=w(u, v)
```

Assignments

- Textbook, Chapter 23, pages 561—574
- Updated information on the class web page:

  www.ece.neu.edu/courses/eceg205/2004fa