Spanning Trees

Problems

- power, water distribution
- network connectivity
- wiring, VSLI
- image segmentation

number of edges?
cycles?

Graph Cuts

In a graph, a *cut* is a partitioning of all the graph's vertices into two disjoint sets.

Many algorithms are concerned with edges that *cross* the cut – edges with their two endpoints in different sets.

Basic Approach

Starting from $\varnothing$, grow spanning tree by adding edges.

Theorem: take any cut that respects the growing tree. A lightest edge crossing the cut can be added to the tree.

Proof: if a MST T includes our edge, our choice was a good one. Otherwise, consider an edge in T that crosses the cut. Substitute it with the lightest one and the cost of the MST can't go up.

Basic Approach

Starting from $\varnothing$, grow spanning tree by adding edges.

Theorem: take any cut that respects the growing tree. A lightest edge crossing the cut can be added to the tree.

Proof: if a MST T includes our edge, our choice was a good one. Otherwise, consider an edge in T that crosses the cut. Substitute it with the lightest one and the cost of the MST can't go up.

Kruskal's Algorithm

Kruskal's Algorithm

Connect separate components until the vertices are spanned.

```
T ← ∅
for all vertex v do
    MAKE-SET(v)
for all edges (u, v) in sorted order do
    if FIND-SET(u) ≠ FIND-SET(v) then
        add (u, v) to T
        UNION(u, v)
return T
```

correctness?
runtime complexity?

Prim's Algorithm

Prim's Algorithm

grow tree until fully connected

```
foreach vertex v ∈ G, v.c ← ∞, v.π ← nil
start.c ← 0
Q ← min-heap of all vertices (sorted on c)
while Q not empty do
    u ← EXTRACT-MIN(Q)
    for all neighbor v of u do
        if v ∈ Q and w(u,v) < v.c then
            v.c ← w(u,v)
            v.π ← u
return {(v, v.π) : v ∈ V}
```

correctness? invariant?

runtime complexity?