Lecture 28: Monday March 24, 2003

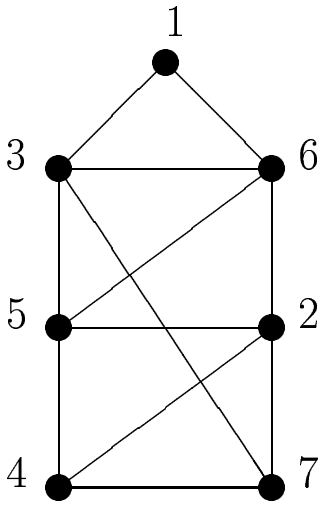
today

- muddytown

announcements

muddytown

- problem: dirt streets, muddy when it rains
- goal: to be able to walk without muddying shoes
- idea: pave enough streets to walk anywhere
- problem: given street paving costs, find min cost paving




```
                          1  2  3  4  5  6  7
1: 3  6                   1
2: 4  5  6  7             2
3: 1  5  6  7             3
4: 2  5  7                4
5: 2  3  4  6             5
6: 1  2  3  5             6
7: 2  3  4                7
```

muddytown = minimum spanning tree [CLRS Ch. 23]

- acyclic graph: no cycles
- tree: connected acyclic graph
- weighted graph: graph together with edge weights (or costs)
- spanning subgraph: includes all vertices of the graph
- spanning tree: spanning subgraph which is a tree
- MST problem: given weighted connected graph,
  find minimum weight spanning tree of the graph
- MSF problem: given weighted graph,
  for each component, find MST of the component

greedy algorithms

- greedy: each step makes best choice (locally optimum)
- in general, greedy solution may not be globally optimum
- for MST problem, two greedy solutions **are** globally optimum

two greedy MST algorithms

- PDB (Prim/Dijkstra/Boruvka)

```
tree <- any vertex
do n-1 times
  add min cost edge e* which grows tree
```

- KB (Kruskal/Boruvka)

```
forest <- all vertices, no edges
do n-1 times
  add min cost edge f* which grows forest
```

- correctness uses graph theory
- different complexities, depend on imp'n

PDB MST simplest implementation

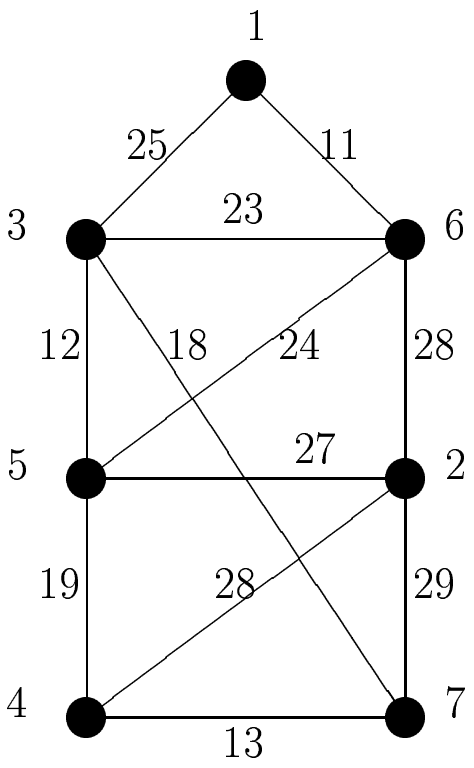
- find e* by examining all neighbours of all vertices
- run time in $\Theta(n^3)$ exercise

PDB MST better implementation

- idea 1: for each non-tree vertex, store min-cost tree neighbour
- idea 2: store values in priority queue
- to find e*:
  - ExtractMin(Q)
  - update 'best-neighbour' data

PDB MST implementation (generic) [CLRS p. 572]

```
MST(G,w,r)           { w: edge weights    r: starting vertex
1 for each u in V[G] do
2   key[u]    <- infty     { min cost tree-neighbour from u
3   parent[u] <- NIL
4 key[r] <- 0          { so r dequeued first
5 Q <- V[G]
6 while not empty(Q) do
7   u <- ExtractMin(Q)
8   for each v in Adj[u] do
9    if v in Q and w(u,v) < key[v] then
10     parent[v] <- u
11     key[v]    <- w(u,v)   { requires Q adjustment
```




```
from 1
          1    2    3    4    5    6    7
Key       0    *    *    *    *    *    *     add   1   to tree
               *   25    *    *   11    *     add (1,6) to tree
              28   23    *   24         *     add (6,3) to tree
              28         *   12        18     add (3,5) to tree
              27        19              18     add (3,7) to tree
              27        13                     add (7,4) to tree
              27                               add (5,2) to tree
```

PDB MST better imp'n: analysis

- assume
  - graph representation: adjacency list
  - priority queue imp'n: heap
- each ExtractMin $O(\lg n)$ total $\Theta(n \lg n)$
- for loop body
  - executed $\Sigma_u \, d(u)$ times total $\Theta(m)$
  - line 11 needs DecreaseKey total $\Theta(m \lg n)$
- total (for connected graph) $\Theta(n \lg n) + \Theta(m \lg n) = \Theta(m \lg n)$

MST algorithm correctness

- loop invariant for both algorithms:
  edges picked form subgraph of some MST [CLRS Thm. 23.1]
- proof, by induction, uses elementary graph theory see text

some graph theory used

- recall tree def'n: connected acyclic graph
- For a connected graph with n vertices,
  1. the graph is a tree (namely connected and acyclic) if and only if
  2. the graph is connected and has $n-1$ edges if and only if
  3. for each x, y there is a unique x, y-path.

proof sketch

- $1 \Longrightarrow 2$: in a tree, the ends of a maximal path have degree one; remove such a vertex, and argue by induction
- $1 \Longrightarrow 3$: suppose not; find a cycle
- $3 \Longrightarrow 1$: suppose not; find two paths
- $2 \Longrightarrow 1$: a connected graph with $m = n-1$ has no vertex with degree 0 (connected) so some vertex with degree 1 (sum the degrees); remove such a vertex, and argue by induction

detailed proof that 1 $\Rightarrow$ 2

- proofs on next page
- how many proofs can you do?

- *maximal* path: not a proper subpath
- *leaf*: degree 1 vertex
- lemma 1: in acyclic graph, end of maximal path has degree ≤ 1
- corollary: acyclic graph has vertex of degree 0 or 1
- lemma 2: tree with degree 0 vertex has only 1 vertex
- lemma 3: removing a leaf from a tree leaves a tree
- theorem: tree with n vertices has exactly $n-1$ edges

- proof of lemma 1: if the end vertex of a path is adjacent to a vertex not in the path, the path is not maximal; if the end vertex p_0 of a path $(p_0, p_1, \ldots)$ is adjacent to a vertex p_j different from p_1, then $(p_0, p_1, \ldots, p_j, p_0)$ is a cycle.

- proof of lemma 2: in a connected graph there is a path between any two vertices, so if there is a vertex of degree 0 in a connected graph, it must be the only vertex in the graph.

- proof of lemma 3: Let T be a tree. Since T is acyclic, $T - v$ is acyclic.

  Let w be the neighbour of v in T. Let x, y be any two vertices of $T - v$. Since T is connected, there is a path $P = (x, \ldots, y)$ in T. Since every vertex in $P - \{x, y\}$ is adjacent to at least 2 path vertices, v is not in $P - \{x, y\}$, so v is not in P, so P is a path in $T - v$, so $T - v$ is connected.

- proof of theorem (by induction):

  base case: the graph with 1 vertex is a tree and has 0 edges

  inductive case: Fix $k \geq 1$ and assume that all trees with k vertices have $k - 1$ edges. Consider a tree T with $k + 1$ vertices. We want to show that T has k edges.

  Since $k \geq 1$, T has at least two vertices, so by lemmas 1 and 2, T has some vertex v with degree 1. By lemma 3, $T - v$ is a tree. $T - v$ has k vertices and (by inductive assumption) $k - 1$ edges. Since v is a leaf, T has exactly one more edge and one more vertex than $T - v$. Thus T has $1 + k - 1 = k$ edges.

KB MST algorithm

```
MST KB(G,w)
1  A <- empty   { edges of MST
2  for each v in V[G] do
3    MakeSet(v)
4  sort edges of G into nondecreasing order by weight w
5  for each edge (u,v) (taken in order) do
6    if Find(u) <> Find(v) then
7      add (u,v) to A
8      Union(u,v)
```

KB MST algorithm: analysis

- assume
  - G connected — so $m \geq n-1$
  - compressFind and rankUnion
- sort edges — total $\Theta(m \lg m) = \Theta(m \lg n)$
  since $m \in O(n^2$ implies $\lg m \in \Theta(\lg n)$
- n Makeset and $O(m)$ cF/rU — total $O((n+m)\alpha(n)) = O(m\alpha(n))$
- total $\Theta(m \lg n) + O(m\alpha(n)) = \Theta(m \lg m)$
  since $\alpha(n) \in O(\lg n)$