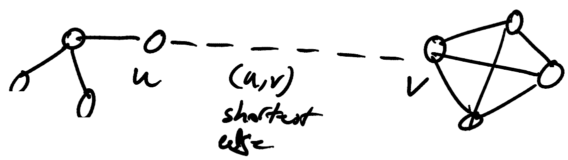
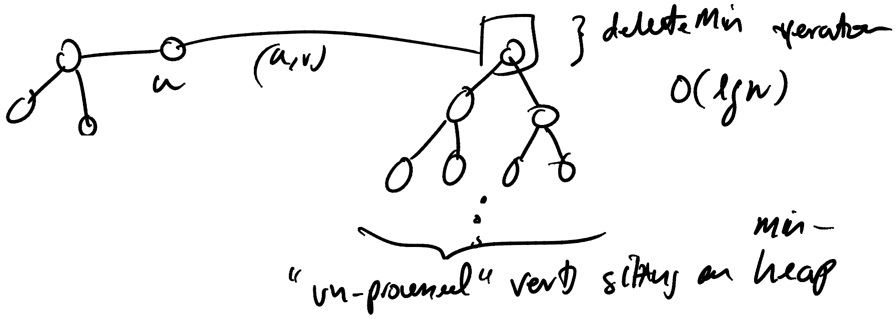
Thursday, October 19, 2006
3:24 PM

Prim's Alg.

- at any point in the alg., we have a set of vertices included in the (MST) tree; and a complementary set of verts not yet included in the tree (each vert has a boolean 'processed' – it's either been processed or not)

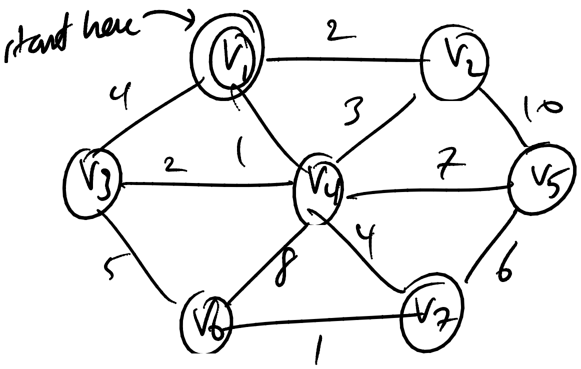
- at each stage alg finds a new vertex to add to the tree by choosing the edge (u,v) s.t. the cost of (u,v) is the smallest among all edges where u is in the tree and v is not





Thursday, October 19, 2006
3:24 PM

example in book (p 373-376)



(source)
(V1: will be our root node)

v	(processed) known	d_v	(parent) p_v
V1	F	0	0
V2	F	∞	0
V3	F	∞	0
V4	F	∞	0
V5	F	∞	0
V6	F	∞	0
V7	F	∞	0

- add V_1 to minheap
  (priority queue: V_1)

initialization ↑

- deleteMin
- set vertex's processed flag to T
- for each of vertex's adjacent verts
  - calc distance
  - insert to minheap

REPEAT

1 2 4 } costs (distances)

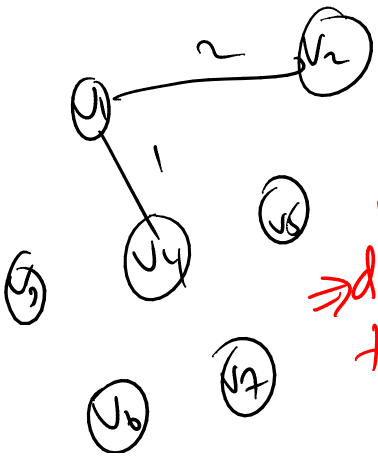
(queue: V_4, V_2, V_3)

V	known	dv	Pv
V1	T	0	0
V2	F	2	V1
V3	F	4	V1
V4	F	1	V1
V5	F	∞	0
V6	F	∞	0
V7	F	∞	0

- deleteMin: V_4
- for each of V_4's adj. verts: (~~V1~~, V2, V5, V7, V6, V3)

if(!processed) {
- calc distance (d_v) cost: 1+3, 1+7, 1+4, 1+8, 1+2
- if (onqueue)

adjust distance value; notify heap
(binheap.deckey)
else
insert to minheap



this updates binheap order ⇒ deckey has to reorg. heap

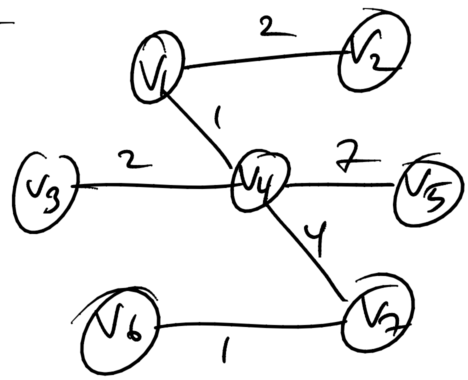
~~v1~~	~~T~~	~~0~~	~~0~~
v2	F	2	v1
v3	F	~~4~~ 3	~~v1~~ v4
~~v4~~	~~T~~	~~1~~	~~v1~~
v5	F	8	v4
v6	F	9	v4
v7	F	5	v4

(I'm adding cost of v4 — book does not)

Thursday, October 19, 2006
3:24 PM

eventually ...

v	k	dv	pv
v1	T	0	0
v2	T	2	v1
v3	T	3	v4
v4	T	1	v1
v5	F	8	v4
v6	F	6	v7
v7	T	5	v4



Thursday, October 19, 2006
3:24 PM

- Prim's alg pseudo code (assumes complete graph)
  input: source vertex (index to), s

1. go thru all verts, reset:
   v_i. dist = ∞
   v_i. processed = F
   v_i. parent = NULL (or ∅)

2. s. dist = ∅ // source vertex distance set to ∅
   s. onqueue = T } // put s on queue
   s. heappos = bh. insert(s) } (binheap)
   ↳ s, the graph vertex, stores its heap position

Thursday, October 19, 2006
3:24 PM

```
3. while ( ! bh.empty()) {
     3.a   // pick vertex v with shortest path
            v = bh.top();      // findMin
            bh.pop();          // deleteMin
     3.b   // book keeping
            v.processed = true;
            v.onqueue = false;
     3.c   for each of v's adjacent verts , w:
              3.c.i   if ( ! w.processed) {
                          c_{v,w} = distance from v to w
                          if (c_{v,w} < w.dist) {
                              w.parent = v
                              w.dist = c_{v,w}
                              if (w.onqueue)
                                  bh.deckey(w.heappos)
                              else
                                  w.heappos = bh.insert(w)
```

oops! just upset heap order → (w.dist = $c_{v,w}$)

if w is on heap fix it → (bh.deckey(w.heappos))