ICS 311 EXAM 2 NOTES

Topic 9 Heap Sort

Complete Binary Tree

- $2^{h+1} - 1$ nodes
- Height : $h = \lfloor \lg n \rfloor$
- # of leaves : $\lceil \frac{n}{2} \rceil$
- nodes of height h : $\lceil \frac{n}{2^{h+1}} \rceil$

Heap Property

Max
- key (parent(i)) $\geq$ key (i)

Min
- key (parent (i)) $\leq$ key(i)

Array Rep.

- root : A [1]
- parent of A[i] : A $[\lfloor \frac{i}{2} \rfloor]$
- left child : A[2i]
- right child : A[2i+1]

Topic 10 : Quick sort

Quick Sort
- partition : hw problem
- partition : $\Theta(n)$
- Worst Case : $\Theta(n^2)$
- Best Case : $\Theta(n \lg n)$

Counting Sort

- determines how many r less or greater than input.
- stable sort
- required $\Theta(n+k) \rightarrow \Theta(n)$ since $k \in \mathbb{R}$

Radix Sort.

- sorting from least to most sig. digit.

If stable $\Theta(n+k)$
else $\Theta(d(n+k))$

Topic 11: Balanced Trees

2 - 3 - 4 Trees

- node prop.

- Internal node has 2,- 4 children
- depth zero.
- tree is balanced

Thm. 2-4 tree sorting has height $\Theta(\lg n)$

2-4 insertion
$\Theta(\lg n)$

Red & Black Tree

Prop

- color: either red or black
- root: always black
- external: leaf is black
- internal:

if(node is red)
both children black

o depth

- Red Black to 2-4
  red nodes: capture internal struc.
  black nodes: capture (2, 3) itself

Topic R Dynamic Programming

- applies when subprob. are overlap unlike D & C method
- solves each subproblem as once & saves on a table

ex rod cutting

4 steps

1) characterize the optimal sol'n structure
2) recursively define value of an optimal sol'n
3) compute the value of optimal sol'n.

(4) construct an optimal sol'n from the computed info.

Topic 13: Greedy Alg

- finds sol'n top -> down
- assume optimized locally
- cannot do interdependent subproblem

Depth Search (DFS)

v.d = discovery time
v.f : finish time

White : undiscovered
gray : discovered but still discovering
black : finished

- DFS use FIFO stack
- Will search ∀ vertex until ∀ edges are discovered

Time Complexity

- $\Theta(V)$ line 1 to 5
- $\Theta(E)$ line 4
- total $\Theta(V + E)$

Type of Edges

- tree edge : (u, v)
- back edge : v descendant of u

- forward edge: v descendant of u but ¬ tree edge

- cross edge

DFS Prop.

- $[u.d, u.f] \cap [v.d, v.f]$ are entirely disjoint & neither u & v are descendants

- $[u.d, u.f] \subseteq [v.d, v.f]$ u is a descendants of v

- $[v.d, v.f] \subseteq [u.d, u.f]$ v is a descendant of u

- white indicates tree edge
- gray -> back edge
- black -> forward or cross edge.

Topological Sort

- if $(u, v) \in E$ then $v.f < u.f$

Time Analysis: $\Theta(M + (|E|)$

- u is gray
- v is not gray of contradicts and becomes a back edge

- if v is white $\rightarrow u.d < v.d < v.f < u.f$
- if v is black $\rightarrow v.f < u.f$

Strongly Connected Comp.

Let $G = (V, E)$ then SCC of G is the maximal set of vertex $C \subseteq V$ s.t. $\forall u, v \in C$ there is a path both u to v & v to u.

Run time $\Theta(V + E)$

Quiz Topic 14B

Topic 15 & Topic 16

CRLS 17.1-17.2 : 21.1 & 21.3

Lemma 21.11, 21.12 21.13 & Thm. 21.14

(skip potential method & Linked List representation)

I) Amortized Analysis

(a) Multipop

- push (S, x)
  - O(1) each call & O(N) for N seq.
- pop (S)
  - O(1) each call & O(N) for N seq.
  - running time of multipop
    - cost (min (s, k)) s: # of item
    - actual running time : linear function (n)

Q) Worse case of seq. of n ?

- push, pop, multipop : $O(n^2)$

II Accounting Method

- amortized cost

amount charged in each operation

difference between aggregate

- in aggregate, all opr. have same cost, & accounting methods differs.

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

[III] Dynamic Tables

[Def] load factor

- $\alpha = \frac{num}{size}$
  - num : # stored items
  - size : allocated size of tables

case : num = size = 0

↳ $\alpha = 1$

Topic 16

(I) Dynamic Disjoint Sets

group n distinct elements into disjoint sets that may change over time

Operations

- Make-set(x)
- Union(x, y)
  if $x \in S_x \wedge y \in S_y \rightarrow$
  combine 2 set
  - destroys S_x & S_y since they must be disjoint

- Find-set(x)
  - return a rep. of set containing x

Analysis

- n : # of Make-set opr.
- m : total # of opr.
- $m \geq n$
- can have most $n-1$ opr.

II) App. of Disjoint Sets

a) Min. Spanning Trees

i) Finding Connected Components

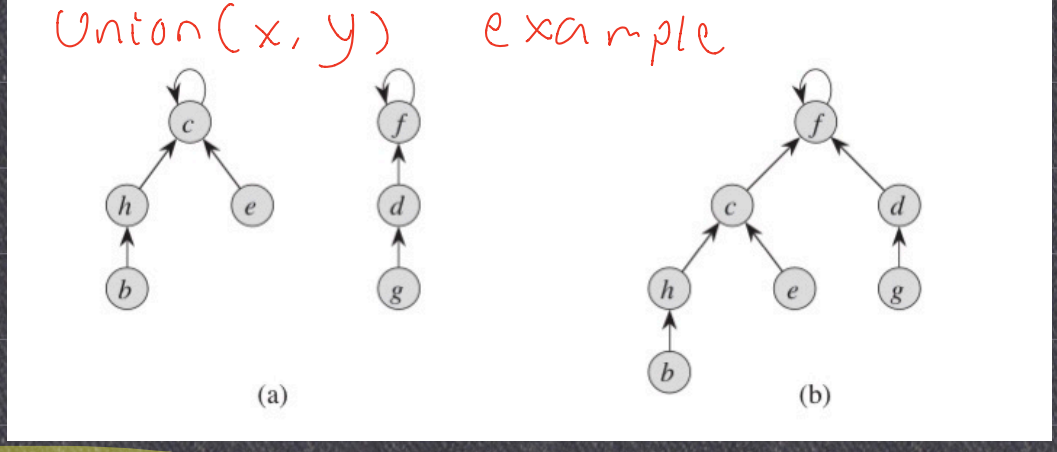
III) Forest Rep. of Disjoint Sets

a) representation

- each tree reps a set
- root of tree is set. rep.
- node points to parent
- root points itself as parent

Ex)

Union(x, y) example



b) Operations

- Make-Set(x) ; creates single node tree with root x
- Find-Set(x) ; follow parent pointers back to the root
- Union(x, y) ; make root one child of others

(c) Union by rank

- make the root of smaller tree a child of larger one

(d) time complexity

$$A_k(j) \begin{cases} j+1 & k=0 \\ A_{k-1}^{j+1}(j) & k \geq 1 \end{cases}$$

on quiz or exam
$O(\alpha(n))$

Lemma 21.11

- amortized cost of Make-Set operation is $O(1)$

Lemma 21.12

- amortized cost of each Link Operation is $O(\alpha(n))$

Lemma 21.13

- amortized cost of each Find-Set opr. is $O(\alpha(n))$

Thm. 21.14

- Seq. of m Make-Set, Union, & Find-Set can be performed on a disjoint sets forest with Union by rank & path in worst case is $O(m\,\alpha(n))$.

EX. 21.1-1

$G = (V, E)$

$V = \{a, b, c, d, e, f, g, h, i, j, k\}$

Edges: (d,i) (f,k) (g,i) (b,g) (a,h) (i,j) (d,k) (b,j) (d,f) (g,j) (a,e)

List the vertices:

Edge							
initial	{a} {h}	{b} {i}	{c}	{d}	{e}	{f}	{g}

$$(C-1)\varepsilon m \geq (1+\varepsilon)m$$

$$(C-1)\varepsilon \geq (1+\varepsilon)$$

$$\varepsilon C - \varepsilon \geq (1+\varepsilon)$$

$$\varepsilon C \geq 1 + 2\varepsilon$$

$$C \geq \frac{1+2\varepsilon}{\varepsilon} = \varepsilon^{-1} + 2$$

$$\left(\frac{1}{2}\right)^{-1} = 2$$

$$\left(\frac{1}{10}\right)^{-1} = 10$$

$$\frac{1}{4} + \frac{8}{4} = \frac{9}{4}$$

Min. Spanning Trees
CLRS: Ch. 23

I) Min. Spanning Trees

(a) Spanning Trees

Def T for a connected graph G is a tree includes all vertices of G

(b) Min. Span Tree $G = (V, E)$

- V: objects or location
- E: available conn.
- $w(u, v)$: weight of each edge
- look 4 $T \subseteq E$ s.t.
  - T connects $\forall$ vertices V of G
  - $w(T) = \sum_{(u,v) \in T} w(u, v)$ is minized
- G must be connected

Q Are min. span. tree unique?

II Generic Alg. & Safe Edge Thm

(a) prop of Min. span. trees

- G has $|V| - 1$
- any tree has no cycles
- one path between vertices
- there might be more than one MST

(b) Building a Sol'n

- initially A has no edges
- maintain loop invariant:
  $A \subseteq \exists MST$

[def] safe

edge (u, v) is safe for A iff $A \cup \{(u, v)\} \subseteq MST$

(C) Finding Safe

- cut $(S, V-S)$

partition of vertex's into disjoint sets.

- $A \subseteq MST$

- cut respects A iff no edge in A crosses cut

- light edge: iff weight is min. over all edges crossing the cut

Safe Edge Thm

Let $G = (V, E)$,
$A \subset G$ s.t. G is MST
$(S, V-S)$ be a cut that respects A & (u, v) be a light edge crossing $(S, V-S)$
Then (u, v) is safe for A.

(III) Kruskal's Alg.

- diff. between Connected-Component edges are processed in greedy
- Analysis: Result $O(E \lg V)$

```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3      MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6      if FIND-SET(u) ≠ FIND-SET(v)
7          A = A ∪ {(u, v)}
8          UNION(u, v)
9  return A
```

(IV) Prim's Algo.

- analysis:

$O(V \lg V) + O(E \lg V)$

if G is connected

$O(E \lg V)$

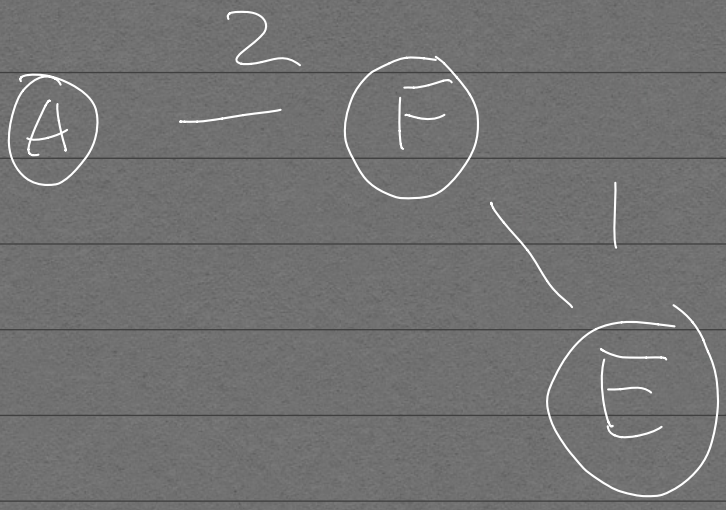
```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

Corollary 23.2

Let $G = (V, E)$ be a connected, undirected graph with a real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, and let $C = (V_C, E_C)$ be a connected component (tree) in the forest $G_A = (V, A)$. If (u, v) is a light edge connecting C to some other component in G_A, then (u, v) is safe for A.

Mini. edge : smallest value

$|E| \cdot (|V| - 1)$


2
A
F
1
E