CS 61B SP19 MT II

$\sum n + n/2 + n/4 \ldots 1 = \boxed{\Theta(n)}$ $\sum 1+2+4+8+16 = \boxed{\Theta(N)}$ BUT ALSO IS 2^N $\sum 1+2+3+4+5 \ldots +N \, \boxed{\Theta(N}$

DATA STRUCTURE	PURPOSE	OPERATIONS + RUNTIMES
LinkedList	ordered Data	get - O(N) add - Ø(1) remove - Ø(N) space - O(N)
Array List	ordered Data	get - O(1) add - Ø(N) remove - Ø(N) space O(N)
Binary Search Tree	comparable, use over hash if it is hard to compute	search - AVG O(log N) Bad O(N) insert - AVG O(log N) Bad O(N) delete - AVG O(log N) Bad O(N)
Balanced Trees	Maintain Runtimes	All operations are Ø(log N)
Hashing Structures	When data is not Ordered, O(1) Needed, and good hashcode	get - AVG O(1) Worst O(N) insert - AVG O(1) Worst O(N) remove - AVG O(1) Worst O(N)
Stacks / Queues	FILO - look at the most recent first for stacks, FIFO is for Queues	All operations run on average constant time Ø(1)
Heap / PriorityQueue	comparable data, need most/least	peek() - Ø(1) search - Ø(1) insert AVG Ø(1) BAD O log N deleteSmallest() O(log N)
Tries / TSTs	Prefixed operations on query M	search - Ø(M) insert - Ø(M) NOT DEPENDENT ON VALUES

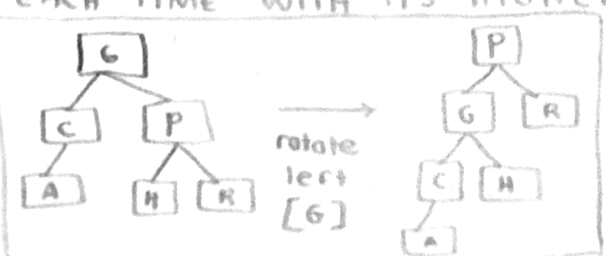
SETS ARE NOT ORDERED AND DO NOT ALLOW DUPLICATES.

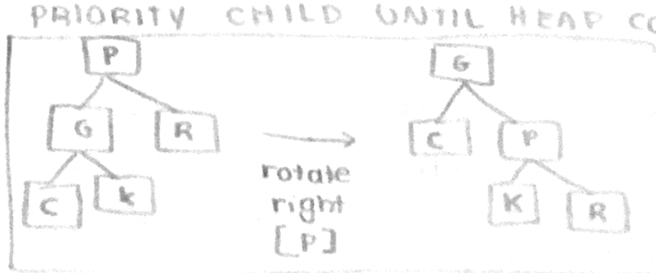
MAPS STORE IMMUTABLE KEYS, BUT VALUES ARE MUTABLE (LIKE ON DUPLICATES)

LLRB INSERTION

1) RED LINK @ CORRECT LOCATION
2) IF IT'S RIGHT, ROTATE IT LEFT
3) IF TWO RED LINKS IN A ROW, ROTATE RIGHT ON THE TOP NODE
3) IF A NODE HAS TWO RED LINKS, FLIP ALL NODES POINTING TO THAT NODE

2-3 INSERTION ADD NODE TO LEAF, POP UP MIDDLE LEFT IF OVERSTUFFED, EACH N LAYER HAS TO HAVE N+1 CHILDREN. HEAP INSERTION INSERT AT THE VERY END AND FLOAT TO RIGHT LOCATION BY SWAPPING WITH ITS PARENT NODE HEAP DELETION SWAP ROOT W/ LAST ELEMENT, SWIM ROOT DOWN BY SWAPPING EACH TIME WITH ITS HIGHER PRIORITY CHILD UNTIL HEAP CONDITION MET.





RL → MAKE NODE LEFT CHILD OF ITS OLD RIGHT CHILD

RR → MAKE NODE RIGHT CHILD OF ITS OLD LEFT

* VALID LLRBS HAVE A 1-1 CORRESPONDENCE [ISOMETRY] WITH A 2-3 TREE

RUNTIME	DEFINITION
O(n)	f(n) GROWS NO FASTER THAN O
Ø(n)	BOUNDED TOP AND BOTTOM BY SAME FUNC
Ω(n)	f(n) GROWS NO SLOWER THAN Ω

TRAVERSAL	STEPS
BREADTH [BFS]	- DIJKSTRA'S WITH ALL EDGE WEIGHTS EQUAL TO 1, GOES IN DIST ORDER
PREORDER [DFS]	MARK, VISIT ROOT, THEN REPEAT PROCESS FOR ALL CHILDREN IN TOP-DOWN ORDER
POSTORDER [DFS]	KEEP MARKING NODES UNTIL THERE ARE NO UNMARKED KIDS → RETURN

TREES

LO 4 3 5
PRE 4 3 5
POST 3 5 4
IO 3 4 5

IITM P192 #18 23

ALGORITHM	PURPOSE	STEPS	RUNTIME
DIJKSTRA'S	SPT [ALL NODES]	- ADD CHILDREN, DISTANCE, SOURCE - POP SMALLEST DISTANCE + VISIT - IF A BETTER WAY TO REACH A NODE IS SEEN, CHANGEPRIORITY()	$2V\log V + E\log V$ ↓ SIMPLIFY $O(E\log V)$
A* HEURISTIC	SINGLE TARGET PATH TREE	- SAME AS ABOVE BUT ADD $h(v)$ - AN ADMISSIBLE (NEVER OVERESTIM) AND CONSISTENT (NEVER GREATER THAN SUCCESSOR + h(SUCCESSOR)) IS ALWAYS RIGHT	DEPENDS ON HEURISTIC BEST CASE IS $\emptyset(n^2)$ HIT EACH NODE 2X
PRIM'S	MST	- CONSIDER V IN ORDER OF DIST FROM CURRENT MST - ADDS THE LIGHT, CONNECTED, AND ACYCLIC EDGE	$O(E\log V)$
KRUSKAL'S	MST	- ADD LIGHTEST, ACYCLIC EDGE REGARDLESS OF IF CONNECTED - PRODUCES MST OF SAME WEIGHT AS PRIM'S ALGO	$O(E\log V)$ $O(E\log E)$ IF WE USE A PQ

$\boxed{\text{HIBBARD DELETION}}$ REPLACE NODE WITH ITS SUCCESSOR WHICH IS LARGEST LEFT VALUE / SMALLEST RIGHT VALUE

$\boxed{\text{NODES, LEVELS, WORK}}$ IF WORK PER NODE IS A CONSTANT, TOTAL WORK = NO_{NODES} * WORK / NODE. # OF NODES IS GENERALLY 2^{LAYERS}. OTHER CASE IS WHEN WORK PER LAYER IS CONST, IN THAT CASE, WORK = LEVELS * WORK / LEVEL.

$\boxed{\text{RUNTIME ORDER}}$ CONSTANT < LOG < LIN < POLY < EXP

$\boxed{\text{LLRB INVARIANTS}}$ EVERY ROOT → LEAF PATH HAS SAME NUMBER OF BLACK LINKS, RED LEANS LEFT.

$\boxed{\text{GRAPH SEARCH}}$ RUNS IN $O(|V| + |E|)$ TIMING

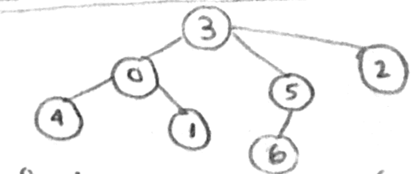
$\boxed{\text{MATH BASICS}}$

$2^{\log N} = N = \emptyset(N)$

TREE	INSERT	DELETE	FIND	HT
BST	$O(n)$	$O(n)$	$O(n)$	$\Omega(\log N)$ $O(N)$
B-TREE	$\emptyset(\log N)$	$\emptyset(\log N)$	$O(\log N)$	$\emptyset(\log N)$
LLRB	$\emptyset(\log N)$	$\emptyset(\log N)$	$O(\log N)$	$\emptyset(\log N)$

$\boxed{\text{CUT PROPERTY PROOF}}$ A CUT IS A PARTITION OF THE VERTICES OF A GRAPH INTO TWO DISJOINT SUBSETS. CROSSING EDGES CONNECT TWO EDGE SETS TOGETHER. FOR ANY CUT C IN GRAPH G, IF THE LIGHTEST EDGE ACROSS C IS UNIQUE, THEN C MUST BE IN EVERY MST OF G. ANY CUT C IN THE GRAPH SPLITS IT TO TWO SUBGRAPHS, CONNECTED BY THE EDGES ALONG THE CUT. THE MST OF AN OVERALL GRAPH CAN ONLY INCLUDE

CS 61B SP19 MT II



0	1	2	3	4	5	6
3	0	3	3	0	3	5

EXAMDLE [id] ARRAV

IMPOSSIBLE WQU CYCLES OR NON-LOG HEIGHT, THE SMALLER TREE GOES UNDER FATTER ONE.

GENERICS public class VendingMachine <T>
public <T> Integer sell (T item) [SOME WEIRD CASE]

HEURISTIC VOCAB AN ADMISSIBLE HEURISTIC WILL NEVER OVERESTIMATE THE TRUE COST TO VISIT A GOAL NODE. A CONSISTENT HEURISTIC IS WHEN THE HEURISTIC VALUE OF n IS NEVER GREATER THAN IT'S SUCCESSOR'S COST + SUCCESSOR'S h(v)

DIJKSTRA/A* CODE

```
dijkstra(G, s):
    while stack Full:
        Visit (minVertex)
Visit (v):
    mark(v)
    For each edge:
        relax(edge)

relax (e):
    v = e.source
    w = e.target
    currBest = dTo(w)
    possBest = dTo(v) + weight
    if better < best:
        visit this edge
```

[A* JUST ADDS + h(v)]
[PRIM'S JUST CONSIDERS DISTANCE TO OVERALL]

KRUSKAL'S CODE

```
Consider each edge:
    visit Smallest ()
    if no cycle
```

CODE	RUN

if(N=0) return
F(N/2)
if (condition) F(N/2)
	Best - Ø logN
	Worst Ø N

if(N=0) return
F(N-1)
if(condition) F(N-1)
	Best Ø N
	Worst Ø 2^N

For(i=0; i<N; i*=2) print	Ø(log N)
if N==0 return F(n/4) F(n/4) F(n/4) F(n/4) g(Quadratic) // runs in N^2	Ø(N^2log N)

TREE RUNTIMES

CONSTANT/NODE = W/NODE * # NODES
CONSTANT/LAYER = W/LAYER * HEIGHT
NUMBER OF NODES = (BRANCHING FACTOR)^(HEIGHT)

KDTREE EXAMPLE