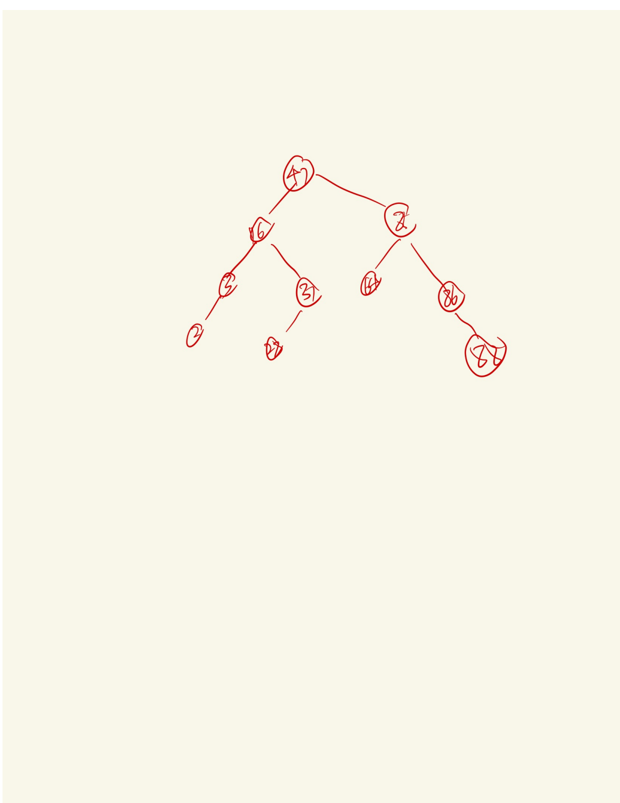
Problem Set 4

Name: Zunmi

Collaborators: None

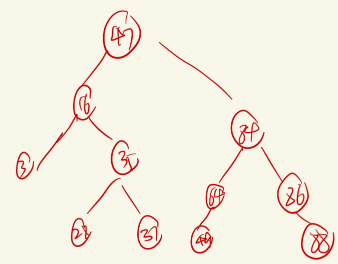
Problem 4-1.

(a) 16(skew=2),37(skew=-2)



(b)

(c)



Problem 4-2.

(a) Min-heap

(b) Mx-heap

(c) [0,2,2,8,9,13]

(d) Min-heap

Problem 4-3.

(a) Build a max-heap from A keyed on s_i,mapping to r_i, then use $delete\_max$ k times to extract the k largest elements.

(b)
```
greaters = []
def greater_than_x(A,i=0, x):
    assert len(A) > 0
    while i < len(A):
        if A[i] <= x:
            greater_than_x(A, (i+1)//2, x)
        else:
            greaters.append(A[i])
            greater_than_x(A, 2*i+1, x)
            greater_than_x(A, 2*i+2, x)
    return greaters
```

Problem 4-4. 1. Priority Queue P on the solar farms(a Max-Heap), storing for each solar farm its address s_i, capacity c_i, and its available capacity a_i (initially $a_i = c_i$), keyed on available capacity;

2. Set data structure B(Hash-Table) mapping each building's name b_j to the address of the solar-powered building's name bj to the address of the solar farm s_i that it is connected to and its demand d_j;

3. Set data structure F(Hash-Table) mapping the address of each solar farm s_i to: (1) its own Set data struc- ture B_i(Hash-Table) containing the buildings associated with that farm, and (2) a pointer to the location of s_i in P .

the operations:

initialize(S): build Set data structures P and then F from S, and initialize B as empty.

power_ on(b_j, d_j): First, find a solor farm to connect by deleting a solar farm si from P having largest available capacity ci (delete_ max) and checking whether it's capacity is at least d_j . There are two cases:

a. $d_j > c_i$, so reinsert the solar farm back into P (relinking a pointr from F to a location in P) and return that no solar farm can currently support the building.

b. $d_j <= c_i$, so subtract d_j from c_i and reinsert it back into P (relinking a pointer). Then, add b_j to B mapping to s_i, and then find the B_i in F associated with s_i and addb_j to B_i.

power off(b_j): Lookup the s_i and d_j associated with b_j in B, lookup B_i in F using s_i, and remove b_j from B_i. Lastly, go to s_i's location in P and remove s_i from P , increase c_i by d_j,and reinsert s_i into P.

customers(s_i): lookup B_i in F using s_i, and return all names stored in B_i.

Problem 4-5. Store the matrices in a Sequence AVL tree T,where every node is augmented with the following information:

1. the size of the subtree of node;

2. the ordered product of all matrices in the subtree rooted of node;

operations:

initialize(M): build the AVL tree T from M

update_joint(k,M):find the kth node v in T,replace the v.matrix with M, and update the ordered product of all matrices in the subtree rooted at v.

full_transformation(): return the ordered product of root in T.

Problem 4-6.

(a) find tne maximum

(b)

(c)

(d) Submit your implementation to `alg.mit.edu`.