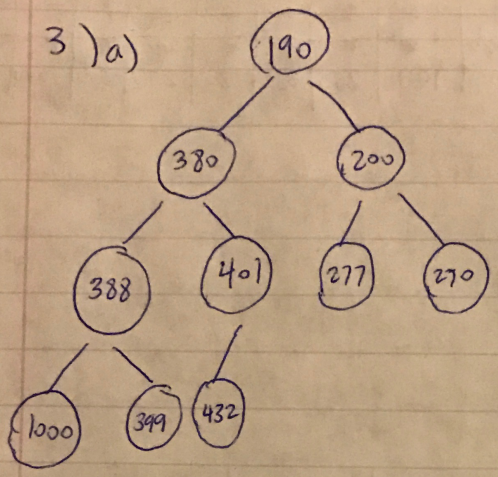
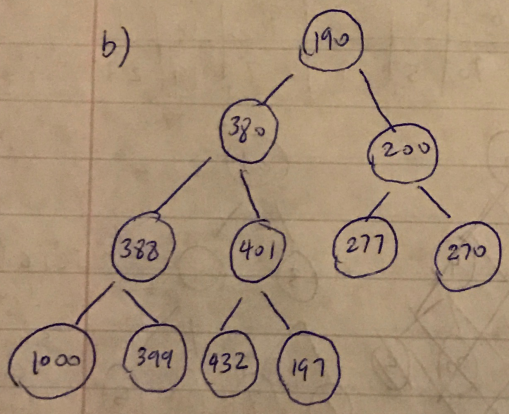
Danny Tan
Dt1462
HW#11

1) O(n+k)
2) Insert (x) : O(1)
   deleteMin(): O(log(n))
   findMin() O(1)

3) a)



b)

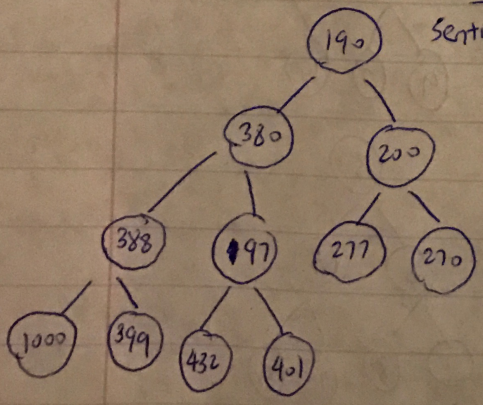


0	1	2	3	4	5	6	7	8
Sentinel	190	380	200	388	401	277	270	1000

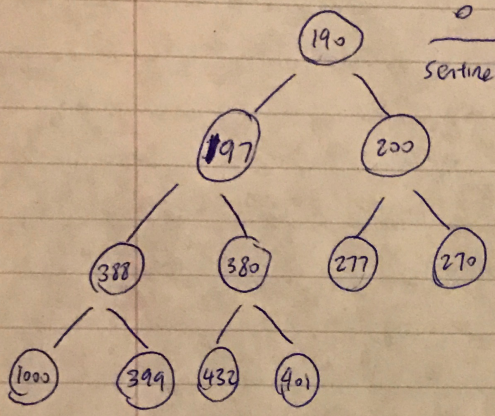
9	10	11
399	432	197

percolate up

0	1	2	3	4	5	6	7	8	9	10	11
Sentinel	190	380	200	388	197	277	270	1000	399	432	197



percolate up

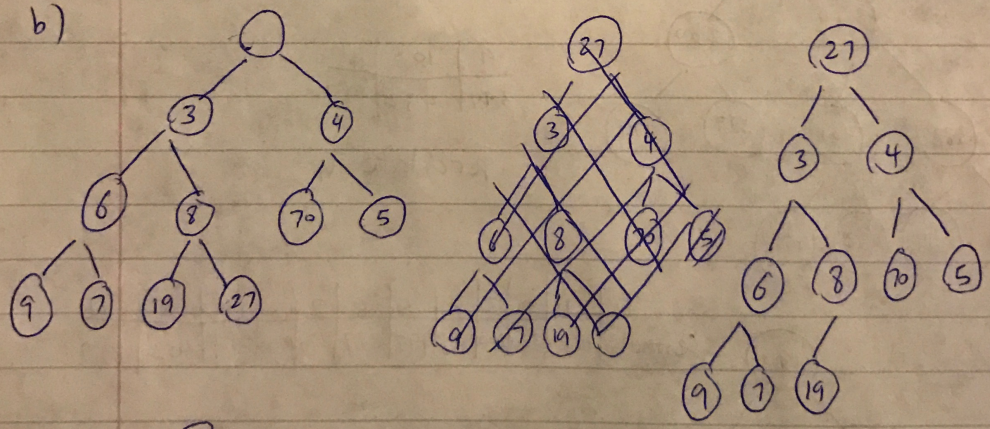


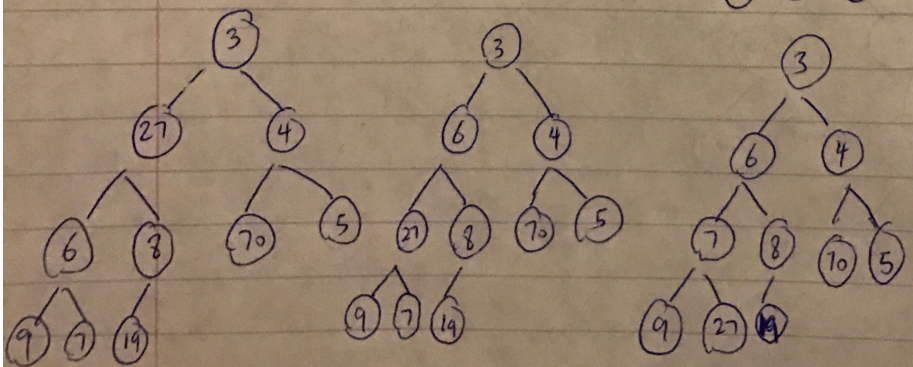
0	1	2	3	4	5	6	7	8	9	10	11
sentinel	190	197	200	388	380	277	270	1000	399	432	401

4) a)

0	1	2	3	4	5	6	7	8	9	10	11
sentinel	2	3	4	6	8	70	5	9	7	19	27

b)

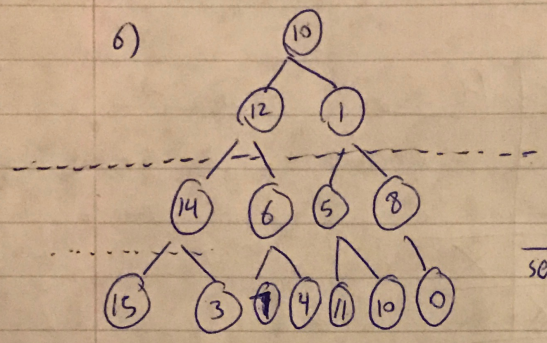




5)

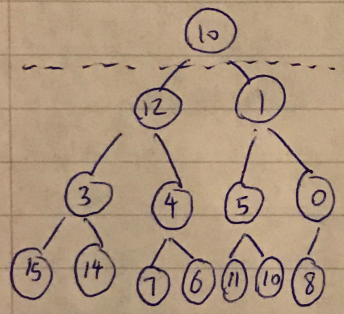
```
template <class Comparable>
void BinaryHeap::insert( const Comparable & x ) {
        if( theSize + 1 == array.size( ) )
                array.resize( array.size( ) * 2 + 1 ); // Percolate up int hole = ++theSize;

        int hole = theSize ++;
        for( ; x < array[ hole / 2 ]; hole /= 2 )
                array[ hole ] = std::swap ( array[ hole / 2 ] );
        array[ hole ] = x;
}
```

6)



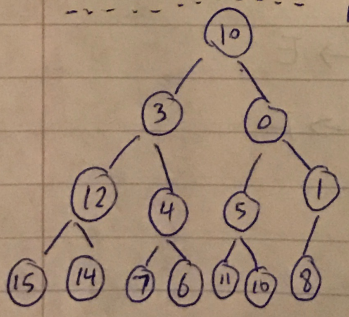
percolate down

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
sentinel	10	12	1	14	6	5	8	15	3	7	4	11	10	0

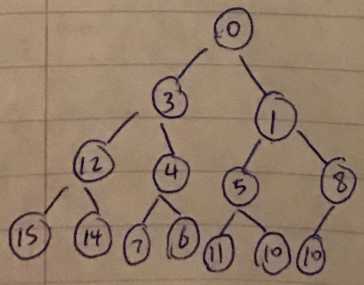


percolate down

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
sentinel	10	12	1	3	4	5	0	15	14	7	6	11	10	8



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
sentinel	10	3	0	12	4	5	1	15	14	7	6	11	10	8



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
sentinel	0	3	1	12	4	5	8	15	14	7	6	11	10	10

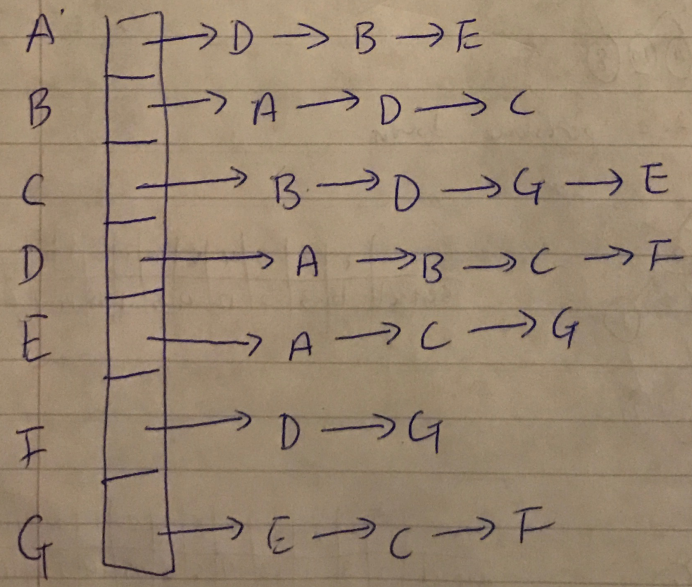
7)

phase	distance/predecessor							visiting	queue
	G	A	B	C	D	E	F		
init	0/-								G
1	0/-	1/G					1/G	G	F A
2	0/-	1/G				2/F	1/G	F	A E
3	0/-	1/G				2/F	1/G	A	E
4	0/-	1/G			3/E	2/F	1/G	E	D
5	0/-	1/G	4/D	4/D	3/E	2/F	1/G	D	B C
6	0/-	1/G	4/D	4/D	3/E	2/F	1/G	B	C
7	0/-	1/G	4/D	4/D	3/E	2/F	1/G	C	

Shortest path from G to C is 4 (GFEDC)

8) Adjacency matrix

	A	B	C	D	E	F	G
A		X		X	X		
B	X		X	X			
C		X		X	X		X
D	X	X	X			X	
E	X		X				X
F				X			X
G			X		X	X	



9)

phase	distance/predecessor							visiting	discovered
	A	B	C	D	E	F	G		
init	0/-1	Inf/ -1	Inf/-1	Inf/-1	Inf/-1	Inf/-1	Inf/-1		A
1	0/-1	6/A	Inf/-1	3/A	1/A	Inf/-1	Inf/-1	A	B D E
2	0/-1	6/A	19/E	3/A	1/A	Inf/-1	20/E	E	B D C G
3	0/-1	3/D	15/D	3/A	1/A	6/D	20/E	D	B C G F
4	0/-1	3/D	8/B	3/A	1/A	6/D	20/E	B	C G F
5	0/-1	3/D	8/B	3/A	1/A	6/D	20/E	F	C G
6	0/-1	3/D	8/B	3/A	1/A	6/D	13/C	C	G
7	0/-1	3/D	8/B	3/A	1/A	6/D	13/C	G	

Shortest path from A to G is 13 (ADBCG)

10)
a->b 3
a->c 2
b->c -3

In this example, since c has the smallest distance it is deleted from priority queue first by using the greedy algorithm. Therefore the final distance from a to c is 2. However, we can do better by going through b which will give a distance of 0.

11)

We can create an array which keeps track the number of different minimum paths from one point to another point. If the dijkstra finds a smaller path, then the count of that point resets to 0. If there is an equivalent minimum path, the count is increment by 1.