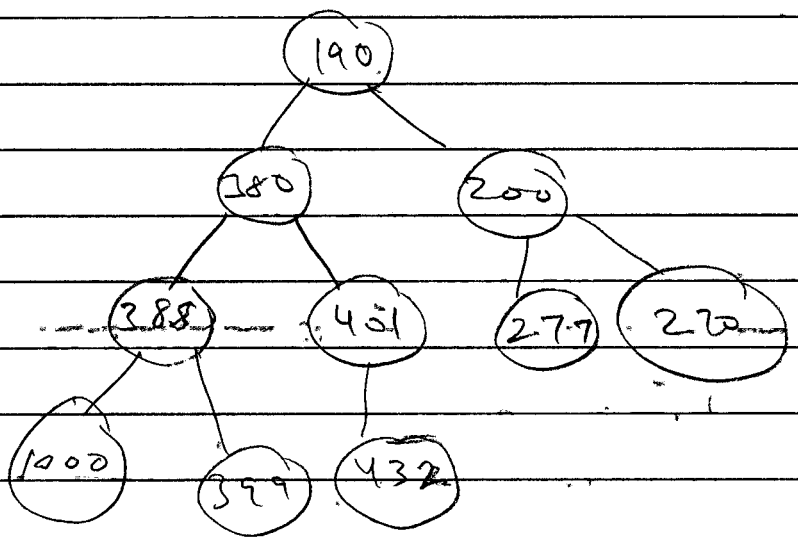
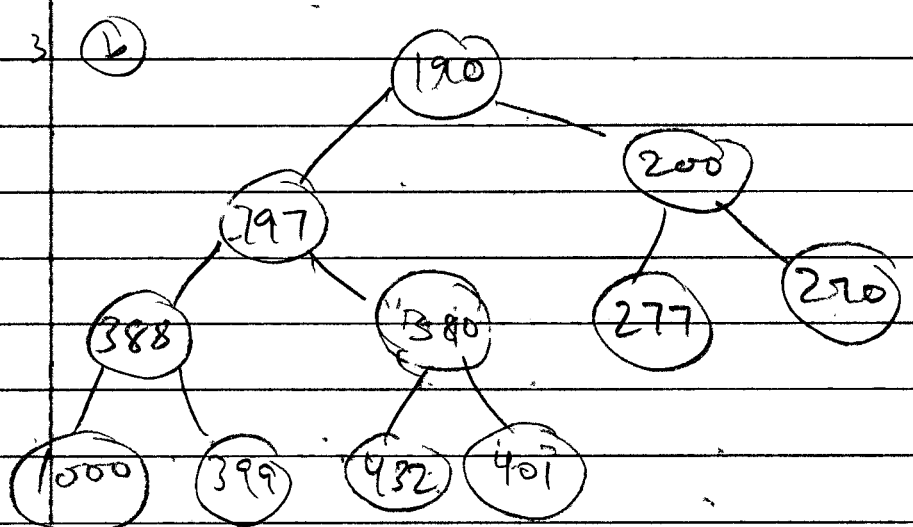
Mohammed Tofiel Chowdhury
mtc405@nyu.edu

1. O(n+k) log(ns)
2. insert(x) : O(1)
   ~~DeleteMin()~~ : O(n)
   findMin() : O(n)

3 (a)



3 (b)

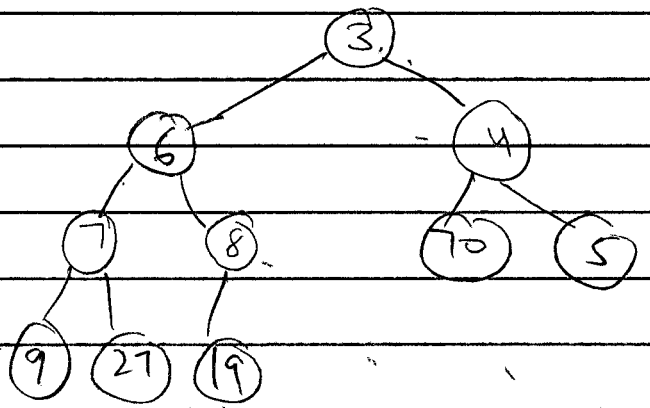


sentinel	190	197	200	388	380	277	270	1000	399	432	401

(4)

	sentinel	2	3	4	6	8	70	5	9	7	19	27
0	1	2	3	4	5	6	7	8	9	10	11	12

(3)



(5)

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

    int hole = theSize - 1;
    while( hole != 0 && x < array[ (hole - 1) / 2 ] )
    {
        array[ hole ] = move( array[ (hole - 1) / 2 ] );
        hole = (hole - 1) / 2;
    }
    array[ hole ] = x;
}
```

(6)

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

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

final →

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/-	-1/-1	-1/-1	-1/-1	-1/-1	-1/-1	-1/-1		G
1	0/-	1/G	-1/-1	-1/-1	-1/-1	-1/-1	1/G	G	FA
2	0/-	1/G	-1/-1	-1/-1	-1/-1	2/F	1/G	F	AE
3	0/-	1/G	-1/-1	-1/-1	-1/-1	2/F	1/G	A	E
4	0/-	1/G	-1/-1	-1/-1	3/E	2/F	1/G	E	D
5	0/-	1/G	4/D	4/D	3/E	2/F	1/G	D	BC
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	

G → F → E → D → C

8.

	A	B	C	D	E	F	G
A	X	6	X	3	1	X	X
B	6	X	5	0	X	X	X
C	X	5	X	12	18	X	5
D	3	0	12	X	X	3	X
E	1	X	18	X	X	X	19
F	X	X	X	3	X	X	14
G	X	X	5	X	19	14	

A → B(6) → D(3) → E(1)

B → A(6) ↔ D(0) → C(5)

C → D(12) → B(5) → E(18) → G(5)

D → A(3) → B(0) → C(12) → F(3)

E → C(18) → G(19) → A(1)

F → D(3) → G(14)

G → C(5) → E(19) → F(14)

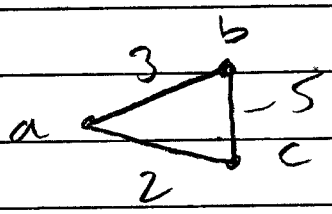
9.

Phase	Distance / predecessor							visiting	Discovered
	A	B	C	D	E	F	G		A
init	0	∞	∞	∞	∞	∞	∞	A	A
1	0	6/B	∞	3/A	1/A	∞	∞	A	B, D, E
2	0	6/B	18/E	3/A	1/A	∞	19/E	E	B, D, C, G
3	0	0/D	12/D	3/A	1/A	3/D	∞	D	B, C, G, F
4	0	0/D	5/B	3/A	1/A	3/D	∞	B	C, G, F
5	0	0/D	5/B	3/A	1/A	3/D	14/F	F	G, G
6	0	0/D	5/B	3/A	1/A	3/D	5/C	C	
7	0	0/D	5/B	3/A	1/A	3/D	5/C	G	

From A to G: A → D → B → C → G

10. use an array to show the different paths from v to w. When a new shortest path is found, update the array.

11.



a	b	c
0/-	3/a	2/a

Dijkstra's algorithm will give us the shortest path from a to c in a → b → c not a → c