#1 insertion sort:

	99	6	1	45	0	3	27	9	8
1		ノ							
	6	99	1	45	0	3	27	9	8
_									
	1	6	99	<mark>45</mark>	0	3	27	9	8
	1	6	45	99	o O	3	27	9	8
7									
	0	1	6	45	99	3	27	9	8
						-			
	0	1	3	6	45	99	27	9	8
				•			ノ		
	0	1	3	6	27	45	99	9	8
						_			
	0	1	3	6	9	27	45	99	8
					<u></u>				
	0	1	3	6	8	9	27	45	99

#2 bubble sort

Pass 1:

_	T_				T		1	1	1	T	
A	L	Р	Н	Α	В	E	Т	I	С	А	L
А	L	P	Н	А	В	E	Т	I	С	А	L
А	L	P	H	А	В	Е	Т	1	С	А	L
		<u> </u>	4								
А	L	Н	P	A	В	Е	Т	ı	С	А	L
		1	2	7		_	1.				
А	L	Н	А	P	В	E	Т	1	С	А	L
	-	111	^	•	<u>*</u>		'	'		^	
	Τ.	1	1.			l =	I_	1.		Ι.	1.
А	L	Н	А	В	P	E	Т	I	С	А	L
					_	_					
А	L	Н	А	В	E	P	ī	I	С	А	L
Α	L	Н	А	В	E	Р	T	ı	С	А	L
							_				
Α	L	Н	А	В	E	Р	I	ī	<u>c</u>	А	L
А	L	Н	А	В	E	Р	ı	С	ī	A	L
L.,	1-	1	1.,	1-	1-	1.	1.	1 -	_	<u> </u>	
^		1	^	р	г	ь			_		
А	L	Н	А	В	E	Р	I	С	А		Ļ
		T	1	1	1	I	I	I	T	_	
А	L	Н	А	В	E	Р	I	С	А	L	T

Pass 2:

A	L	Н	А	В	E	Р	I	С	А	L	T		
А	L	H	А	В	E	Р	1	С	А	L	T		
А	Н	L	A	В	E	Р	I	С	А	L	T		
А	Н	А	L	В	E	Р	I	С	А	L	T		
А	Н	А	В	L	E	Р	I	С	А	L	T		
А	Н	А	В	E	L	P	I	С	А	L	T		
А	Н	А	В	E	L	P	ı	С	А	L	T		
А	Н	А	В	E	L	I	P	C	А	L	T		
							_						
А	Н	А	В	Е	L	I	С	P	A	L	T		
А	Н	А	В	E	L	I	С	А	P	L	T		
									_				
А	Н	А	В	E	L	I	С	А	L	P	T		

Pass 3:

A	H	А	В	E	L	ı	С	А	L	P	T
_		l	l				l				
А	H	A	В	E	L	I	С	А	L	P	T
А	А	H	В	E	L	I	С	А	L	P	T
А	А	В	H	E	L	I	С	А	L	P	T
				_							
А	А	В	E	H	L	I	С	А	L	P	T
А	А	В	E	Н	L	I	С	А	L	P	T
						_					
А	А	В	E	Н	I	L	C	А	L	P	T
А	А	В	E	Н	I	С	L	A	L	P	T
								-			
А	А	В	E	Н	I	С	А	L	L	P	T
А	А	В	E	Н	I	С	А	Ц	L	P	T

Pass 4:

A	A	В	E	Н	I	С	А	L	L	P	T
	l 	l 	I_	l	I.	I _	l .	l.	<u>-</u>	l <u> </u>	_
А	A	В	E	Н	I	С	А	L	L	P	T
А	А	В	E	Н	I	С	А	L	L	P	T
А	А	В	E	B	I	С	А	L	L	P	T
А	А	В	E	B	i	С	Α	L	L	P	T
А	А	В	E	Н	i	C	А	L	L	P	T
						_					
А	А	В	E	Н	С	I	A	L	L	P	T
А	А	В	E	Н	С	А	I	L	L	P	T
А	А	В	E	Н	I	С	А	L	<u>L</u>	P	T

	Pass	s 5:									
A	A	В	Е	Н	I	С	А	L	L	P	T
А	A	B	E	Н	1	С	А	L	L	P	T
А	А	В	Ē	Н	1	С	А	L	L	P	T
А	А	В	Ē	H	I	С	А	L	L	P	T
А	А	В	E	H	i	С	А	L	L	P	T
А	А	В	E	Н		C -	А	L	L	P	Ţ
А	А	В	E	Н	С		A	L	L	P	T
А	А	В	E	Н	С	А	I	L	L	P	T
	Pass	S 6:	E	Н	С	А	i	L	L	P	T
A	A	В	E	Н		А		ļ.	<mark>L</mark>	<u> </u>	<u> </u>
А	A	В	E	Н	С	А	I	L	L	P	T
А	А	B	E	Н	С	А	-	L	L	P	T
A	А	В	Ē	H	С	А	I	L	L	P	Ţ
А	А	В	E	H	C	А	I	L	L	P	T
А	А	В	E	С	H	<u> </u>	I	L	L	P	T
A	А	В	E	С	А	H	ı	L	L	P	T
	Pass	s 7:	·				·	·			·
A	A	В	Е	С	А	H	I	L	L	P	Ţ
A	A	В	E	С	А	Н	I	L	L	P	T
A	А	B	Ē	С	А	H	I	L	L	P	Ţ
A	А	В	E	C	А	H	ı	L	L	P	T

E

Α

С

С

А

А

А

А

В

В

A

E

Н

Н

ļ

L

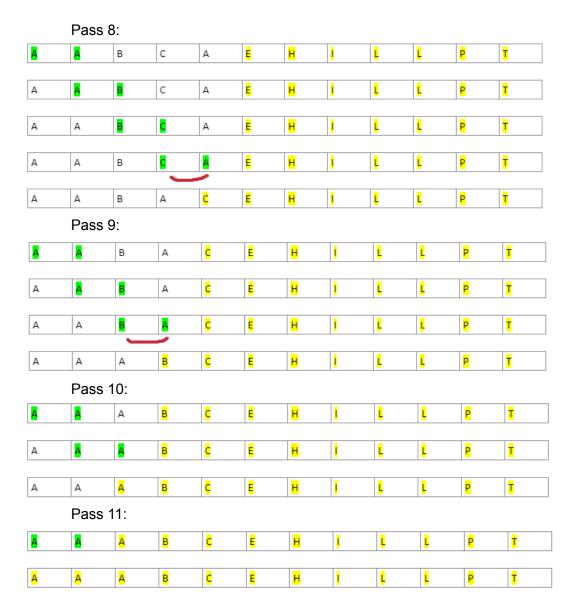
L

L

P

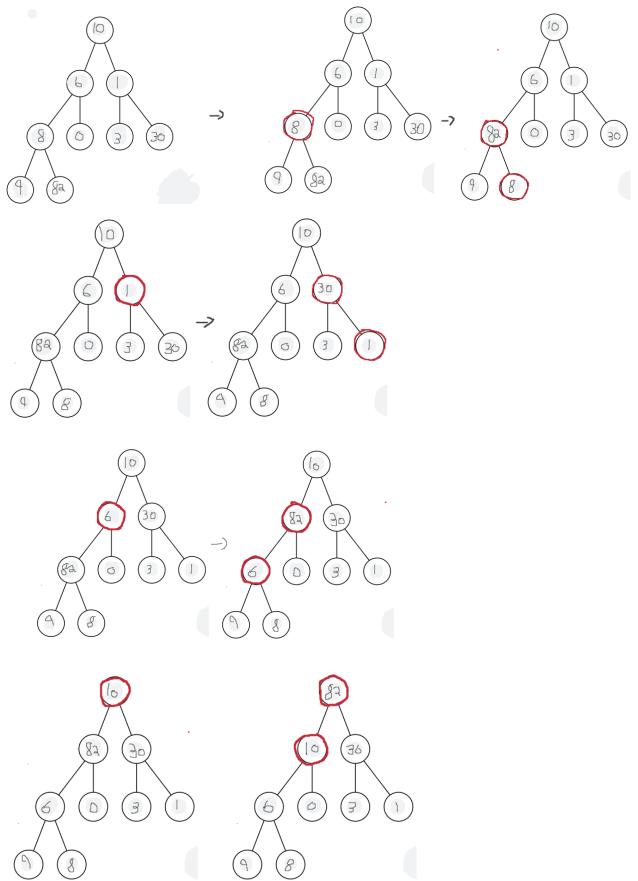
T

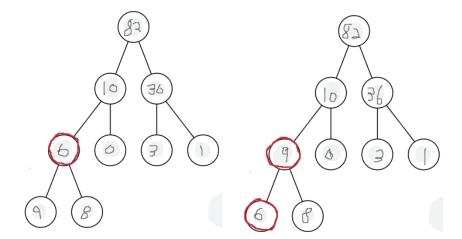
T



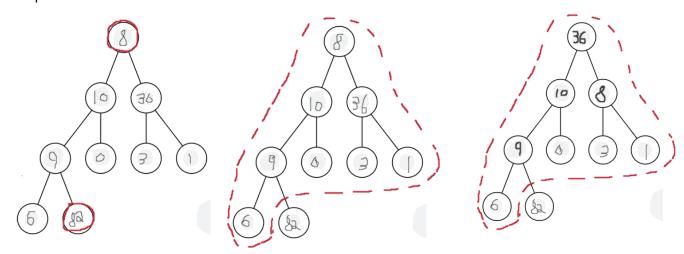
#3 O(n^2)

#4
Building initial heap / running heapify():

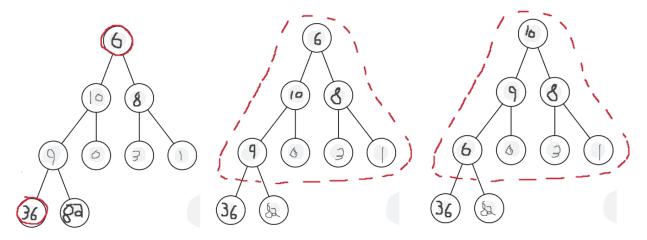




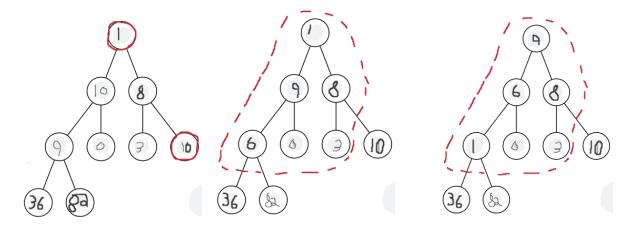
Heapsort:



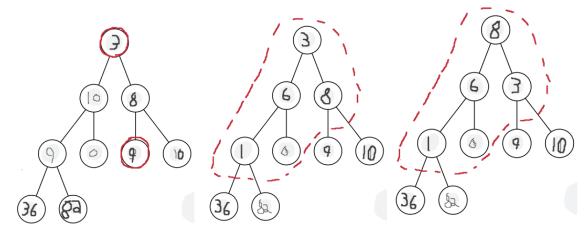
Swap first and last value, last value now sorted, then run heapify() on unsorted values



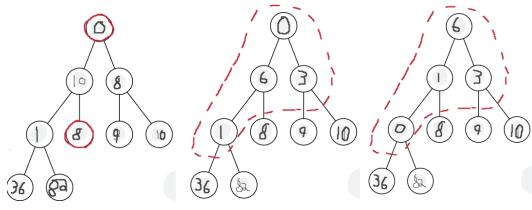
Again:



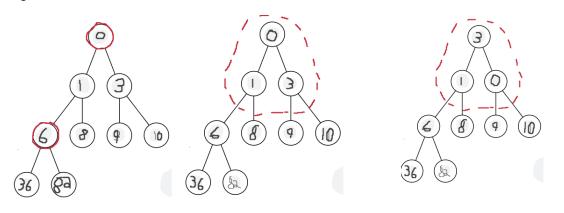
Again:



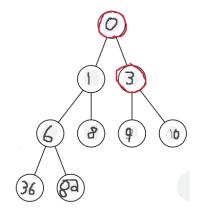
Once more:



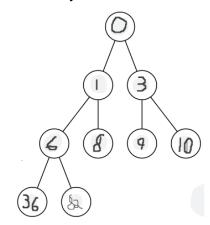
Again:



One final time:



And finally it is sorted:



#5

a.

- The root node is A[1]
- -Node i is A[i]
- -The parent of node i is A[i/3]
- -The left child of node i is A[3i]
- -The middle child of node i is A[3i + 1]
- The right child of node i is A[3i+2]

```
    b. Log3(n)
    c.
    EXTRACT-MAX(A) {
        if (heapSize < 1){
            Error
        }
        Max = A[1]
        A[1] = A[heapSize[A]]
        heapSize[A]--</li>
```

```
heapify(A,1)
           Return Max
   }
   Running time: O(log3(n))
d.
   INSERT(A, item) {
           heapSize[A]++
           i = heapSize[A]
           A[i] = -∞
           HeapChangeKey(A, i, item)
   }
   Running time: O(log3(n))
e.
   INCREASE-KEY(A, i, k) {
           if( k < A[i]) {
                  Error
           }
           Else {
                  A[i] = k
                  While (i > 1 && A[parent(i)] < A[i]) {
                         swap(A[i], A[parent(i)]
                  }
           }
   }
   Running Time: O(log3(n))
```