

OS Final

Winter 2016

North Weiner

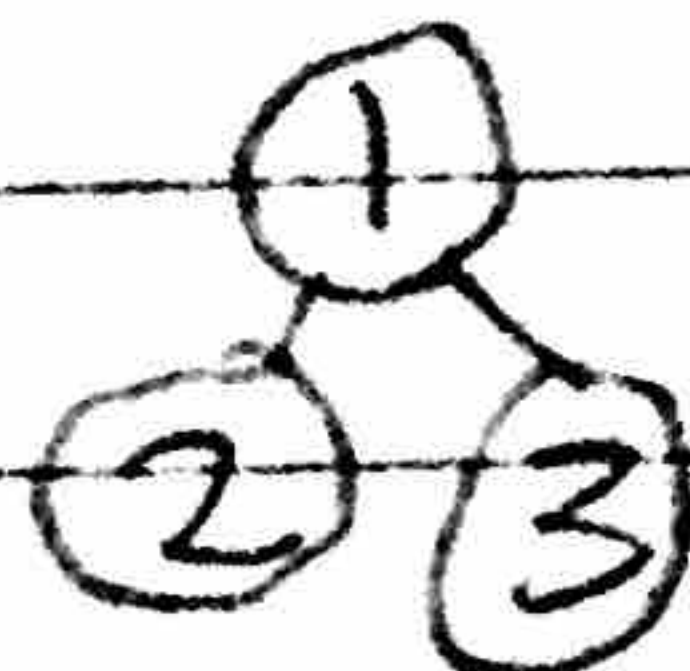
DS Final

1. Base case:

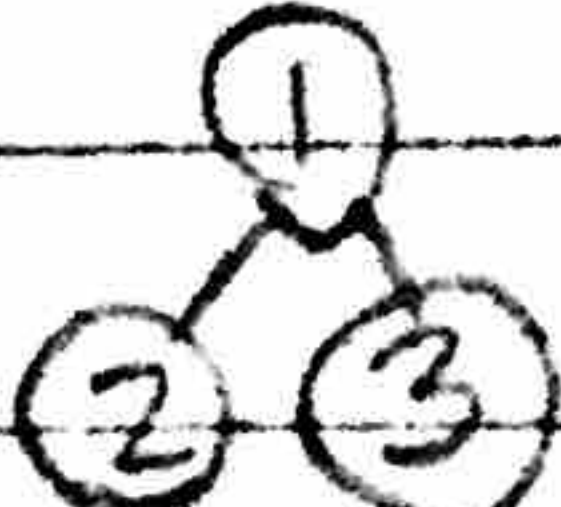
Insert 1 to $2^1 - 1$ keys.

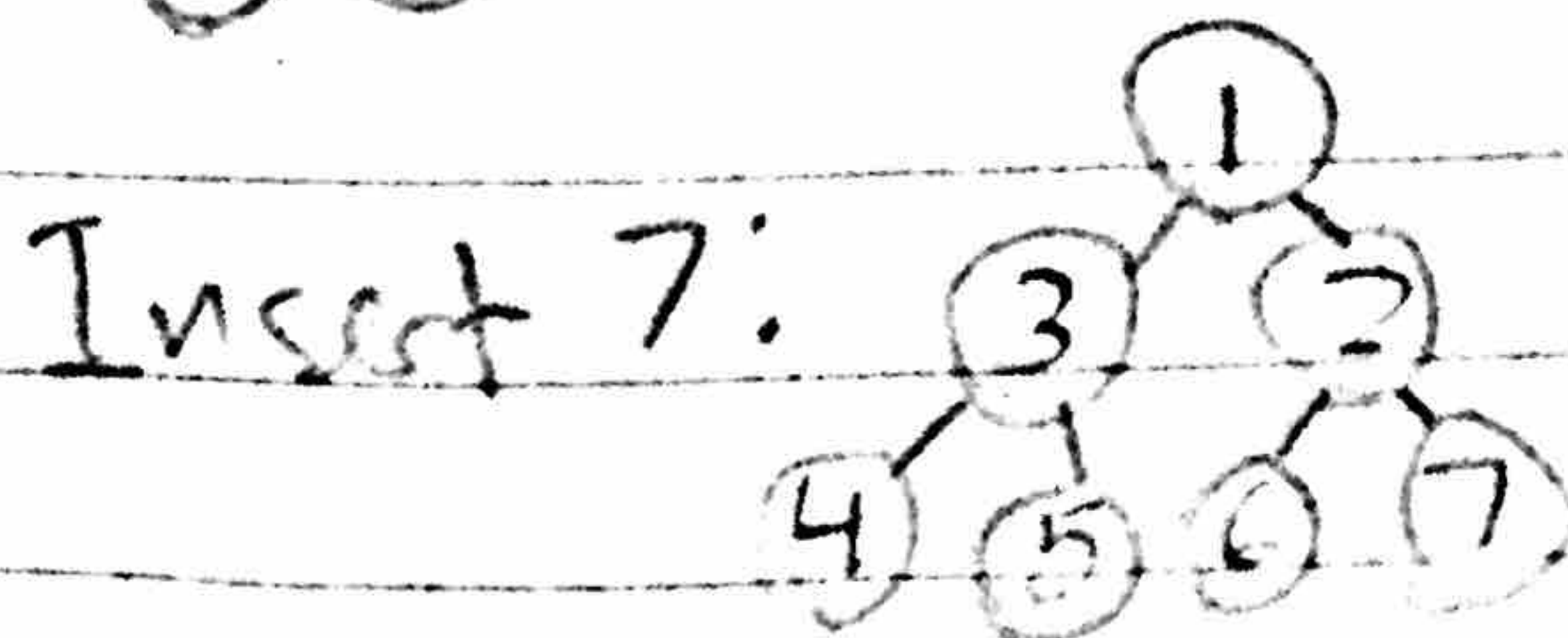
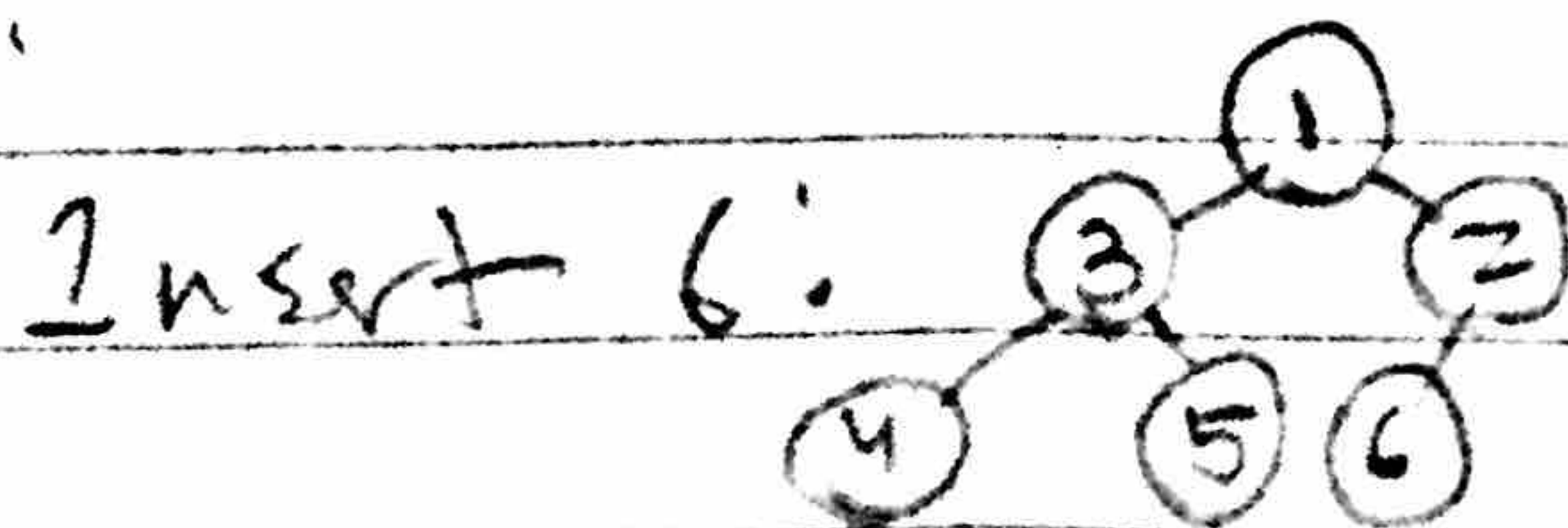
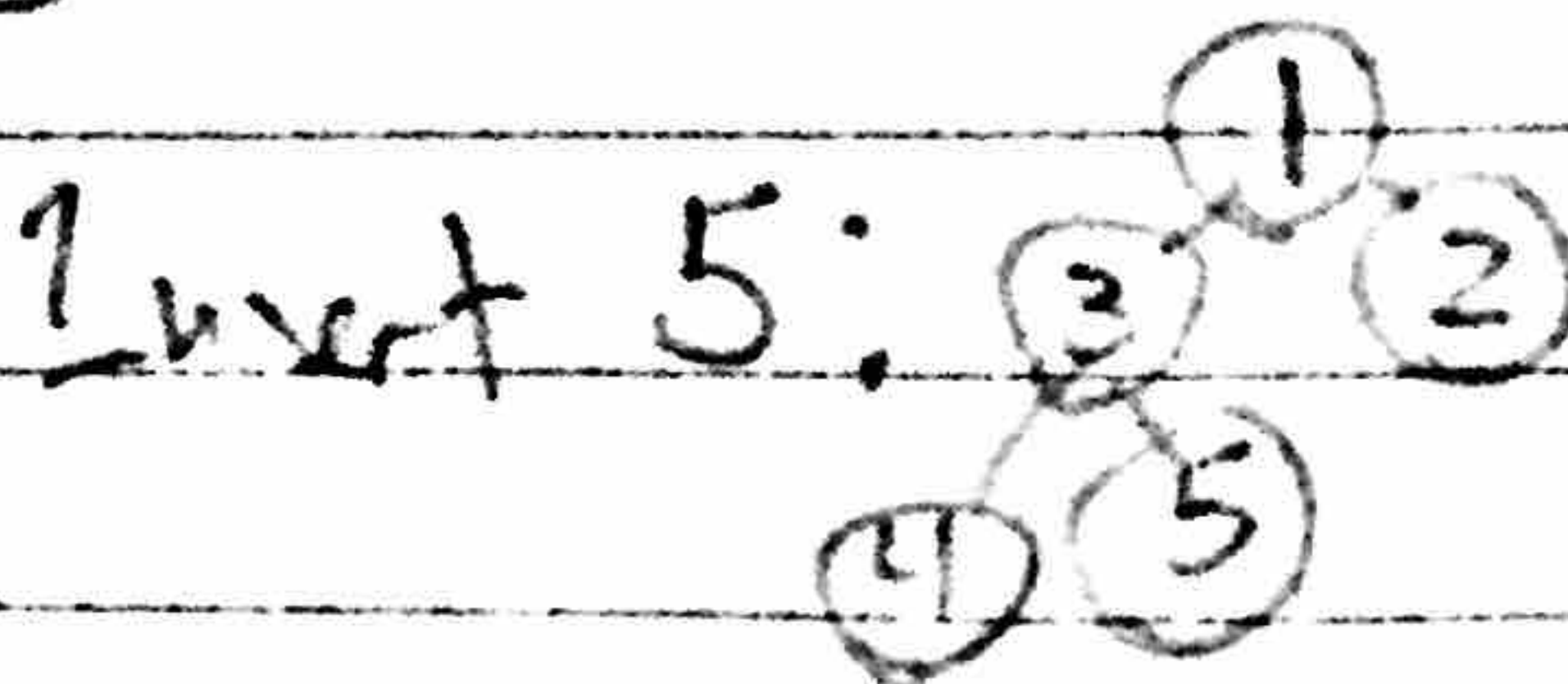
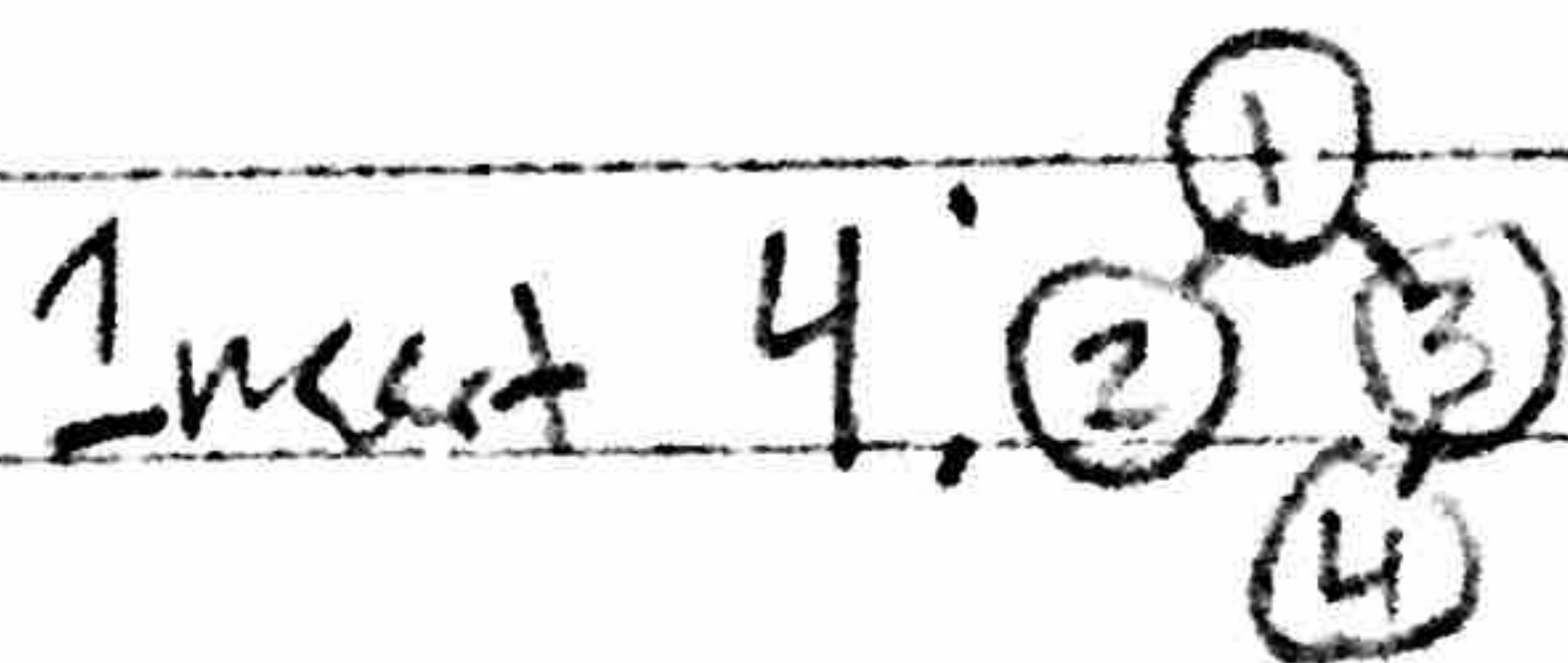


Insert 1 to $2^2 - 1$ keys.



If a tree elements 1 to $2^k - 1$ is balanced, inserting elements 2^k to $2^{(k+1)} - 1$ in order will fill a new a new depth of the tree, resulting in another perfectly balanced tree.

For a tree :



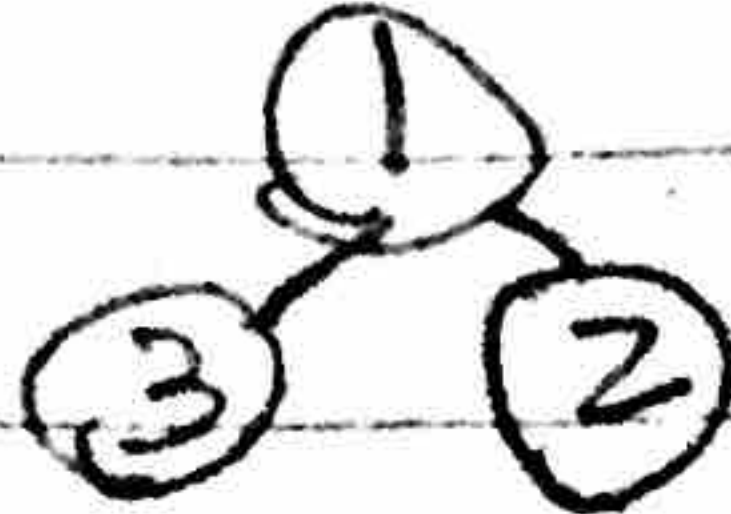
DS Final

2. Base cases

1 to $2^1 - 1$:



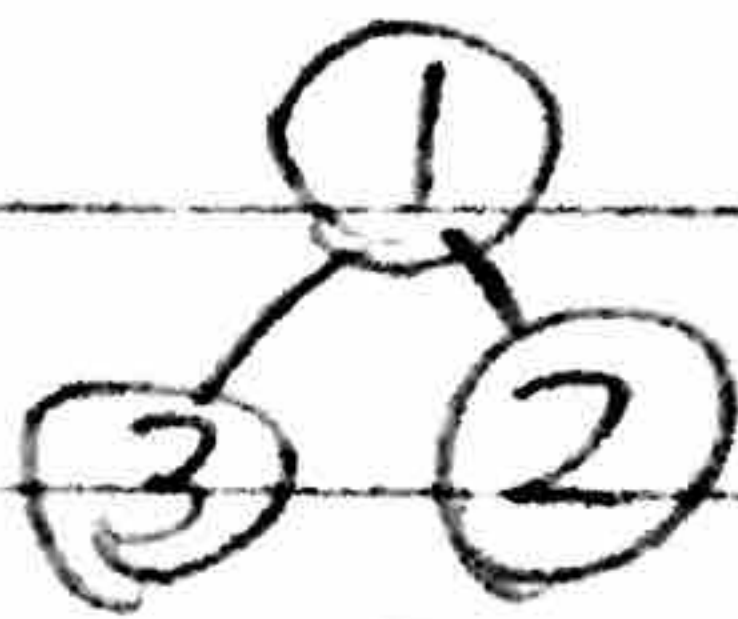
1 to $2^2 - 1$:



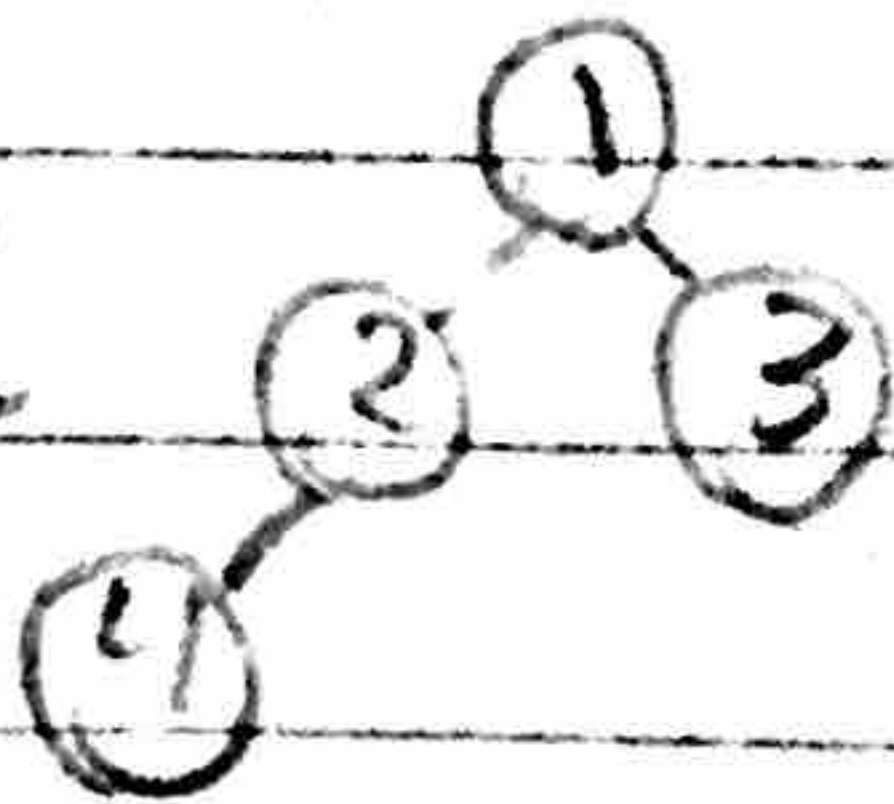
perfectly balanced

For each \checkmark tree of elements 1 to $2^k - 1$, inserting elements 2^k to $2^{k+1} - 1$ in order results in a perfectly balanced tree of a height one larger than the initial tree.

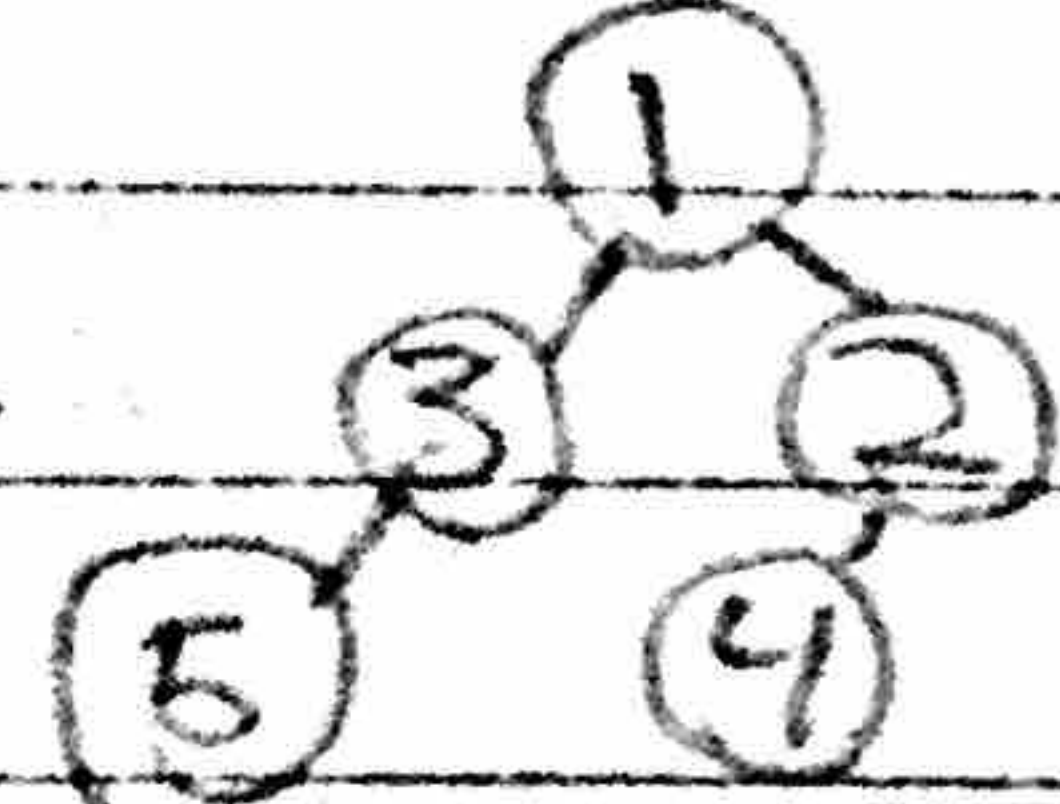
For tree



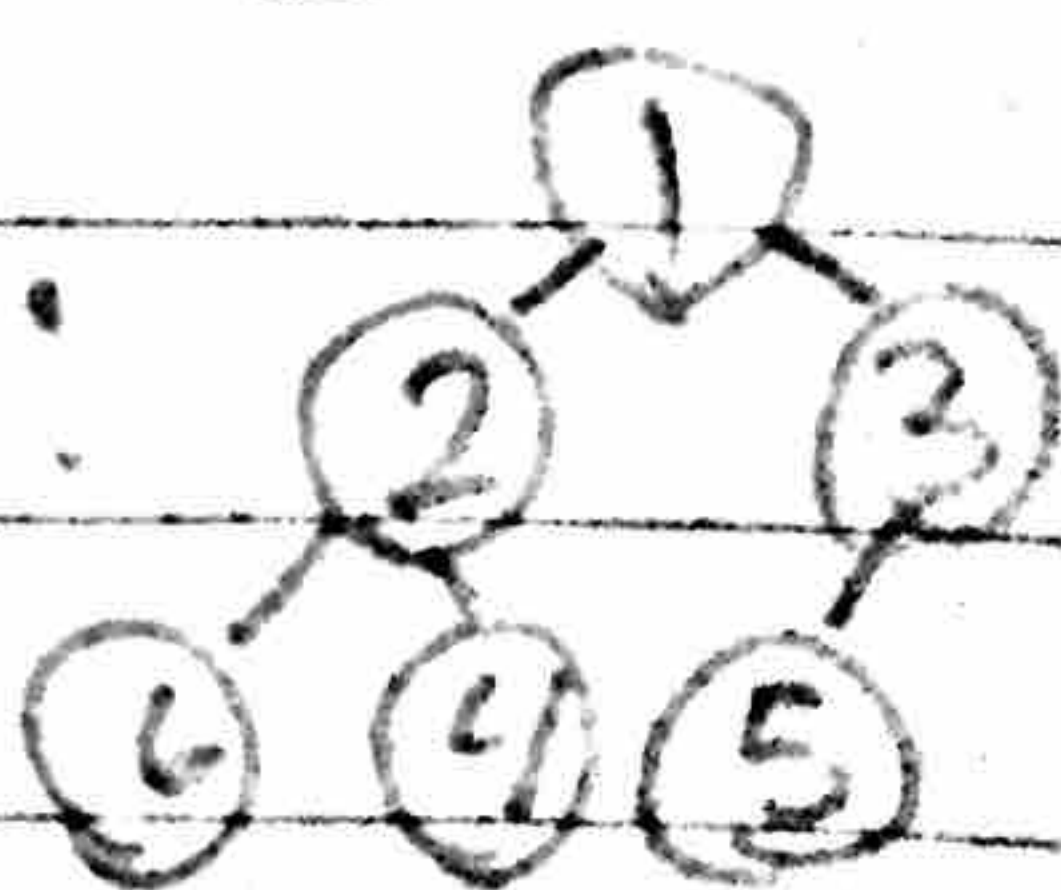
Insert (4):



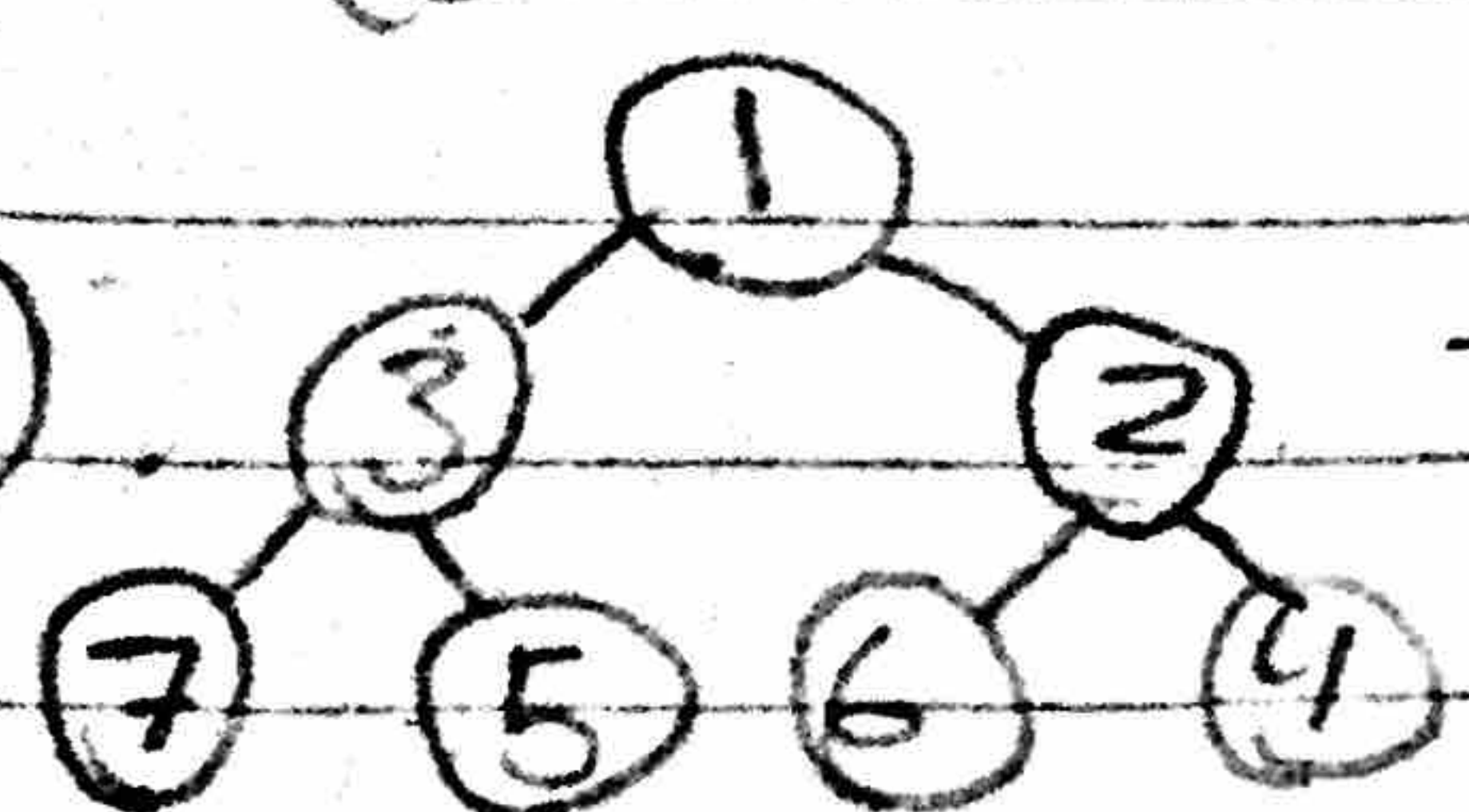
Insert (5):



Insert (6):



Insert (7):



DS Final

3 a) add a condition to the for statement on line 23 (4th ed figure 6.55)

```
for (int i = 0, j = 1;  
     j <= currentSize &&  
     !rhs.isEmpty() &&  
     carry == nullptr;  
     ++i, j *= 2)  
{  
    ...  
}
```

b) After line 10 (4th ed figure 6.55), insert:

```
if (currentSize < rhs.currentSize)  
    std::swap(this, rhs)
```

c) These work together to create a faster algorithm. the rhs tree will always be smaller, and the for loop will terminate as soon as rhs is exhausted.

DS Final

[L]

[K, G, I, K, N, V, S, S, W, Q]

[K]

[G, I, K], [N, V, S, S, W, Q]

[5] [142 | 543 | 123 | 65 | 453 | 879 | 572 | 434 | 111 | 242] 3
[811 | 102]

Heapify

[65 | 111 | 102 | 142 | 242 | 123 | 572 | 434 | 543 | 453 | 811 | 879]

Delete Min iterations:

1) [65 | 102 | 111 | 123 | 142 | 242 | 879 | 572 | 434 | 543 | 453 | 811]

2) [65 | 102 | 111 | 102 | 123 | 434 | 242 | 879 | 572 | 811 | 543 | 453]

3) [65 | 102 | 111 | 123 | 142 | 572 | 434 | 242 | 879 | 811 | 543 | 453]

4) [65 | 102 | 111 | 123 | 142 | 242 | 572 | 434 | 453 | 879 | 811 | 543]

DS Final

5 (continued)

Running time for pre-sorted input will be $N \log N + O(N)$, since it takes $O(N)$ to heapify and N delete-min operations, each of which performs $\log N$ operations.

6 a) because any 4 elements will have a total of 6 possible combinations. One pair can be sorted by default using some algorithms, leaving 5 necessary comparisons.

b) For array $[a, b, c, d]$: (mergesort)

if $b < a$: swap a, b

if $d < c$: swap c, d

if $a < c$:

if $b < c$: return $[a, b, c, d]$

else if $b < d$: return $[a, c, b, d]$

else: return $[a, c, d, b]$

else:

if $d < a$: return $[c, d, a, b]$

else if $b < d$: return $[c, a, b, d]$

else: return $[c, a, d, b]$

17

a) $N \log N$

b) $2N \log N$