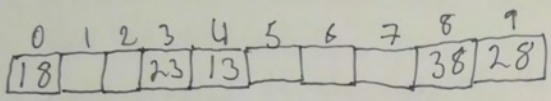
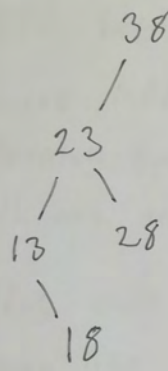
1 a) 13, 18, 23, 38, 28

1 b)



1 c)



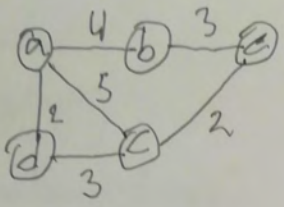
2a) Starting with the tree root as current node

1. Return true if current node is nil or has no children.
2. Return false if the key of the current node is less than either left or right child key if they exist.
3. Process left and right child recursively returning true only when the result of both are true. Otherwise false

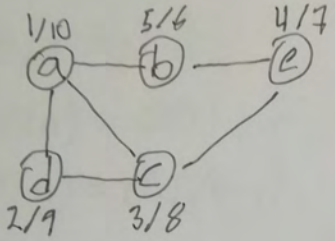
2b) Starting with the tree root as current node.

1. Return true if current node is nil or has no children
2. Return false if the key of the current node is less or equal to the left child's key, or if it is greater than the right child's key if they exist
3. Process left and right child recursively returning true only when the result of both are true, otherwise false.

3a)



3b)



3c) i) (a, d, 2]
(d, c, 3)
(c, e, 2)
(e, b, 3)

ii) (a, d, 2)
(c, e, 2)
(d, c, 3)
(c, b, 3)

4. a) False

b) $\Theta(|V|^2)$

c) False ($m=1$)

d) $\Theta(n \cdot \log^2 n)$

e) $T(n) = O(1)$, if $n \leq 1$

$T(n) = 2^n \cdot T(n-1)$, if $n > 1$

f) True

g) True. $\Theta(k \cdot \log|V| + |E|)$

h) True.

5.1) c

5.2) b

5.3) d

5.4) d.

5.5) d

6.) Starting with the tree root as current node.

1. if current node is nil
   return perfect = true and size = 0.
2. Process left child recursively assigning the result to P_L and S_L.
3. Process right child recursively assigning the result to P_R and S_R.
4. assign size to $S_L + S_R + 1$
5. set perfect to false
   if either P_L or P_R is false
   or if $|S_L - S_R| > 5$, otherwise set perfect to true.
6. return perfect and size.

7.) Given the input A array with length n.

1. if OR(A) is false return an empty list.
2. if A is empty return an empty list.
3. if A has a singel element then return it as a list.
4. recursively process left half of A into L.
5. recusively process right half of A into list R.
6. return the joined list of L with R at the end.

1, 2, 3 and 6 are all $O(1)$

4 and 5 are both $T(n/2)$ which gives:

$$T(n) = 2T(n/2) + O(1) = 2^m \cdot T(n/2^m) + O(1)(2^{(m+1)} - 1)$$

Worst-case is when the bad k elements are placed n/k apart. With 1 bad element 4 an 5 are $T(n/2)$ together which gives

$$T(n) = T(n/2) + O(1) = O(\log n)$$

$$\Rightarrow T(n/k) = O(\log n/k), \quad n/2^m = n/k \Leftrightarrow m = \log k$$

$$T_{worst}(n) = 2^{\log k} \cdot T\left(n/2^{\log k}\right) + O(1)\left(2^{\log k + 1} - 1\right)$$

$$= k \cdot O(\log n/k) + O(1)(2 \cdot k - 1) = \underline{O(k \cdot \log \tfrac{n}{k})}$$