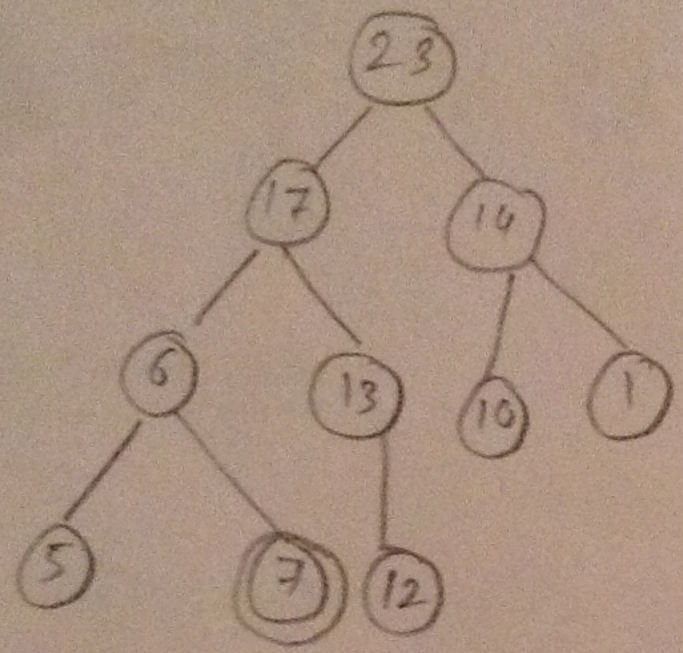
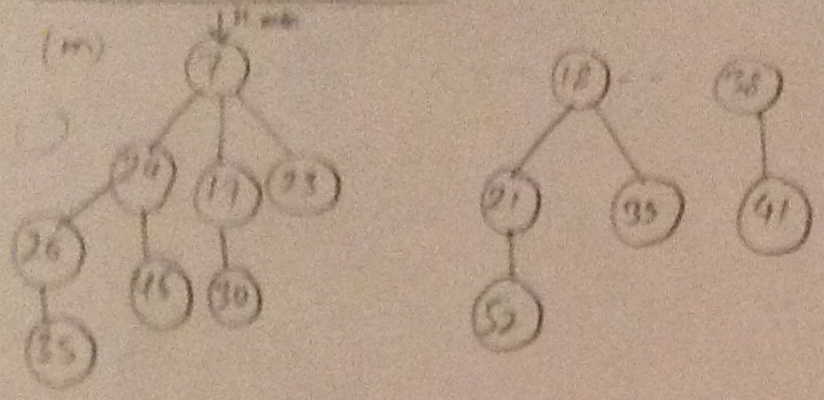
① 6.1-6

No.

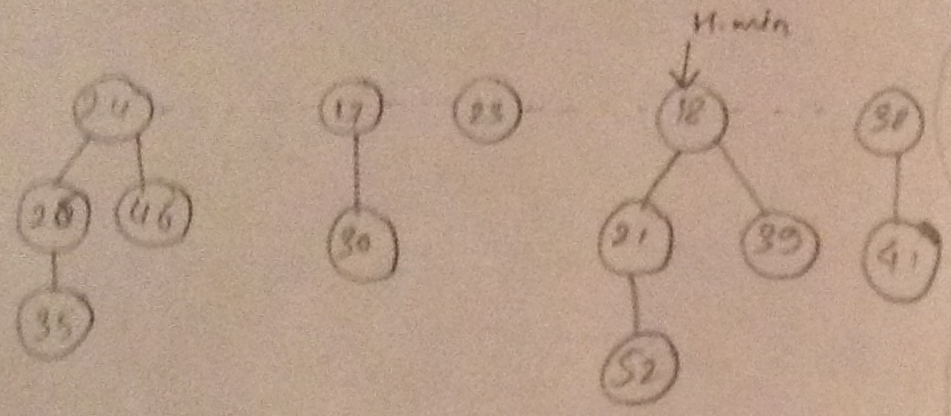


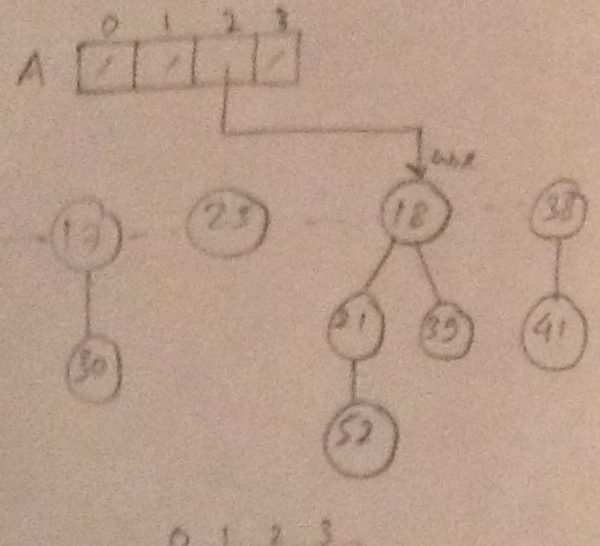
Here, I violates the max-heap property i.e; every node is smaller than its parent in max-heap

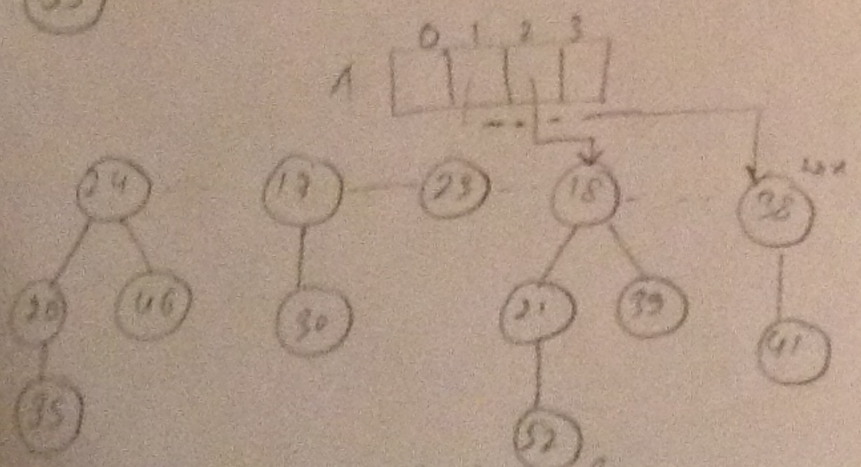
(2) 9-2-1

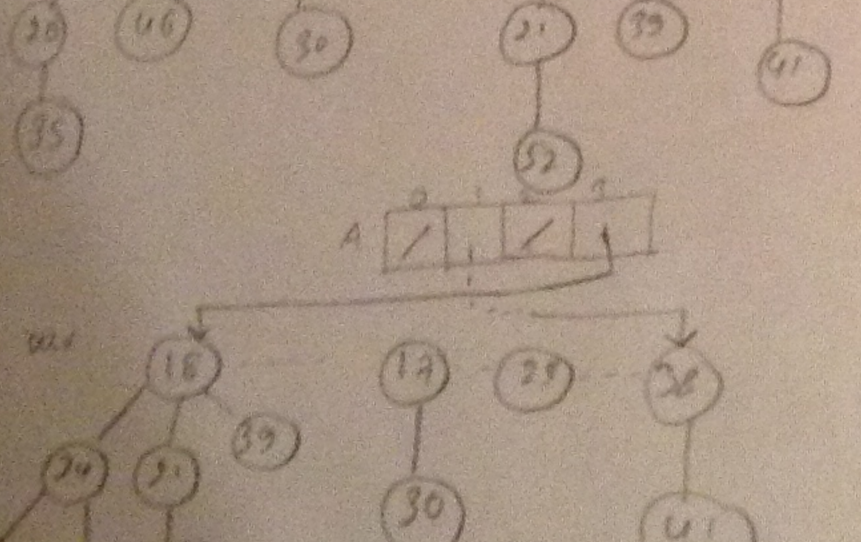


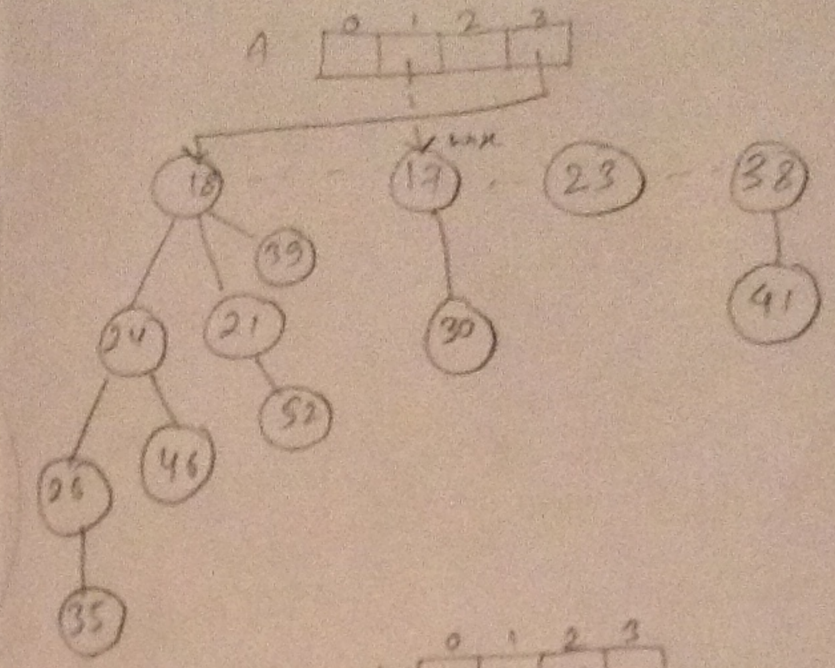
Fib-Heap-Extract-Min

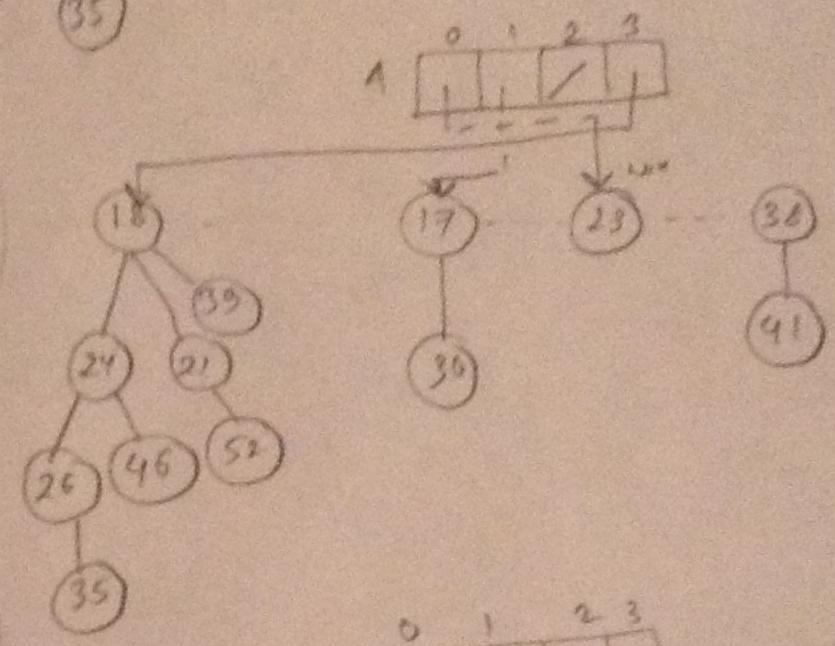


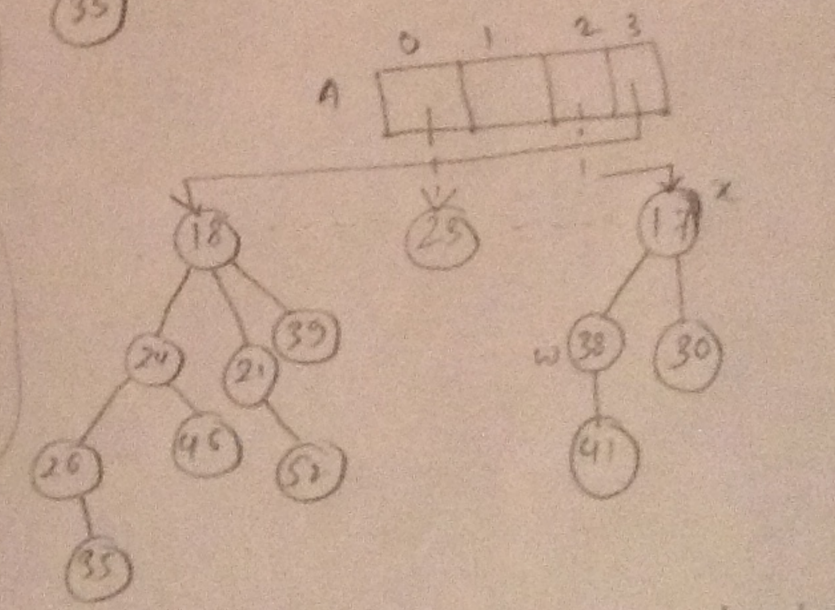


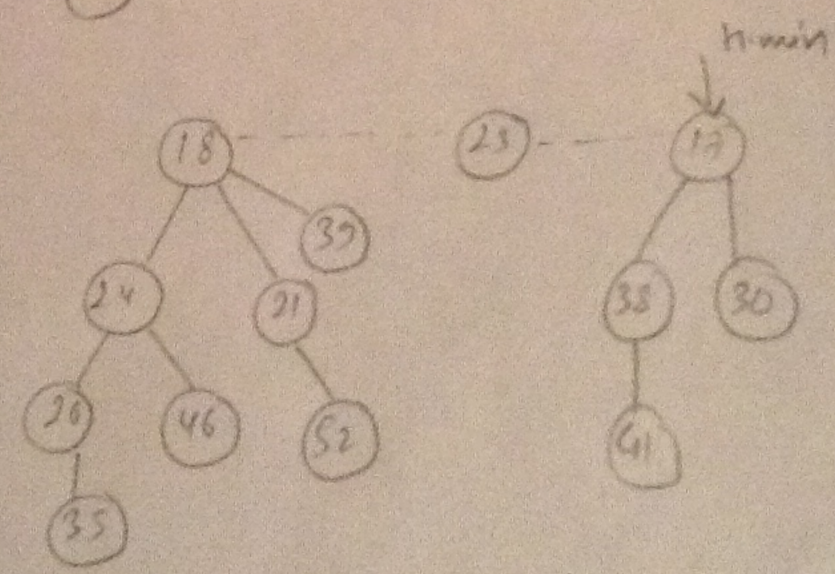






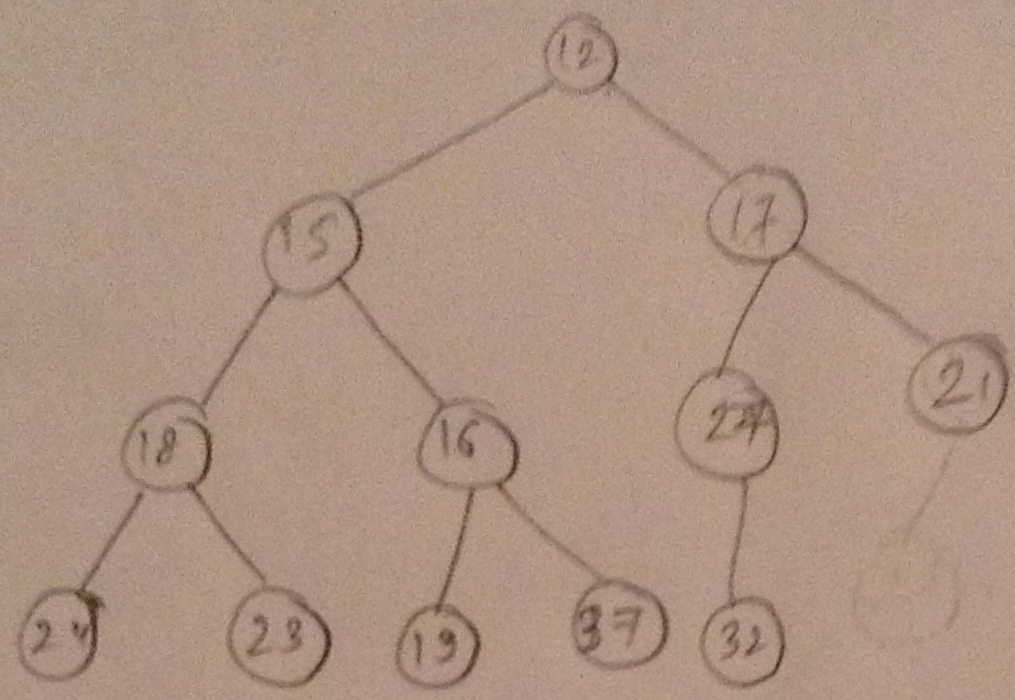




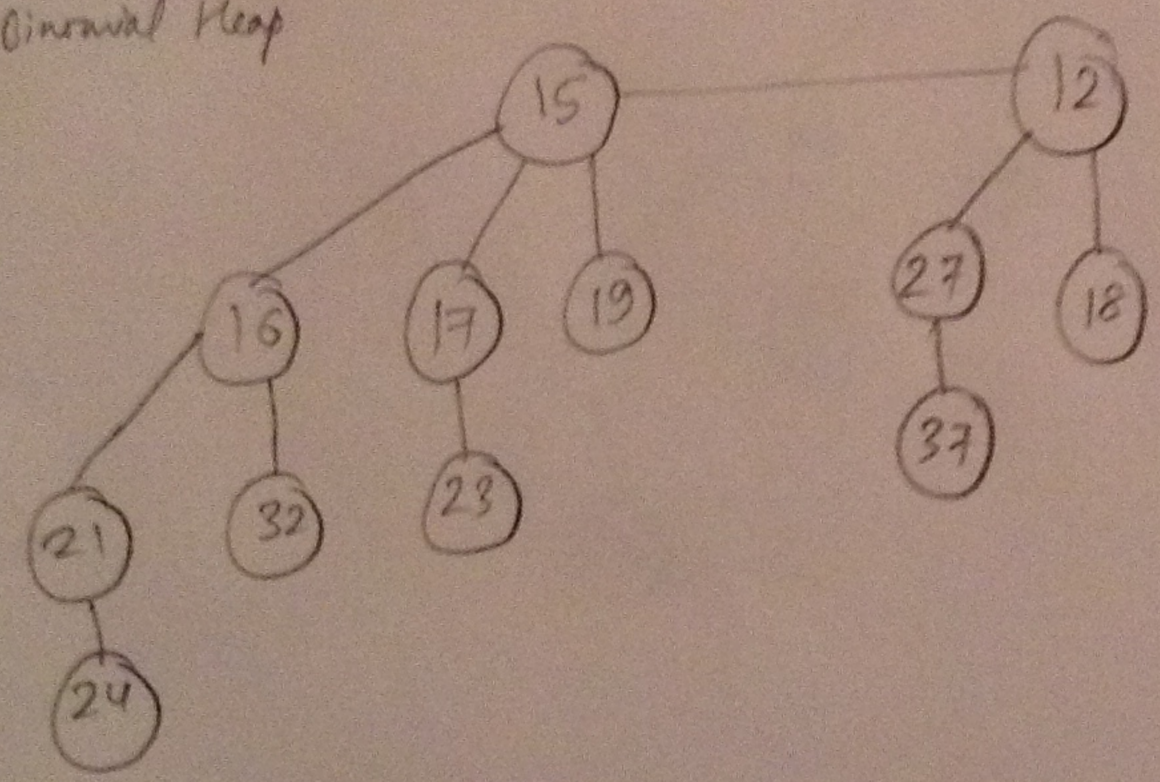


3 (a)

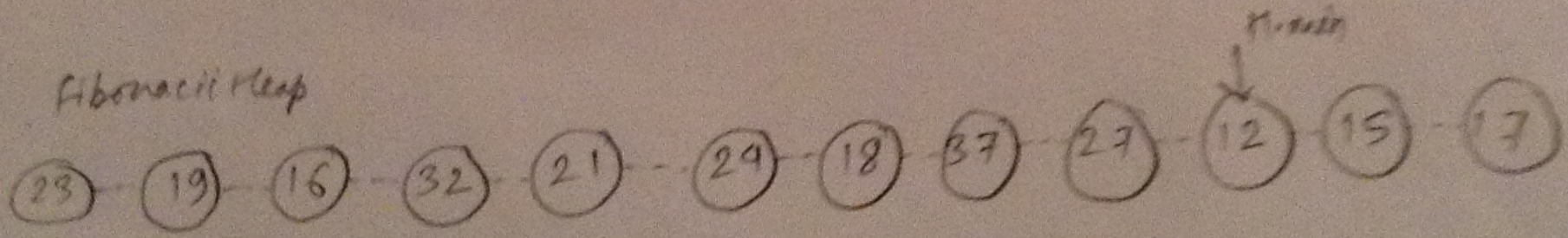
Heap (Min)



(b) Binomial Heap



(c) Fibonacci Heap

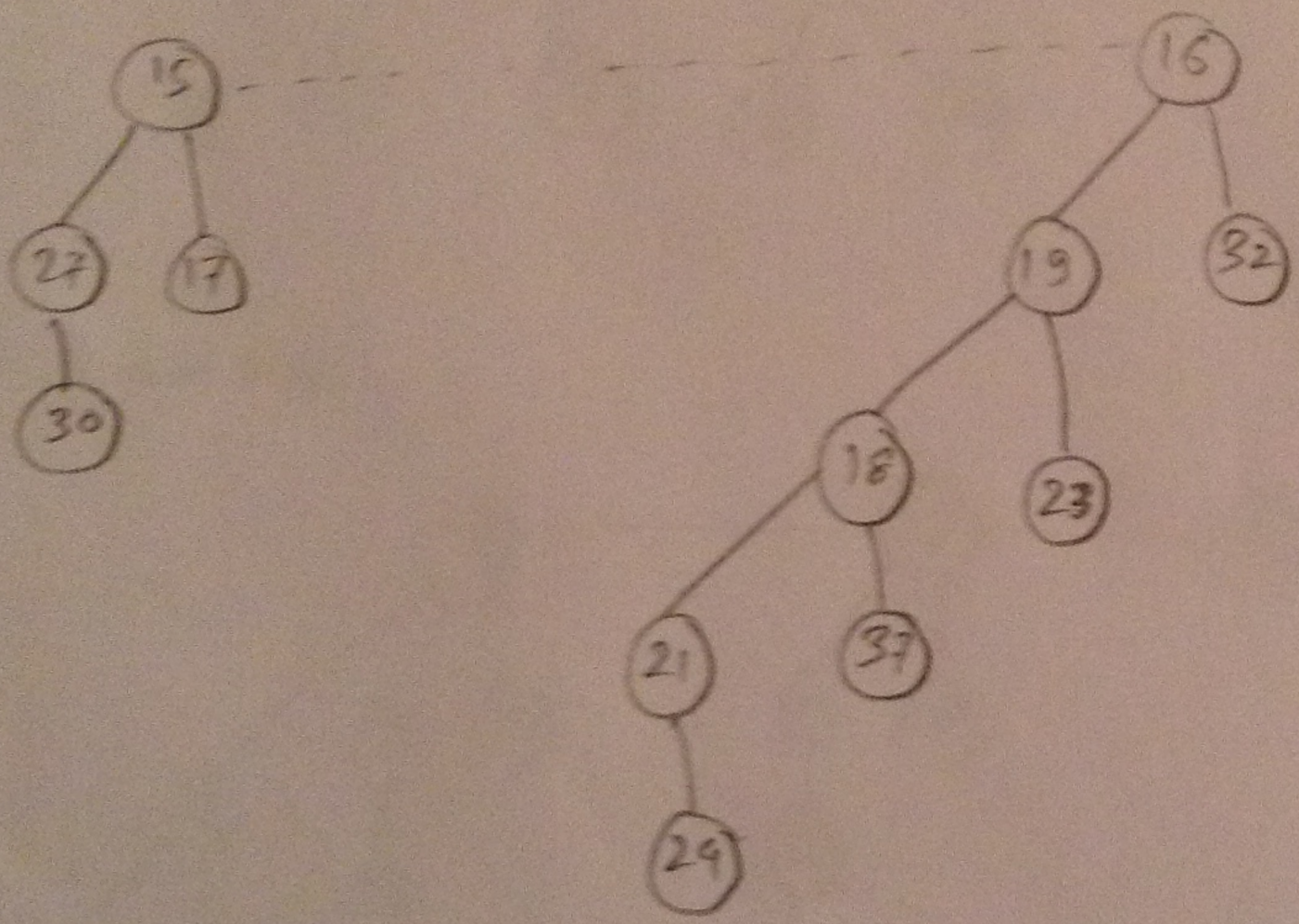


3(d)

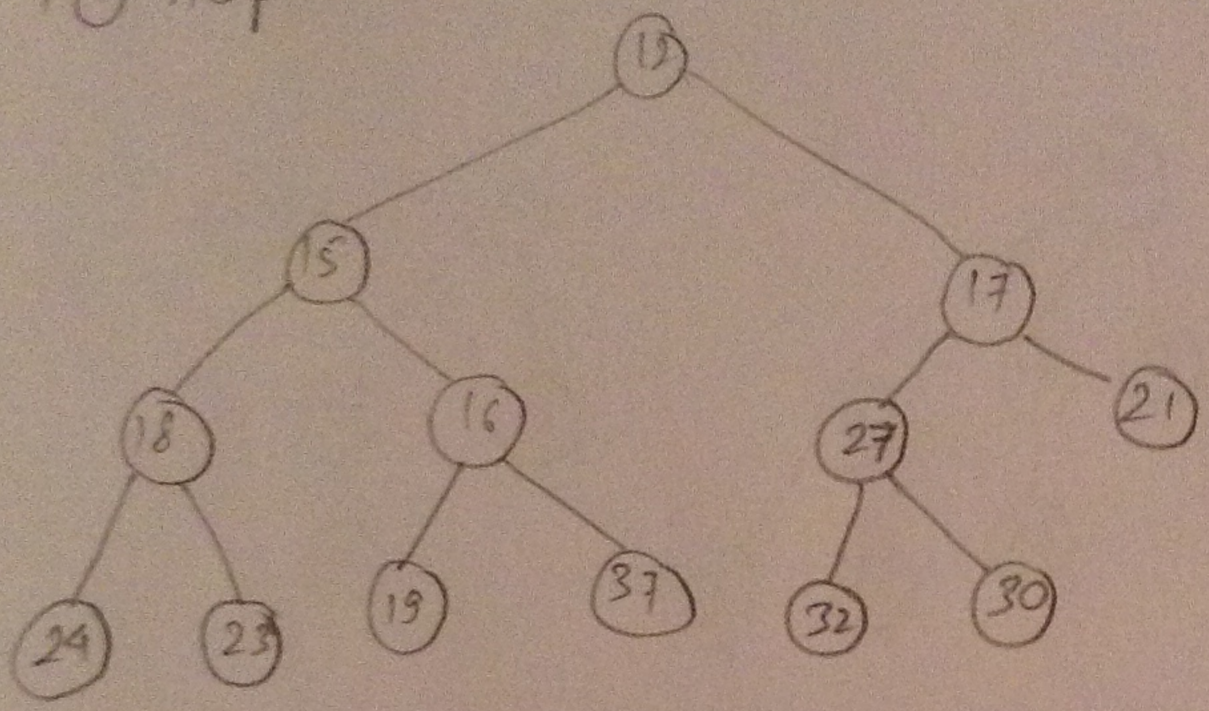
H.min

After

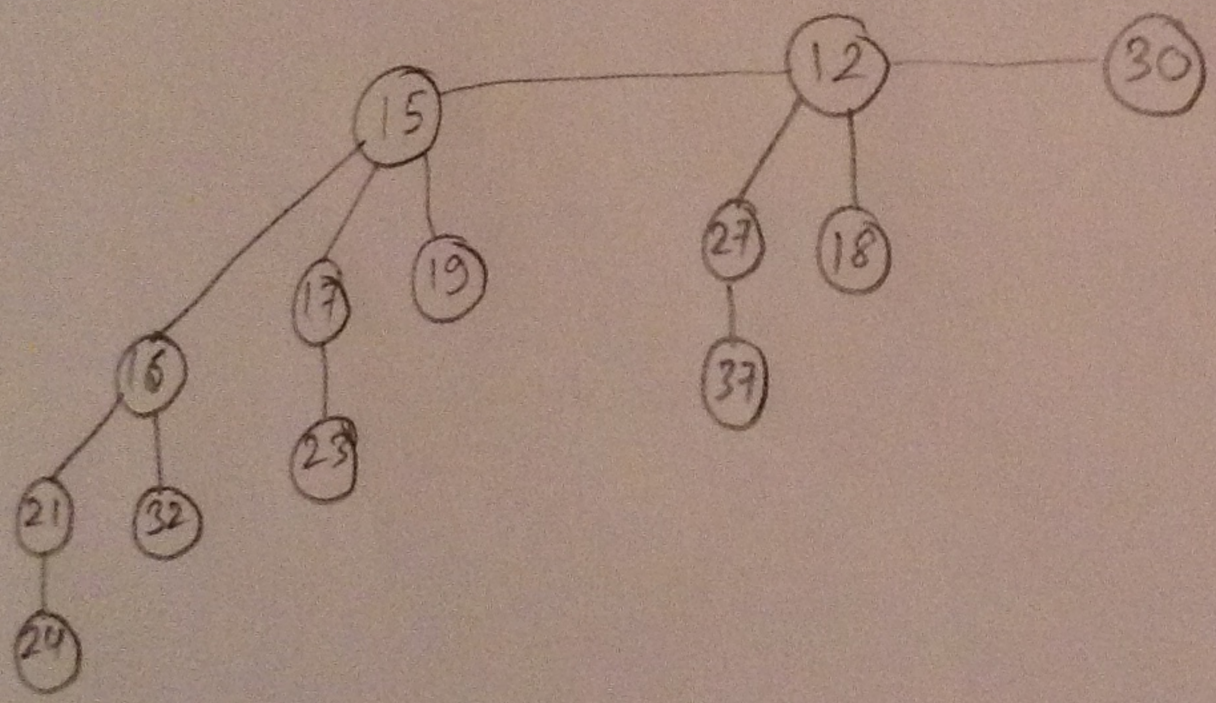
Fib-Heap-Extract-min



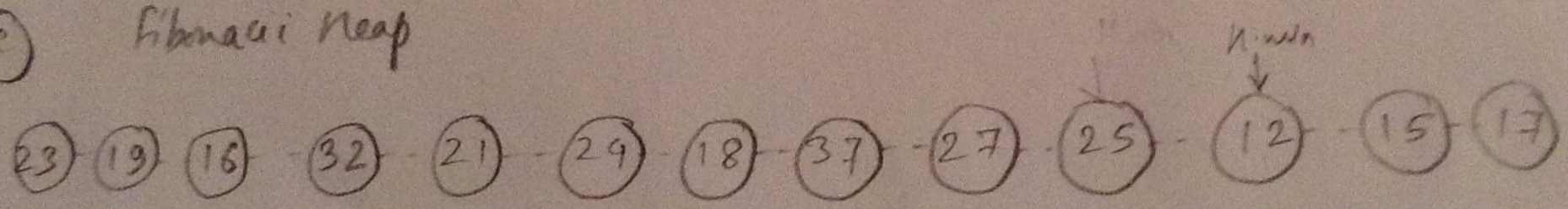
4 (a) Heap



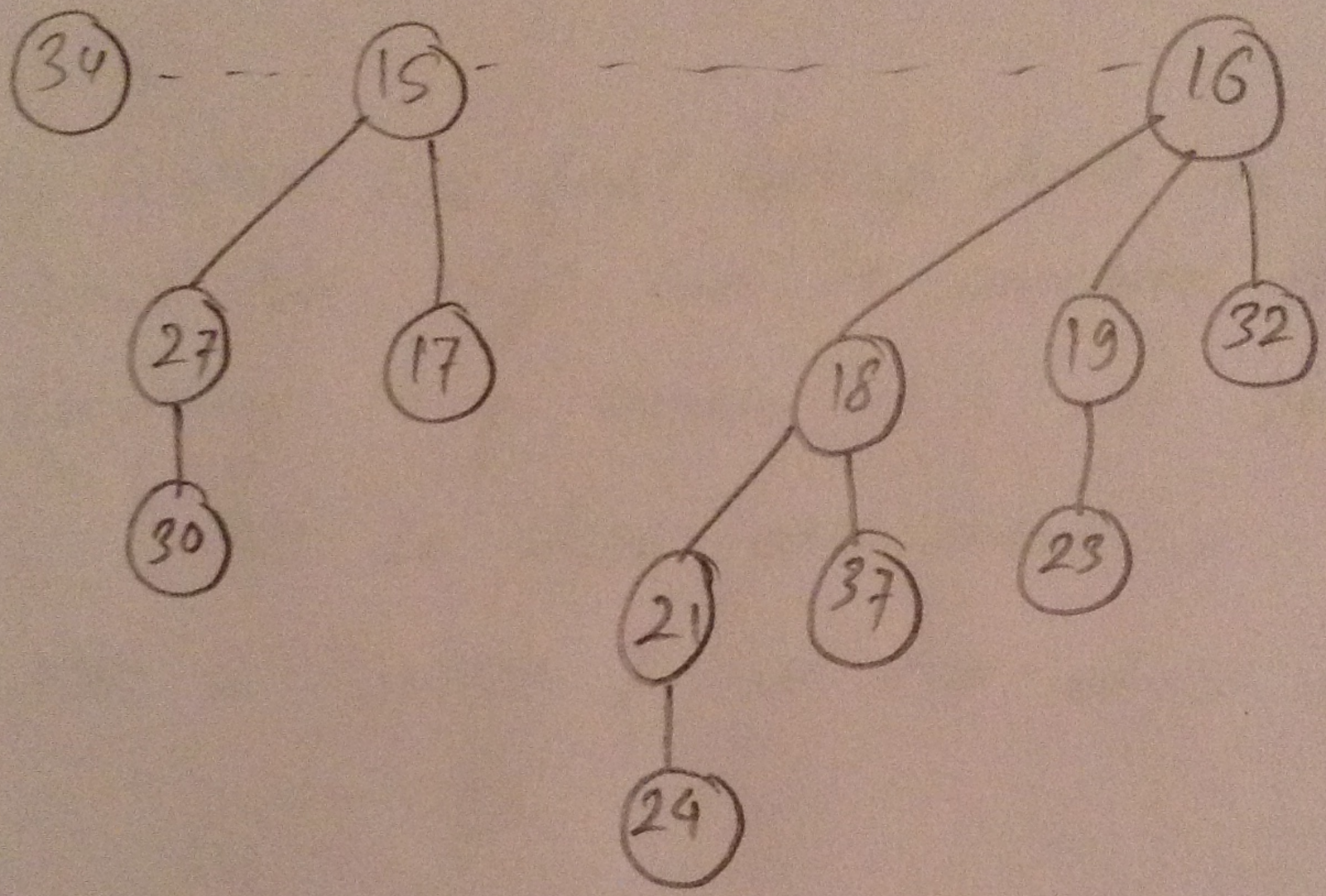
(b) Binomial Heap



(c) Fibonacci Heap



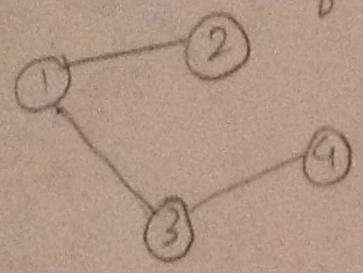
40



(5)

Adjacency matrix

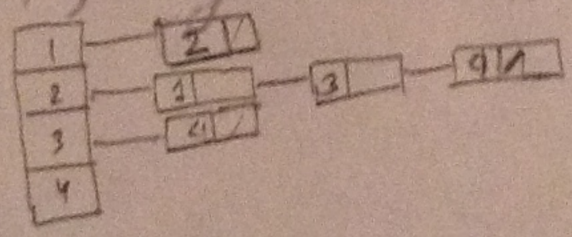
Adjacency matrix has fast lookup time to find a particular edge i.e; O(1), because the edge can be accessed through the index. However, if memory is a concern it is not suitable, because it requires $O(n^2)$ memory. Here the edges are stored in a nxn sized array. Adjacency matrix is slow to traverse if all the edges have to be accessed.



	1	2	3	4
1	0	1	1	0
2	1	0	0	0
3	1	0	0	1
4	0	0	1	0

Adjacency List:

Adjacency List may be used because it uses less memory compared to Adjacency matrix. It uses memory in relation to the number of edges, therefore usually it saves memory. It is also faster to iterate through all the edges, because the list is easy to traverse. However, it is not as efficient as Adjacency matrix to lookup a particular edge



Edge List:

Edge list is just an array that stores the pair of vertices that make up the edges. This property makes it easy to implement. However, since there is no straight forward traverse plan to the linked vertices, it should be looked up through the array to find other edges and maintain the traverse logic externally. However, edge list can be used for algorithms like minimum spanning tree & Kruskals algorithm

1	1	3
2	3	4

This stores the pair (1,2), (1,3) & (3,4)