Selection Algorithms

Main Focus: The Tournament Method

We analyse and discuss algorithms for finding the second largest element in a list of elements

(We automatically end up finding for the largest as well)

Brute Force Method:

1. We find the largest element. It takes $n-1$ comparisons.
2. Pop the largest element from the list.
3. Repeat the procedure to find the largest element again. It takes $n-2$ comparisons.

> Total Number of Comparisons: $(n-1)+(n-2)=2n-3$

In order to reduct the number of comparisons given above, we learn and implement the 'Tournament Method'.

The Tournament Method:

Since we are just understanding the intution behind the algorithm, we assume that all numbers in the list are distinct and that n is power of 2. We pair the keys and compare these pairs in rounds until only one round remains.

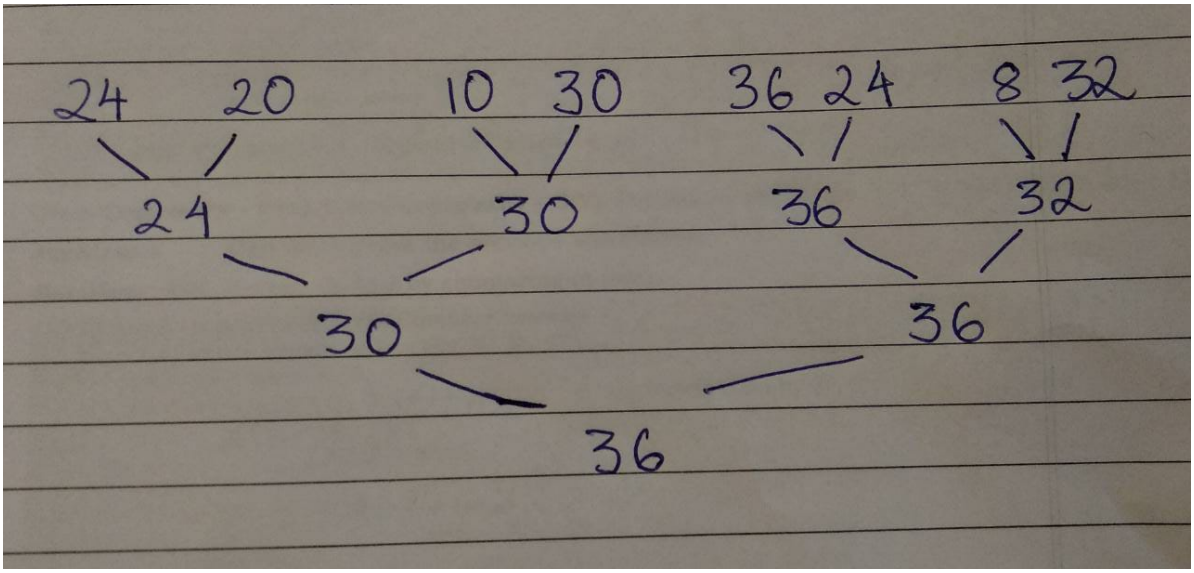
Now, when we discard the assumptions, we must realise that that this method *only works for powers of 2*. If this condition is not met, we can append enough items to the end of the list/array to make its size the power of 2. If the tree in complete, then the maximum height of the tree is $logn$.

To find the largest element it takes us $n-1$ comparisons. But, we need to find the second largest element. In order to find the second largest element, it becomes obvious that this element must have lost in comparison to the largest element at some point in the tree. The number of elements which would have lost to the largest element would simply be the height of the tree which is $logn$.

Now, we take those elements and use selection algorithm to find the largest among them. This is $logn-1$ comparisons.

> The beauty of the above approach is that we can find the largest and second largest elements in just: $(n-1)+(logn-1)=n+logn-2$.

The figure which explains the above 'Tournament Method' is:


24 20 10 30 36 24 8 32
24 30 36 32
30 36
36