Prob. 1	Prob. 2	Prob. 3	Prob. 4

Problem 1.

We can use contradiction or construction:

    max: sth like suppose there is another binary tree with greater potential than the binary list but with the same n number of nodes that would mean that at some height of the tree there is one node having more nodes in its sublist than the equivalent node in the linked list however this would be impossible since in a linked list the only nodes of a tree that are not included in a node's subtree are its parent nodes

    well we can just say that the w(root) = n-1, and the biggest weight for its child is n-2 constructing such a tree will end up with a linked list

    min: well summing n/2 + n/2 or (n/2)-1 + (n/2)+1, doesnt matter, we get n BUT by doing n/2 + n/2 we will have the tree with the smallest height -¿ smallest number of element in the sum

    and besides this way we maximize the number of nodes with no children

Problem 2.

Problem 3.

We can assume that we have a requested sequence of elements :

$$A_1, A_2, A_3, , A_{n-1}, A_n$$

Then no matter what the initial structure of our elements was, the optimal stati c algorithm would produce a sequence that is exactly the same with the requested one:

$$A_1, A_2, A_3, , A_{n-1}, A_n$$

Now let us suppose that the initial structure of our elements is:

$$A_n, A_{n-1}, , A_3, A_2, A_1$$

MTF actual cost analysis:

Given the above data structure and the requested sequence, the move-to-front alg orithm would need to search until the end of the list each time, and then move t hat element to the front. Given that an element can be moved to the front in con stant time, the actual cost for the given sequence would be n2.

OPTIMAL-STATIC cost analysis:

It appears that since the sequence of elements in the optimal list is the same w ith the sequence of requested elements, the actual cost of would be the sum of an arithmetic progression with a1 = 1 and an = n. Thus the actual cost would be

$$S = \frac{n(1+n)}{2} = \frac{n2+n}{2}$$

$$\lim_{n \to \infty} \frac{n^2}{\frac{n^2+n}{2}} = \lim_{n \to \infty} \frac{2n^2}{n^2+n} \text{(using rule de l' hospital to calculate the limit)}$$
$$= \lim_{n \to \infty} \frac{4n}{2n} = 2$$

As a result, we can claim that since the cost of the move-to-front algorithm is twice as that of the static optimal algorithm in the limit, then the competitive analysis for the move-to-front algorithm against the static optimal algorithm is asymptotically tight.

Problem 4.