1.) linked list: All main memory

$16 \times 100 = \boxed{1{,}600 \text{ ns}}$

vector:

$$\begin{array}{ccccccc} 0 & & 1 & & 2 & & 3 \\ 100\text{ ns} & + & 7\text{ ns} & + & 7\text{ ns} & + & 7\text{ ns} \end{array} = 121\text{ns} \times 4 = \boxed{484\text{ ns}}$$

main memory →

cache

2.)

(a.) This could definatly be parrallized

Steps: divide linked list into sections based on the number of proccessors/threads

2. Push the largest from each section into a Queue

3.) return max(Queue)

b.) I dont think this should be parrallized unless the stack can support pushing multiple thires at once and the order of the stack does not matter. If only one element can be pushed at a time, than a mutex would be needed, and at that point it is bassically serial.

c.) I dont think this should be parrallized because of the unique structures of binary search trees. When searching these trees there is only one possible path from the head to the wanted node, so multiple threds are kinda pointless.