Q1)

The function returns a tuple containing the maximum total value and the list of indices corresponding to the path that yields that value. The function works by first checking if the current position is the last element in the points array. If this is the case, the function returns the value at that position and the current path. If not, the function generates a list of possible moves and calls itself recursively on each of these positions, keeping track of the maximum total value and the corresponding path.

The time complexity of this function is O(2^n), where n is the number of elements in the points array, because at each recursive call, the function makes two recursive calls on the next element in the points array.

Q2)

These functions contain an implementation of the selection sort algorithm for finding the kth smallest element in an array.

The FindMedian function takes as input an array arr and returns the median element in the array. It does this by calling the helper function, which is an implementation of selection sort, with the appropriate value of k.

The helper function is a selection sort algorithm and it takes as input an array arr and an integer k and returns the kth smallest element in the array. It does this by iterating through the array and swapping the minimum element found so far with the current element until it reaches the kth element. It then returns the kth element.

This implementation has a time complexity of O(n^2) and a space complexity of O(1), since it sorts the array in place.

Q3)

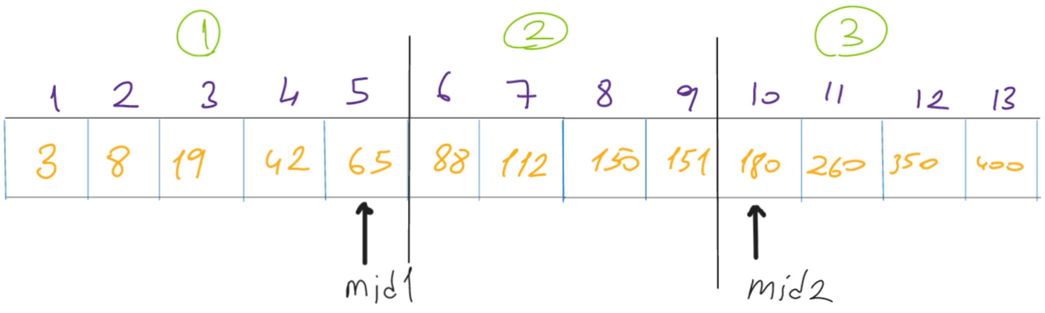
The Q3_A method traverses the list, setting each node's next pointer to the node two steps ahead. It then returns the data value of the last node in the list. This method will return the data value of the middle node if the list has an odd number of nodes, and the data value of the first node in the list if the list has an even number of nodes.

The time complexity of that is O(n), where n is the number of nodes in the circular linked list. This is because the function traverses the entire list once, performing a constant number of operations on each node.

The Q3_B method finds the middle node of a circular linked list using two pointers that move at different speeds. The FindMid method is a helper function that uses a recursive binary search approach to find the middle node of a sublist between two nodes. The time complexity of these methods is O(log n), where n is the number of nodes in the list, and the space complexity is O(1).

Q4)

In Ternary search algorithm, at each step the array will be divided 3 parts with 2 divisors. (mid1 and mid2) at each step we compare searching values with mid1 and mid2 and then decide where to look at the next step:



According to comporison's result, we will know where to look at the next level(1st or 2nd or 3th part of array). We will do $\boldsymbol{log_3 n}$ comparisons and complexity will be **O($\boldsymbol{log_3 n}$).**

In the Binary search algorithm always there will be 2 parts and we will do total $\boldsymbol{log_2 n}$ comparisons so that comlexity will be **O($\boldsymbol{log_2 n}$).**

if we divide the array into n parts at the beginning, the time complexity of the algorithm becomes $\boldsymbol{O(n)}$**,** since the algorithm will need to check every element in the array. This is because at each step, the algorithm will create n subproblems, one for each part of the array, and the number of steps required to complete the algorithm will grow linearly with the size of the input

Q5)

a) The best case of the interpolation search is $\boldsymbol{O(1)}$**.** This happens when the searching is completed at one step. Generally, if the target value is **in the middle** of the array then it will be found at one step.

b) Complexity of the Interpolation search is on average **log(log(n))**, on the other hand binary search is **log(n).** this means that on the average case the interpolation search is more effective. When performing a search, binary search always checks the element of the array, while interpolation search may start searching at a different location based on the value of the search key. If the value of the search key is closer to the end of the array, interpolation search will go to search towards the end of the array. Worst case time complexity of binary search is **O(log n)** on the other hand interpolation search's is **O(n).** In the worst-case scenario binary search is more effective

Hacı Hasan Savan

1901042704