COMP3121 21T2 Assignment 1 Q4

Written by Zheng Luo (z5206267)

Binary search can be implemented to search the existence of given target value in a sorted array, in this case, the given array is sorted, but the target value is missing, hence the standard binary search cannot be implemented. However, a modified version of binary search can be used with time $O(n)$ and $2^n - 1$ given values. The $2^n - 1$ represents the binary tree with root counted as level 0, and the complexity of $O(n)$ means the height of binary tree, which indicates the missing value has to be found by searching only one specified direction in each level. Here is the process for modified binary search:

1. Each time binary search function locates the middle value of array, it has to check its preceding value is 1 unit smaller than current value, and following value is 1 unit larger than current value, otherwise the missing value is the one which failed the checking function.
2. Then the proceeding direction of either $LowerBound$ to $(current - 1))$ or $current + 1$ to $higherBound$ by comparing $currentMiddleValue - nextSmallerMiddleValue$ and $nextLargerMiddleValue - currentMiddleValue$, the function will go to the lower bound if first function is greater than second function, and vice-versa. Because the side with larger compared value means that side has missing term compare to another side.
3. Repeat the process until step 1 failed.

For example, when $n = 3$, and array of [1,2,4,5,6,7,8].

1. locate 5, check left is 4 and right is 6, proceed to next step.
2. left middle is 2, right middle is 7, and $5 - 2 = 3$ and $7 - 5 = 2$, left side is larger, so left side has missing value, proceed to left side.
3. locate 2, check left is 1 and right is 3, but there is no 3, so missing value is 3.

This algorithm satisfied the requirement of determine the missing term with at most $O(n)$ complexity.