Marcus Anderson

Homework 2 - Arithmetic Progression

CS 6515: Introduction to Graduate Algorithms

**Algorithm: Modified Binary Search**

- We will use a modified binary search.
- First, before starting the recursive search, find the common difference by subtracting A[n] by A[1], and dividing the difference by size n. This will be called *common_diff*.
    - *common_diff* = (A[n] - A[1]) / n
- Next, during the recursive search:
    - First, check for the missing value by looking at the left and right elements of A[mid].
        - If A[mid - 1] != A[mid] - *common_diff*, return A[mid] - *common_diff* as the missing value.
        - If A[mid + 1] != A[mid] + *common_diff*, return A[mid] + *common_diff* as the missing value.
    - Next, check if A[mid] = A[1] + ((mid - 1) * *common_diff)*.
        - If valid, recursively search the right side of A, otherwise, recursively search the left side of A.
        - This continues until the missing value is returned.

**Correctness:**

- Using a modified binary search, we can find the missing element in A by recursively searching the left and right sides of A.
- We first calculate the common difference using the formula: (A[n] - A[1]) / n. By subtracting A[n] and A[1], which will never be missing, we get the sum of differences between the consecutive elements in that range. Then dividing that result by the number of elements in A, returns the common difference.
- Using the arithmetic progression formula: *a[n] = a[1] + (n – 1) * d*, where *a* is *A*, *n* is *mid*, and *d* is *common_diff*, we can determine if the sequence of elements from A[1] to A[mid] is a valid arithmetic progression.
    - If A[mid] = A[1] + ((mid - 1) * *common_diff*), we know that every element on the left side of A is in the correct arithmetic progression because the

elements are either all increasingly or decreasingly sorted with a constant difference between one other. So, we recursively search the right side of A for the missing element.

    - If A[mid] != A[1] + ((mid - 1) * *common_diff*), we know that every element on the right side of A is in the correct arithmetic progression, so we recursively search the left side of A for the missing element.
- At the beginning of each recursive search, we check for the missing element by validating if the elements to the left and right of A[mid] are equal to A[mid] - *common_diff* and A[mid] + *common_diff* respectively. If either is invalid, we've found the missing element and return it accordingly.

**Runtime:**

- The known runtime for binary search is O(log n).
- Modifying the checks within binary search are done in O(1) time.
- Overall runtime is O(log n). This is confirmed using the recurrence relation of T(n) = T(n/2) + O(1), which results in O(log n) by the Master Theorem where *a = 1, b = 2, and d = 0*.

**References:**

- https://en.wikipedia.org/wiki/Arithmetic_progression

**Collaborators:**

Daniel Smith (Dsmith628@gatech.edu) , Michael Chen (mchen493@gatech.edu), Humberto Evans (hevans39@gatech.edu), Jordan Chen (jchen60@gatech.edu), Ryan Wade Robinson (rrobinson79@gatech.edu), Andrew Gingrich (agingrich3@gatech.edu), Jonathan Greene (jgreene82@gatech.edu), Miranda Riggs (mriggs30@gatech.edu), Stanley Kwok (skwok30@gatech.edu), Christopher Vance (cvance@gatech.edu), Lijun Liu (gtg884x@gatech.edu), Matthew Thomas (lthomas97@gatech.edu), Mason Munro Costa (mcosta31@gatech.edu), Connor Tibedo (ctibedo3@gatech.edu)

Wagoner, Julianne (jwagoner6@gatech.edu), Diallo, Ammar (adiallo39@gatech.edu), Dassanayake, Aravinda B (adassanayake3@gatech.edu), Fung, Lokwai (lfung7@gatech.edu), Shah, Krushang A (krushang.shah@gatech.edu), Walsh, Joshua B (jwalsh65@gatech.edu), Mac'Kie, Ann (amackie3@gatech.edu), Whaley, Ethan G (ewhaley8@gatech.edu), Borger, Alexander Q (aborger3@gatech.edu), Li, Xin (andy.li@gatech.edu)