Question #2 : Proof of Correctness of Huffman Encoding Problem

Allen Kim, Yeshuchan (Jack), Gautam Ramasubramanian

October 19, 2016

Problem Statement

For a square $N \times N$ matrix A, assume the elements are sorted in ascending order along the horizontal and vertical directions already, i.e., $A[i][k] \leq A[j][k]$ and $A[k][i] \leq A[k][j]$, where $i < j$. Develop an efficient algorithm to search for the query value v from A, return the location if found, None otherwise. Analyze the time complexity of your algorithm.

Algorithm Description

To find v in a given a sorted 2D matrix, we first start from the top right corner $A[0][N-1]$ and check if it is equal to v. If it is, then we are done. Else, we check if $v < A[0][N-1]$. If it is, we check the element to the left. If $v > A[0][N-1]$, then we check the element below it. We then iterate this process, always moving left to check smaller elements or moving right to check larger elements.

Algorithm Pseudocode

```
bool find(int Matrix[N][N], int v){
    int r = 0;
    int c = N-1;
    while (r < N && c >= 0){
        if (v == Matrix[r][c])
        return true;
        else if (v < Matrix[r][c])
            c--;
        else if (v > Matrix[r][c])
            r++;
    }
    return false;
}
```

Algorithm Time Complexity

Since the number of rows and columns are both equal to N, the time complexity is $O(N)$. This time complexity holds even for the worst case as the algorithm will only traverse at most $2(N-1)$ elements ($N-1$ "downs" and $N-1$ "lefts").