Homework 3: Q2

Name: Xinkai Lin, xinkaili

*Don't forget to input your list of collaborators and sources on **AutoLab**.*
Please submit this file as a PDF.

1 Algorithm Idea

Begin Algorithm Idea: Since our matrix A is guaranteed to have some structure, we can see that given a vector r with length of n we can compute the matrix as

$$U^{(r_1, r_2, r_3)} = \begin{pmatrix} r_1 & r_1 & r_1 \\ 0 & r_2 & r_2 \\ 0 & 0 & r_3 \end{pmatrix}$$

And given vector x = {x_1, x_2, x_3}.

To find the y, we have to use row-column operation which add all the sum of row of matrix times vector x. Such as $r_1*x_1+r_1*x_2+r_1*x_3$. Also, we can rewrite it as $r_1*(x_1+x_2+x_3)$. Since it is a special matrix, we can see that as the row increase, the number of element on row decrease.

For example:

Row1: $r_1*(x_1+x_2+x_3)$
//since the special matrix, the first element on second row just going to be 0, $0*x_1 = 0$.
Row2: $r_2*(x_2+x_3)$
//since the special matrix, the first two elements on third row just going to be 0.
Row3: r_3*x_3

From a different perspective, we can see that the numbers on the row is increasing by one from bottom to top. So, my algorithm started from bottom to top. Creating a temp variable to save the value of x as we follow along with the algorithm. Such that loop1, temp = x_3. Loop2, temp = (x_2+x_3), etc. Created the for loop from n-1 to 0. Then, multiply temp by vector r. Since it is from backward, so first loop will be temp * r_3, which temp is the value of x_3, then add the product to the list. Second loop will be temp*r_2, which temp is the value of (x_2+x_3), then add the product to the list. When the loop completed, we will get a list of y in reverse order. Finally, use reverse function to fix it. The operation inside the loop will takes runtimes of O(1). And since we will loop through all the elements at least once, and so the runtimes for the for loop is O(n). Also, the runtimes for the reverse will takes O(n) as well. So, the runtimes for this algorithm is O(n)+O(n), still O(n) time. And we can use this algorithm to solve any two vectors r and x of length n correctly computes $U^r \cdot x$

2 Algorithm Details

Algorithm: Algorithm1

```
n;              //length of vectors
temp = 0;       //temp variable of int type, initially 0
rList;          //Array of ints, and stores the vector r's values
xList;          //Array of ints, and stores the vector x's values
result;         //Array of ints, will store the y values
y;              //store each y value compute from for loop

//loop from backward
for (int i=n-1; i>=0; i--) {
        temp += xList[i];          //store vector x's values from backward
        y = temp * rList[i];       //each y value compute from U^r • x
        result.add(y);        //add each y value to result (ArrayList in Java)
}
reverse(result);                   //reverse the order of the list
return result;
```

3 Big-Oh Analysis

Begin Big-Oh analysis: In this algorithm, I have one for loop and one reverse function. In the for loop, each of the operation will run at O(1) time. So, when the loop finished it will runs O(n), which is loop all the element at once. The reverse method will take O(n) time since it will loop through all the value and reverse them. So the runtimes of this algorithm will be O(n)+O(n), which is still O(n) time.

4 Big-Omega Analysis

Begin Big-Omega Analysis: Each operations inside the for loop will run at least once, which is $\Omega(1)$, and since for loop will loop through all the element, therefore, the lower bound of the runtime in worst case is $\Omega(n)$. Moreover, since the upper bound O(n) and the lower bound $\Omega(n)$ are equal, we can say that the runtime of this algorithm is $\theta(n)$.