1 Problem 1

In this problem, we choose the comparsion as the relevant operation to count. Under the worst-case scenario, we needs to traverse the entire array to conclude if the double array contains the number we are looking for. Therefore, for an array with n elements, we have

$$T(n) = n$$

as number of operations for double array with given length.

We conclude $T(n) \in O(n)$ because the inequality $\mid f(n) \leq c \mid g(n) \mid\mid$ holds when $c = 1$ and $n_0 = 0$, since $f(n) \leq O(n)$ for $n \geq 0$.

2 Problem 2

We find the pattern of the fastModExp method as:

$$x^y = \begin{cases} 1 \bmod m & x = 0 \\ (x^2 \bmod m)^{\frac{y}{2}} \bmod m & \text{when y is even} \\ (x * (y^{(y-1)} \bmod m)) \bmod m & \text{when y is odd} \end{cases}$$

Therefore, we define all arithmetic operation as relevant operation and assume all of them is $O(1)$ operation. Then, we obtain the following recurrence for fastModExpt:

$$T(y) = T([y/2]) + 2$$

where we assume the input y is a power of two. And this recurrence implies the closed-form solution $T(y) = 2 * O(\log y) + 3$ if we draw the recursion tree.

We conclude $T(y) \in O(\log n)$. To see this, we node that, for $n \geq 2$, we have $T(n) \leq 3 * \log_2 n$. Therefore we have $c = 3$ and $n_0 = 2$.

3 Problem 3

In this problem, we consider array access as the relevant operation. Then we can obtain the model

$$T(n) = 3n^2$$

where n is the number of element of the input array.

Then we can conclude that $T(n) \in O(n^2)$. To see this, note that for $n \geq 0$, $T(n) \leq 3 * n^2$. Therefore we have $c = 3$ and $n_0 = 0$.

4 Problem 4

4.1 Constant-time String Concatenation

In this case, we define string concatenation as the relavent operation. We can write a model T of the time coplexity as

$$T(n, m) = nm$$

where n is the n is the numbers of repetition and m is the number of strings in the input array. We conclude $T(n, m) \in O(n^2)$ because the program contains two loops and number of loops decides by both n and m.

4.2 Linear-time String Concatenation

For the sake of the simplity, we assume all the string have the same length l and we believe such simplication will not influence the result of analysis. Then, we can translate the loop bounds to summations bounds:

$$\sum_{i=1}^{mn} il$$

and we can transform above form into

$$T(m, n) = \frac{mn(mnl + l)}{2}$$

Therefore, we conclude that $T(m, n) \in O(n^4)$ if we expand the formula above. The run-time of the program change from $O(n^2)$ to $O(n^4)$ because the cost of concatenation changes from $O(1)$ to $O(n)$.

5 Problem 5

5.1 Time Complexity

For the analysis of time complexity, we decide to choose array access as relevant operation and then we have:

$$T(n, m) = m + n$$

where n and m is the number of elements in each of the input arrays. Then we can conclude $T(n, m) \in O(n)$ because the number of array access increase linearly as we increase the length (number of elements) of input array.

5.2 Space Complexity

The space complexity of the function can be described by:

$$T(m, n) = m + n$$

where n and m is the number of elements in each of the input arrays. And we conclude that $T(n, m) \in O(n)$ for space complexity the length of the new array we produce is the sum of two input array.

5.3 Review

We found that there is not any connection between space complexity and time complexity in genreal. However, under certain scenario, we can establish a connection between them. For example, when the critical operation cost both extra time and space to perform.