CS161 SU24: Homework 1 (Due July 1 11:59pm)

❖ Problem 1

Reorder the following functions such that if f appears before g, then $f = O(g)$. In other words, reorder the functions in increasing order with respect to their rate of growth. Here, we are using the convention of $\lg n := \log_2 n$, and $\log n := \log_{10} n$.

A: $4n$	B: $5 \lg n$	C: $5 \lg \lg n$	D: $\log n$
E: n^3	F: $(\log n)^{\log n}$	G: 4^{4^n}	H: 4^{n^4}
I: $n^{(1/2)\lg^2 n}$	J: $\log(n^{\log n})$	K: $n^{n^{(1/4)}}$	L: $n^{(n/4)}$
M: $4n^{5 \lg n}$	N: $n \lg n$	O: $5^{\log n}$	P: $(n/4)^{(n/4)}$

Use inequalities to order your answers, and if two functions are asymptotically the same, use an equals sign. As a starting point and demonstration, we have

$$B = D < A \tag{1}$$

Furthermore, show the asymptotic relation between the following pairs of function, and justify your answer by stating appropriate values for c and n_0.

1. B and N
2. F and J
3. C and D

❖ Problem 2

Look at the following function written in Python. Suppose that each line takes 1ms to run.

```python
def complex_function(arr):
    n = len(arr)
    total_sum = 0

    for i in range(n):
        for j in range(i, n):
            temp_sum = 0
            for k in range(i, j + 1):
                temp_sum += arr[k]
            total_sum += temp_sum

    return total_sum
```

1. Write down the time the function will take to run in ms as a function of n.
2. Express your answer from above in big-O notation.

❖ Problem 3

Emmet is a herpetologist (person who studies reptiles and amphibians). On a given day, he takes n special frogs and separates them into $\sqrt{n}$ separate tanks in the following way:

- Every frog is a different size, ranging from 1 to n cm. The size of each frog is an integer in these units (Yes, some of these frogs are huge!).
- Each tank is placed in a line from left to right.
- In tank i, Emmet places frogs ranging in size from $i\sqrt{n} + 1$ to $(i + 1)\sqrt{n}$ where the numbering of the tanks starts from 0 and goes up to $\sqrt{n} - 1$. (This also tells us that every frog in a tank is smaller than all frogs in tanks that come to the right of it).
- You may assume that n is a perfect square.

We want to find Frankie the frog from these tanks. We don't know how big Frankie is, and Emmet will only answer the following two questions:

- Is Frankie bigger than x?
- Can you take out a random frog from tank i?

We know what Frankie the frog looks like, so as soon as Emmet takes her out of a tank, we will know that we have found her.

1. In English, describe a strategy to **find the tank** Frankie lives in using only $O(\log n)$ questions.
2. In the **worst case**, how many questions do we need to ask Emmet?
3. Assuming that Emmet's selection is uniformly random (every frog in a tank has the same probability of being chosen), what is the **expected value** or **average** number of questions we have to ask him?
4. (Bonus) Draw Frankie the frog.

❖ Problem 4

Consider the following sorting algorithm. In the beginning, search for the smallest element in `arr[0:n]` and swap that item with `arr[0]`. Next, search for the smallest element in `arr[1:n]` and swap that item with `arr[1]`. In the i-th iteration, search for the smallest element in `arr[i-1:n]` and swap that item with `arr[i-1]`.

1. The above algorithm is called **selection sort**. Implement the above algorithm in Python and paste your code here.
2. State the **loop invariant** that this algorithm maintains. Use this to prove that the algorithm is **correct** in the two ways we discussed in class.
3. State the worst case running time of this algorithm in big-O notation, and briefly explain where that running time comes from.
4. What is the best case running time for this algorithm? Is it any better than the worst case running time?

❖ Leetcode

Here is a sample of some easy-medium Leetcode problems that you should be able to solve and/or understand the solutions to. As mentioned at the beginning of class, set a timer to try solving these on your own, then once the timer is up check a solution and try to understand why that works. For any solution you write, make a habit of stating what your running time is, and check to see if anyone else has a solution that has a better running time.

- Search insert position [Easy]

  https://leetcode.com/problems/search-insert-position/description/

- Search in rotated sorted array [Medium]

  https://leetcode.com/problems/search-in-rotated-sorted-array/description/

- Merge two sorted lists [Easy]

  https://leetcode.com/problems/merge-two-sorted-lists/description/

- Permutations [Medium]

  https://leetcode.com/problems/permutations/description/

  - You may want to use a recursive algorithm.

- Insertion sort list [Medium]

  https://leetcode.com/problems/insertion-sort-list/description/

- Most beautiful item for each query [Medium]

  https://leetcode.com/problems/most-beautiful-item-for-each-query/description/