Problem 1

Rank the following functions by order of growth; that is, find an arrangement \( g_1(n), g_2(n), \ldots, g_{24}(n) \) of functions satisfying \( g_i(n) = O(g_{i+1}(n)) \) for every \( i \in \{1, \ldots, 23\} \). Partition your list into equivalence classes such that \( f(n) \) and \( g(n) \) are in the same class if and only if \( f(n) = \Theta(g(n)) \). You do not have to prove your answers.

\[
\begin{array}{cccccccc}
 n \log n & n^{4/3} & n^{3/4} & n^{1/\log n} & \log n & 1 & (\log n)^{\log n} & 2^n^2 \\
 \log_8 n & 8^{\log n} & n & 2^n & 2^{n+1} & \log \log n & n^{\log \log n} & n! \\
 2^{2n} & 2^{2n} & \log^2 n & \log(n^2) & \sqrt{2^{\log n}} & \sqrt{\log n} & n2^n & n + n^3/10^30
\end{array}
\]

Remarks:

- In this class we use \( \log n \) to denote the logarithm base 2.
- Use the Stirling’s formula to figure out how to rank \( n! \). The Stirling’s formula is:
  \[
  n! = \sqrt{2\pi n} \left( \frac{n}{e} \right)^n \left( 1 + O\left( \frac{1}{n} \right) \right)
  \]
- Use also this fact: for any constants \( b_1, b_2 > 0 \):
  \[
  \log^{b_1} n = O(n^{b_2}) \quad \text{and} \quad n^{b_2} \neq O(\log^{b_1} n)
  \]

In words, logarithm of \( n \) raised to any power grows slower than any power of \( n \).

Problem 2

This problem is about a puzzle with strings. We are given two strings \( u \) and \( v \), each of length \( n \) and using only lower-case letters. We want to modify \( u \) into \( v \). The tricky part is that we are allowed to use only this operation: take a contiguous segment of the current string and either

- shift each letter in the segment forward in the alphabet (‘a’ becomes ‘b’, ‘b’ becomes ‘c’, and so on; the segment cannot contain letter ‘z’), or
- shift each letter in the segment backward in the alphabet (‘b’ becomes ‘a’, ‘c’ becomes ‘b’, and so on; the segment cannot contain letter ‘a’).

What is the minimum number of such operations we need to use to modify \( u \) into \( v \)?

**Example 1:** Suppose we want to modify \( u = \text{“hello”} \) into \( v = \text{“teams”} \). There are several possible ways how modify \( u \) into \( v \) using 27 operations. For example, we can first shift “lo” forward, getting “helmp”. Then shift “h” forward 12 times, getting “tehmp”. Then shift “l” 11 times backward to get “teamp” and then shift “p” forward three times to get “teams”. Total number of operations is \( 1+12+11+3=27 \). It turns out that 27 is the smallest possible number of operations.
Example 2: Suppose we want to modify $u = \text{"aacccaaaa"}$ into $v = \text{"bbbbbbbbb"}$. There are several possible ways to modify $u$ into $v$ using 3 operations. For example, we can first shift the entire string forward, getting “bbdddbbbb”. Then shift “ddd” backward twice to get “bbbbbbbbb”. This requires $1+2=3$ operations.

Your task: Design and implement an $O(n)$ algorithm that computes the minimum number of operations needed to modify $u$ into $v$.

Note: Polynomial-time algorithms slower than $O(n)$ are accepted for partial credit. In such case, instead of arguing that your algorithm is $O(n)$, estimate your algorithm’s running time.

Problem 3

Consider the following “shopping list” problem: given is a shopping list of $n$ items and for each item you know its location in the store (that is, for the $i$-th item, you are given its coordinates $(x_i, y_i)$). You are starting at coordinates $(0,0)$, where the entrance to the store and also the cashier stations are located. You want to buy all items on your list and minimize time spent in the store. For simplicity, assume that the time you need to move from one location to another is simply the Euclidean distance between the points (converted to seconds). Also assume that for each item it takes 5 seconds to put it into your shopping cart.

Now consider the following (greedy) algorithm for this problem:

1. Find the item on the current shopping list that is the closest to your current location, go to it and put it in your cart. Cross out the item from your shopping list.

2. If the current shopping list is nonempty, repeat step 1. Otherwise, go to a cashier station and leave the store.

It is tempting to say that the algorithm works but, sadly, there are inputs for which it does not produce an optimal solution (that is, it collects the items in suboptimal time). Your task is to:

a) Give an input on which the algorithm fails. Sketch the location of the items on your input and draw the route taken by the above algorithm. Then sketch an optimal route that is shorter than the algorithm’s route.

b) Estimate the running time of the pseudo code using big-Oh notation. Make your estimate as accurate as possible (for example, if the running time is $O(n)$, state $O(n\log n)$, not $O(n^2)$).

Include a paragraph that reasons the estimate.

Problem 4

Given is a large paper with $n$ different points with coordinates $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Notice that by folding the paper along a single line we can make some of the points align. For example, if the points are $(1,2), (2,1)$, and $(4,3)$, then if we fold along the line going through the origin at the 45 degree angle, the points $(1,2)$ and $(2,1)$ will align. Design an $O(n^2 \log n)$ algorithm that finds the maximum number of pairs of points that can be aligned.