1. **Asymptotic Notation**

   For each list of functions, order them according to increasing asymptotic growth. Provide a brief argument justifying each successive step in the ordering.

   List 1, fast growing functions: $2^{3n}, 3^{2n}, n!, n^{\log^* n}, n^{n\sqrt{\log n}}$.

   List 2, slow growing functions: $2^{\log^* n}, \log^*(2^{2n}), 2^\sqrt{\log n}, \log \log \sqrt{n}, \log(n^5)$

   Recall that $\log^*(n)$ (the “log star” function) calculates how many times you would need to take the iterated $\log_2$ of $n$ before you would go below 2. Formally, $\log^*(x) = 0$ for all $x \in (0, 2)$, and for all $x \geq 2$, $\log^*(x) = 1 + \log^*(\log_2(x))$. Thus for example $\log^*(2) = 1, \log^*(2^2) = 2, \text{ and } \log^*(2^{2^2}) = 3$.

2. **Asymptotic Estimates**

   In this problem, your solutions should begin with a displayed equation of the following form:

   \[ \text{[expression to be analyzed]} = \Theta(f(n)), \]

   where $f(n)$ is a “simple” function. For example, the expression for the Harmonic number $\sum_{i=1}^{n} (1/i) = \Theta(\log(n))$.

   You should then give an explanation of how you derived your answer (but a completely formal proof is not required).
(a) Asymptotically simplify \( \sqrt{n+1} - \sqrt{n} \)

(b) Asymptotically simplify 
\[
\sum_{i=1}^{n} i^3
\]

(Hint: Try relating the discrete sum to an integral.)

(c) Asymptotically simplify 
\[
\sum_{i=n+1}^{n^2} \frac{1}{i}
\]

(d) Assume \( m \geq 1 \) is the solution of \( m^m = n \). Asymptotically express \( m \) as a function of \( n \).

3. **Recurrences**

Solve the following recurrences, giving your answer in \( \Theta \) notation. For each of them, assume the base case \( T(x) = 1 \) for \( x \leq 5 \). Show your work.

a) \( T(n) = 3T(n/4) + n \)

b) \( T(n) = T(n-2) + n^4 \)

c) \( T(n) = 2T(n-5) \)

d) \( T(n) = \sqrt{n}T(\sqrt{n}) + n \)

4. **Fast Multiplication**

Given two \( n \)-bit strings, you are asked to output their product in the form of binary string. For instance, if you received 101 and 11 as inputs, you should output 1111. The time complexity of the algorithm in this problem is the number of basic bitwise operations such as AND, OR, XOR. You may also assume that you have access to primitive operations of addition and subtraction of \( n \)-bit strings which use \( O(n) \) basic bitwise operations.

a) Show the naive algorithm we learnt from elementary school (potentially kindergarten) with time complexity \( O(n^2) \);

b) Show a faster algorithm with time complexity \( o(n^2) \) and state the time complexity in \( O \) notation.

Hint 1: Use the same trick as we did in class to solve matrix multiplication recursively.

Hint 2: Think about calculating 11 times 11. Using naive multiplication, this can be done with 4 multiplications. Can you do it with just 3?

5. **Linear Algebra**
There are at least two ways that one can be given a subspace $W$ in $\mathbb{R}^d$. The first is by a set of generators: We say that the vectors $P_1, \ldots, P_k \in \mathbb{R}^d$ are generators for the subspace

$$W = \{ \alpha_1 P_1 + \cdots + \alpha_k P_k \mid \alpha_i \in \mathbb{R} \}$$

The second as a set of constraints: Let $A \in \mathbb{R}^{n \times d}$ be matrix. We say that $A$ is a constraint matrix for the space $W$ if

$$W = \{ x \in \mathbb{R}^d \mid Ax = 0 \}$$

1. Given the subspace $W$ by a set of generators $P_1, \ldots, P_k \in \mathbb{R}^d$ show how to write $W$ via a constraint matrix $A$.

2. Given the subspace $W$ by a constraint matrix $A$ show how to write $W$ by a set of generators.

(10) 6. Greedy Algorithms

You are at a conference and would like to attend as many talks as possible. There are $n$ talks given and each has a starting time $s_i$ and ending time $e_i$. If you attend a talk you must sit-in through the entire talk (from start to finish) and you can only attend one talk at a time (i.e. you can’t attend a pair of talks if their intervals intersect). You are interested in finding the maximum amount of talks you can attend.

a) Find an incorrect greedy algorithm that tries to solve this problem and give a counterexample explaining why it fails.

b) Find a correct greedy algorithm that solves this problem and give an argument why it always works.