

Math Problem 1

(1) $f(x) = -x^2$

Taking derivative of $f(x)$, we get

$$f'(x) = -2x$$

For $x < 0$, $f'(x)$ is negative

$x > 0$, $f'(x)$ is ~~negative~~ positive

Hence, $f(x)$ is decreasing for x . It is not eventually non decreasing because for all $x > 0$, it keeps decreasing and never is stable.

(2) $f(x) = x^2 + 2x + 1$

Taking derivate, $f'(x) = 2x + 2$

$$\text{For } x < -1 \Rightarrow f'(x) < 0$$

$$x > -1 \Rightarrow f'(x) \geq 0$$

It ~~stop decreasing~~ is not increasing, but eventually non-decreasing

(3) $f(x) = x^3 + x$

$$f'(x) = 3x^2 + 1$$

For $-\infty < x < \infty$, $f'(x)$ is positive

Hence, it is increasing everywhere, also eventually increasing

Math Problem 2

$$(1) f(x) = 2x^2, g(x) = x^2 + 1$$

$$\Rightarrow O(x^2) \quad \Rightarrow O(x^2)$$

They are asymptotically equal.

$$(2) f(x) = x^2, g(x) = x^3$$

Hence, x^3 grows faster than x^2 ,

$$(3) f(x) = 4x + 1, g(x) = x^2 - 1$$

x^2 grows faster than $4x$

Hence, $f(x)$ grows no faster than $g(x)$, not vice versa.

```

public class GCDAAlgorithm {
    1 usage
    public static int findGCD(int m, int n) {
        while (n != 0) {
            int temp = n;
            n = m % n;
            m = temp;
        }
        return m;
    }
    public static void main(String[] args) {
        System.out.println("GCD of 12 and 8: " + findGCD(m: 12, n: 8));
    }
}

```

Problem 2:

Brute Force Solution:

```

no usages
public class SubsetProblem {
    no usages
    public static List<Integer> subsetSum(int[] set, int k) {
        return subset(set, i: 0, k, new ArrayList<>());
    }

    3 usages
    private static List<Integer> subset(int[] set, int i, int k, List<Integer> subset) {
        if (k == 0) return new ArrayList<>(subset);
        if (i == set.length || k < 0) return null;
        subset.add(set[i]);
        List<Integer> include = subset(set, i: i + 1, k: k - set[i], subset);
        if (include != null) return include;
        subset.remove(index: subset.size() - 1);
        return subset(set, i: i + 1, k, subset);
    }
}

```

Problem 3: Greedy Strategy

Consider the sorted set of numbers: {2, 3, 7, 8, 10}, and the target sum is 11.

- **Greedy Output:** {2, 3} (Total = 5, which is incorrect!)
- **Correct Subset:** {3, 8} (since $3 + 8 = 11$)

This shows that the greedy strategy doesn't always work. Here's why:

- The algorithm starts by picking the smallest numbers, which can sometimes lead to missing better combinations.
- In this case, picking 2 and 3 first blocked us from considering 7 or 8, which would have added up to the exact sum of 11.

Problem 4:

It really depends on the situation. If the subset T minus the smallest number (s_{n-1}) still adds up to the target sum (k'), then removing s_{n-1} can still lead to a valid solution. However, sometimes s_{n-1} is crucial to reaching the target sum, and removing it could break the solution. So, removing s_{n-1} from the set T doesn't guarantee we still have a valid solution.