Data Structure and Algorithms

Written Assignment 1 （2024）

Problem 1. (15 marks)

How many basic operations does the following code consume in total? Please give an exact answer and list your calculation steps.

```
public static int function(int n){
        if (n == 1 || n == 2) {
                return 1;
        }
        if (n > 2) {
                return function(n - 1) + 1;
        }
        return -1;
}
```

Problem 2. (20 marks)

For each pair of f(n) and g(n) below, determine whether f(n) = O(g(n)), f(n) = Ω(g(n)), or f(n) = Θ(g(n)). Justify your answer using the definitions of these asymptotic notations. Note that for a given pair, more than one of these relations may hold; list all that hold.

(a) $f(n) = \sqrt{n} + \sin(n)$ and $g(n) = \sqrt{n} + n$.

(b) $f(n) = 2n$ and $g(n) = \log^k n$ (k is a constant).

(c) $f(n) = 4n^2+3n+2$ and $g(n) = 2n^2+3n+4$.

Problem 3. (20 marks)

Let *f(n) and g(n)* be two asymptotically positive functions, respectively. Prove or disprove each of the following conjectures.
Hint: You can prove a conjecture using its definition or disprove a conjecture by giving negative examples.

(a) f(n) + g(n) = Θ (2f(n) + 2g(n)).

(b) f(n)* g(n) = Θ (f(2n)*g(2n)).

Problem 4. (15 marks)

Solve the following recurrence relation and represent *T(n)* using a formula of *n*.

$$T(n) = \begin{cases} 3T(\frac{n}{3}) + \mathrm{n}, & n > 1 \\ 1, & n = 1 \end{cases}$$

Problem 5. (15 marks)

In the merge sort algorithm, we divide an array into two halves, recursively sort the subarrays, and then merge them into a sorted array. Now Tim proposes a "merge sort pro" algorithm. In "merge sort pro", an array is divided unevenly into two subarrays and the ratio of data to be sorted is a:b, and the rest of the steps are similar to those of merge sort. What do you think is the time cost of "merge sort pro" if the input size is *n*? Prove your answer.

Problem 6. (15 marks)

Given two integer structure arrays $\mathrm{items1}$ and $\mathrm{items2}$, representing two sets of items. The number of items are n and m, respectively. Each array item has the following properties: $\mathrm{items[i]} = (\mathrm{valuei}, \mathrm{weighti})$ where valuei represents the value of the i-th item and $\mathrm{weighti}$ represents the weight of the i-th item. The value of each item in items is unique. It is expected to return a structure array ret, where $\mathrm{ret[i]} = (\mathrm{valuei}, \mathrm{weighti})$, $\mathrm{weighti}$ is the sum of the weights of all items with valuei. Please

1) describe your algorithm in *pseudo-code*.
2) give the big-O of your algorithm.

Note: ret should be returned sorted in ascending order by value.

For example, input: $\mathrm{items1} = \{(1,1),(4,5),(3,8)\}$, $\mathrm{items2} = \{(3,1),(1,5)\}$,
output: $\{(1,6),(3,9),(4,5)\}$.

Explanation: The item with $\mathrm{value} = 1$ has $\mathrm{weight} = 1$ in $\mathrm{items1}$ and $\mathrm{weight} = 5$ in $\mathrm{items2}$, so the total weight is $1 + 5 = 6$. The item with $\mathrm{value} = 3$ has $\mathrm{weight} = 8$ in $\mathrm{items1}$ and $\mathrm{weight} = 1$ in $\mathrm{items2}$, so the total weight is $8 + 1 = 9$. The item with $\mathrm{value} = 4$ has $\mathrm{weight} = 5$ in $\mathrm{items1}$, so the total weight is 5. So, we have $\{(1,6),(3,9),(4,5)\}$.