Written Assignment 1 Sample Answer

Problem 1. (15 marks)

Answer:

Code	Cost
if (n == 1 || n == 2) { return 1; }	Segment 1: 4
if (n > 2) {	Segment 2: 1
return function(n - 1) + 1;	T(n-1)+1+1+1
return -1;	Segment 3: 1

①when n<1, T(n)=5
②when n=1, T(n)=3
③when n=2, T(n)=4
④when n>2, T(n)=4+T(n-1)+1+1+1= T(n-1)+1*7=T(n-i)+i*7, suppose i=n-1,T(n)=T(n-(n-1))+(n-1)*7=7n-6

Problem 2. (20 marks)

Answer:

(a) $f(n) = O(g(n))$
Let $c = 1, n_0 = 2$. If $n > n_0, c \cdot g(n) - f(n) = \sqrt{n} + n - \sqrt{n} - sin(n) = n - sin(n) > 0$. Hence $f(n) = O(g(n))$.

(b) $f(n) = \Omega(g(n))$
If k=1, let c=1, $n_0 = 2$. If $n > n_0, f(n) - c \cdot g(n) = 2n - \ log^k n > 0$. Hence $f(n) = \Omega(g(n))$.

(c) $f(n) = \theta(g(n))$
Let $c = 2, n_0 = 1$.If $n > n_0, c \cdot g(n) - f(n) = 4n^2 + 6n + 8 - 4n^2 - 3n - 2 = 3n + 6 > 0$. Hence $f(n) = O(g(n))$.

Let $c = 1, n_0 = 2$.If $n > n_0, f(n) - c \cdot g(n) = 4n^2 + 3n + 2 - 2n^2 - 3n - 4 = 2n^2 - 2 > 0$. Hence $f(n) = \Omega(g(n))$.

Therefore, $f(n) = \theta(g(n))$.

Problem 3. (20 marks)

Answer:
(a) True.
2f(n) + 2g(n) = 2(f(n) + g(n)), f(n) + g(n) = 1/2(2f(n) + 2g(n)). According to the definition of Θ,

there exist constants c1, n1 and c2, n2 that satisfy c1 · (2f(n1)+2g(n2))≤f(n) · g(n)≤c2 · (2f(n2)+2g(n2))

(b) False.
Suppose f(n)=g(n)=2^n, then f(n) · g(n)=2^n · 2^n=2^2n, f(2n) · g(2n)=2^2n · 2^2n=2^4n.

Problem 4. (15 marks)

Answer:

$$\begin{aligned} T(n) &= 3T\left(\frac{n}{3}\right) + \mathrm{n} \\ &= 3\left[3T\left(\frac{n}{3^2}\right) + \frac{n}{3}\right] + \mathrm{n} \\ &= 3^2 T\left(\frac{n}{3^2}\right) + \mathrm{n} + \mathrm{n} \\ &= 3^2\left[3T\left(\frac{n}{3^3}\right) + \frac{n}{3^2}\right] + \mathrm{n} + \mathrm{n} \\ &= 3^3 T\left(\frac{n}{3^3}\right) + n + n + \mathrm{n} \\ &= \cdots \\ &= 3^k T\left(\frac{n}{3^k}\right) + kn \end{aligned}$$

Let $n = 3^k$:

$$\begin{aligned} T(n) &= nT\left(\frac{n}{n}\right) + n\log_3 n \\ &= n + n\log_3 n \end{aligned}$$

Problem 5. (15 marks)

Answer:

$$T(n) = T\left(\frac{a}{a+b} \cdot n\right) + T\left(\frac{b}{a+b} \cdot n\right) + O(n)$$

In the first layer, the cost of merging the two subarrays is O(n).
In the second layer, each subarray is further divided in the ratio a:b: $T\left(\frac{a}{a+b} \cdot n\right)$ is divided into $T\left(\frac{a^2}{(a+b)^2} \cdot n\right)$ and $T\left(\frac{a}{(a+b)^2} \cdot n\right)$, $T\left(\frac{b}{a+b} \cdot n\right)$ is divided into $T\left(\frac{ab}{(a+b)^2} \cdot n\right)$and $T\left(\frac{b^2}{(a+b)^2} \cdot n\right)$. The merge cost for this level remains O(n).
Every level has a total merge cost of O(n), and the problem size reduces with each split.
So, The merge cost per level is O(n), and the number of levels is approximately O(logn). Therefore, the total time complexity is: T(n)=O(nlogn)

Problem 6. (15 marks)

1)
Using MergeSort algorithm to sort values, and an additional array C is used to store the weight associated with each value after sorting values.

mergeSimilarItems (items1, items2){

```
    Array B,C,ret
    values=CONCATENATE(items1.values, items2.values)
    weights=CONCATENATE(items1.weights, items2.weights)
    MERGESORT(values, left, right)
    FOR i FROM 0 TO B.Length
    ret[i]=Array(B[i],C[i])
    RETURN ret
}

MERGESORT(A, left, right){
  IF left>=right
          RETURN
    center = (left+right) / 2
  MERGESORT(A, left, center)
  MERGESORT(A, center+1, right)
  MERGE(A, left, center, right)
}

MERGE(A, left, center, right)
  i1 = left, i2 = center+1, i=0
  WHILE i1<=center AND i2<=right
          IF A[i1]<A[i2]
    C[i] = weights[i1]
  B[i++] = A[i1++]
    ELSE IF A[i1]==A[i2]
    C[i] = weights[i2]+weights[i1]
    B[i++] = A[i2++]
    i1++
    ELSE
          C[i] = weights[i2]
          B[i++] = A[i2++]
  WHILE i1 <= center
          B[i++] = A[i1++]
  WHILE i2 <= right
          B[i++] = A[i2++]
  Copy B to A[left..right]
```

2) O(nlogn)