**Chapter 3**

**R-3.8 Order the following functions by asymptotic growth rate.**

**4nlog n+2n 2^10 2^log =n**

**3n+100logn 4n 2^n**

**N ^2 +10n n^3 nlogn**

1. ***2^10 (constant)***
2. ***2^log(n) = n***
3. ***3n + 100logn***
4. ***4n***
5. ***nlogn***
6. ***4nlogn + 2n***
7. ***n^2 + 10***
8. ***n^3***
9. ***2^n***

**C-3.35 Assuming it is possible to sort n numbers in O(nlogn) time, show that it is possible to solve the three-way set disjointness problem in O(nlogn) time.**

**#C3.35**

**#assume all the lists are sorted**

**def check(A,B,C,n,m,k):**

**if A[n] == B[m] and B[m] == C[k]:**

**return "COMMON ELEMENT"**

**elif A[n] == B[m] and B[m] < C[k]:**

**if n == len(A)-1 or m == len(B)-1:**

**return "No common element"**

**return check(A,B,C, n+1, m+1, k)**

**elif A[n] == B[m] and B[m] > C[k]:**

**if k == len(C)-1:**

**return "No common element"**

**return check(A,B,C,n,m, k+1)**

**elif A[n] < B[m] and B[m] == C[k]:**

**if n == len(A)-1:**

**return "No common element"**

**return check(A,B,C, n+1, m, k)**

**elif A[n] < B[m] and B[m] < C[k]:**

**if n == len(A)-1 or m == len(B)-1:**

**return "No common element"**

**return check(A,B,C, n+1, m+1, k)**

**elif A[n] < B[m] and B[m] > C[k]:**

**if n == len(A)-1 or k == len(C)-1:**

**return "No common element"**

**return check(A,B,C, n+1, m, k+1)**

**elif A[n] > B[m] and B[m] == C[k]:**

**if m == len(B)-1 or k == len(C)-1:**

**return "No common element"**

**return check(A,B,C, n, m+1, k+1)**

**elif A[n] > B[m] and B[m] < C[k] and A[n] > C[k]:**

**if m == len(B)-1 or k == len(C)-1:**

**return "No common element"**

**return check(A,B,C, n, m+1, k+1)**

**elif A[n] > B[m] and B[m] < C[k] and A[n] < C[k]:**

**if m == len(B)-1 or n == len(A)-1:**

**return "No common element"**

**return check(A,B,C, n+1, m+1, k)**

**elif A[n] > B[m] and B[m] < C[k] and A[n] == C[k]:**

**if m == len(B)-1:**

**return "No common element"**

**return check(A,B,C, n, m+1, k)**

**elif A[n] > B[m] and B[m] > C[k]:**

**if m == len(B)-1 or k == len(C)-1:**

**return "No common element"**

**return check(A,B,C, n, m+1, k+1)**

**if \_\_name\_\_ == '\_\_main\_\_':**

**print(check([1,3,8,9,10,15], [2,4,7,11,13], [5,6,12,14],0,0,0))**

**print(check([1,3,8,9,11,15], [2,4,7,11,13], [5,6,11,14],0,0,0))**

**C-3.42 Bob built a Web site and gave the URL only to his n friends, which he numbered from 1 to n. He told friend number i that he/she can visit the Web site at most i times. Now Bob has a counter, C, keeping track of the total number of visits to the site (but not the identities of who visits). What is the minimum value for C such that Bob can know that one of his friends has visited his/her maximum allowed number of times?**

**Bob has n friends who are allowed to visit at MOST i times. C has to be (n(n-1)/2) +1 to find if at least one person reaches their max.**

**Chapter 4**

**R-4.1 Describe a recursive algorithm for finding the maximum element in a sequence, S, of n elements. What is your running time and space usage?**

**Running time: O(n)**

**Space usage: O(n)**

**import math**

**#send in n as the length of the set**

**def maxRec(S,n):**

**if n == 0:**

**return S[0]**

**else:**

**return max(S[n], maxRec(S,n-1))**

**if \_\_name\_\_ == "\_\_main\_\_":**

**print(maxRec([17,22,34,1,12,16,55,87], 7))#7 is the length**

**R-4.3 Draw the recursion trace for the computation of power(2,18), using the repeated squaring algorithm, as implemented in Code Fragment 4.12.**

**def power(a,b):**

**if b == 0:**

**return 1**

**else:**

**instant = power(a, b//2)**

**finalpower = instant \*\* instant**

**if b%2 == 1:**

**finalpower = a \* finalpower**

**return finalpower**

**print (power(2,18))**

**C-4.9 Write a short recursive Python function that finds the minimum and maximum values in a sequence without using any loops.**

**import math**

**#send in n as the length of the set**

**def maxRec(S,n):**

**if n == 0:**

**return S[0]**

**else:**

**return max(S[n], maxRec(S,n-1))**

**def minRec(S,n):**

**if n == 0:**

**return S[0]**

**else:**

**return min(S[n], minRec(S,n-1))**

**if \_\_name\_\_ == "\_\_main\_\_":**

**print(maxRec([17,22,34,1,12,16,55,87], 7))#7 is the length**

**print(minRec([17,22,34,1,12,16,55,87], 7))#7 is the length**