Tutorial 02

Soumya Dubey

E21CSEU0760

EB27

1. There is a staircase of n steps, and you can climb either 1 or 2 steps at a time. You need to count the total number of unique ways to climb the staircase

Sol. T(n) = way to climb n stairs by taking 1 or 2 or 3 steps

T(1) =1

T(2) =2

T(3) =4

T(4) =7

a. Find the recurrence relation for the number of ways to climb n stairs

Sol. T(4) = T(3) + T(2) + T(1)

T(n-1) + T(n-1) + T(n-3) if n>2

b. How many ways can a person climb a flight of 8 stairs?

Sol. T(5) = T(4) + T(2) + T(1)

= 7 + 4 + 2 + 1

= 14 ways

2. Find the time complexity of the following recurrence relation using backward substitution method.

a. $T(n) = T(n-1) + n, T(1) = 1$

sol. (T((n-1)-1)+(n-1) + n

= T(n-2) + (n-1) + n

T(n-2-1) + (n-2) + (n-1) + n

.

.

.

.

= T(1) + 2 + 3 + ------ + n

= n(n+1)/2

= n^2/2 + 2/2

= O (n^2)

b.$T(n)$ = $T(n-1)$ + 1/n , $T(1)$ = 1

sol. Substituting

T(n-1) = T(n-2) + 1/n-1 in above eq.

T(n) = T(n-2) + 1/n-1 +1/n

T(n) = T(1) + 1/n(n-2) + 1/n-(n-3)+_ _ _ _ _ _ _ + 1/n

T(n)=1+1/2+1/3+_ _ _ _ _ _ _ _ _ +1/n

T(n) = O(log n)

c. $T(n)$ = 8T (n /2) + n^2 , $T(1)$ = 1, n = 2^k

Sol. 8[8T(n/2^2) +n^2/2^2 + n^2

=8^2 T(n/2^2) +2 n^2+n^2

=8^2[8T(n/2^3) +n^2/16]+n^2+n^2

=8^3 T(n/2^3) +8n^2+4n^2+2n^2+2n^2

=8^3 T(n/2^4) + 4 n^2/2^6] + 4n^2 + 2n^2 + n^2

=8^4 T (n^2/2^4) + 8 n^2 + 4 n^2 + 2 n^2 + n^2

=8^4 T (n/2^4) + 2^3n^2+2^2n^2 + 2^1n^2 +2^0n^2

=8^k + (n/2^k)+2^k-1n^2+2^k-2n^2+_ _ _ _ _ _ _ _ _ +2^1n^2+2^0n^2

=(2^k)^3+n^2[2^0+2^1+2^2+_ _ _ _ _ _ _ _ +2^k-1]

=n^3+n^2[1-(2^k -1)/2-1]

=n^3+n^3-n^2

=2n^3-n^2

=O(n^3)

d. $T(n)$ = 2T (n /2) + n/log2 n , $T(1)$ = 1, n = 2^k

Sol. 2[2T(n/2^2) +n log n/2 + n log n

=2^2[2T(n/2^3)+n/2^2 log n/2^2]+ _ _ _ _ _ _ +T(n/2^2)=2T(n/2^3)+n/2^3logn/2^2

=2^3 T (n/2^3) +n log n/2^2+nlogn/2+ n log n

=2^3[2T(n/2^4) +n/2^3logn/2^3]

=2^4T(n/2^4) + n log n/2^3 + n log n/2^2 + n log n/2 + n log n

.

.

.

.

.

=2^k T (n/2^k) + n log n/2^k-1 + n log n/2^k-2 + _ _ _ _ _ _ _ _ _ _ + n log n/2 + n log n

= n/2^k=1

N=2^k

K=log n

n/2^k-1 = n/2^k

=4^n/2^k

= n + n [log n/2^k-1 + log n/2^k-2 + _ _ _ _ _ _ _ _ _ + T(n)

= n + n[log 2 + log 2^2 + log n/2 + log n + log 2^3 + _ _ _ _ _ _ _ + log n/2+logn]

=n + n [log n * log n + log n/2]

=n +n(log n)^2 + n log n

=O n (log n) ^2)

= O (n log^2 n)

3. Find the time complexity of the following recurrence relation using recursion tree method.

a. $T(n)$ = 4T (n/4) + n, T(1) = 1

sol. L0 = n

L1 = n/4 + n/4 + n/4 + n/4 = n

Q3(a) $T(n) = 4T\left(\frac{n}{4}\right) + n, \quad T(1) = 1$

cost L0 $= n$

$\therefore L1 = \frac{n}{4} + \frac{n}{4} + \frac{n}{4} + \frac{n}{4} = n$

$\therefore L2 = \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} \ldots$

Size of sub problem at $L0 = \frac{n}{4^0}$

$1 = \frac{n}{4^k}$

$2 = \frac{n}{4^1}$

$\therefore k = \frac{n}{4^k}$

$\frac{n}{4^x} = 1$

$4^x = n$

$\Rightarrow x \log 4 = \log n$

$\Rightarrow x = \log_4 n$

$= \log_4 n + 1$

(b) $T(n) = T\left(\frac{3n}{7}\right) + T\left(\frac{4n}{7}\right) + n$

$\frac{3n}{7} \quad \frac{4n}{7}$

Total cost $= n$

$\left(\frac{3}{7}\right)^2 n$ and $\left(\frac{4n}{7}\right)^2$

Total cost $\Rightarrow$

$\left(\frac{4}{7}\right)^k n = 1$

$n = \left(\frac{7}{4}\right)^k$

$k = \log_{\frac{7}{4}} n$

Tutorial 3

Soumya Dubey

E21CSEU0760

1) Find the time complexity of the following recurrence relation using Master method.

a. $T(n)$ = T (n/2) + 2^n

a =1

b = 2

k = 0

f(n) = 2^n

n^log a base b = n^ log1 with base 2

n^0 =1

f(n) >n^log a base b

So, T(n) = O(f(n))

= O(2^n)

b. $T(n)$ = 2T (n/4) + √n

a = 2

b = 4

k = 0

f(n) = root n

n^log a with base b = n^log 2 with base 4

= n^0.5

F(n) = n^ log a with base b

T(n) = (n^0.5 * log k+1 * n)

T(n) = (root n log n)

c. $T(n)$ = 100T (n 10) + n√

a = 100

b = 10

f(n) = n root n = n^1.5

n^log 100 with base 10

= n^2

f(n) < n^log a with base b

T(n) = theta (n^2)

d. $T(n)$ = 25T (n/4) + n

a = 25

b = 4

k = 0

f(n) = n

n^log 24 with base 4

n^2.3

Now, f(n) < n^log a with base b

T(n) = theta (n^2.3)

e. $T(n)$ = 13T (n/3) + n^3

a = 13

b =3

k =0

f(n) = n^3

n^log 13 with base 3

n^2.33

f(n) > n^log a with base b

T(n) = O(f(n))

= O(n^3)

2. Find the time complexity of the following recurrence relation.

a. $T(n)$ = T(√n) + log2 n

a = 1

b = 1

k =1

f(n) = log 2n

n^log 1 with base 1

n^0

=1

f(n) < n^log a with base b

$T(n) = \text{theta}\ (1)$

b. $T(n)$ = 8T (n/2) + n^3 (log2 n)^2

a = 8

b = 2

k = 2

f(n) = n^3(log 2 n)^2

n^log 8 with base 2

n^3

f(n) > n^log a with base b

T(n) = O(n^3(log 2 with base n)^2)

3. Find the time complexity of following program

a. T(n/10) for the else part

and n(i) for if part

so, T(n) = T(n/10) + n

b. T(n/4)*log n for the else part

N for the if part

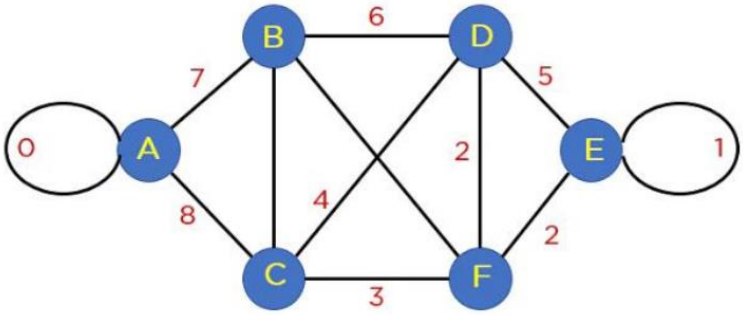
T(n) = T(n/4) * log n + n

TUTORIAL 4

Soumya Dubey

E21CSEU0760

1. kruskal's algorithm is a greedy algorithm as in each step it adds the next lowest-weight edge that will not form a cycle to the minimum spanning forest. It makes a locally optimal choice, intending to find the global optimal solution.
   Use Kruskal's algorithm as greedy approach to determine minimum spanning tree in following graph. Each edge is associated with a cost. Determine cost of MST.



Graph G(V, E)

B C 1

D F 2

E F 2

C F 3

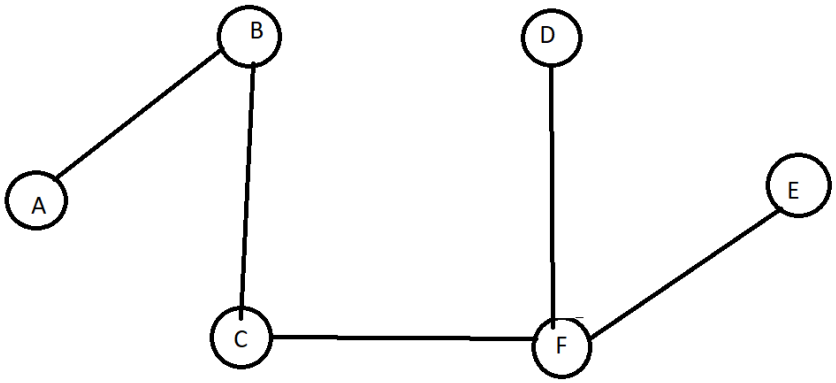
D C 4

D E 5

B D 6

A B 7

F B 9



2. **Greedy approach for job sequencing problem:** *Greedily choose the jobs with maximum profit first, by sorting the jobs in decreasing order of their profit. This would help to maximize the total profit as choosing the job with maximum profit for every time slot will eventually maximize the total profit.* Follow the given steps to solve the problem:

1) Sort all jobs in decreasing order of profit.
2) Iterate on jobs in decreasing order of profit. For each job , do the following :
   - Find a time slot i, such that slot is empty and i < deadline and i is greatest. Put the job in this slot and mark this slot filled.

- If no such i exists, then ignore the job.

Given the jobs, their deadlines and associated profits as shown-

Jobs	J1	J2	J3	J4	J5	J6
Deadlines	5	3	3	2	4	2
Profits	200	180	190	300	120	100

Answer the following questions-

1. Write the optimal schedule that gives maximum profit.
2. Are all the jobs completed in the optimal schedule?
3. What is the maximum earned profit?

Jobs	J4	J1	J3	J2	J5	J6
Deadlines	2	5	3	3	4	2
Profits	300	200	190	180	120	100

180	300	190	120	200

180 + 300 + 190 + 120 + 200 = 990

3. Activity selection using greedy approach: *The greedy choice is to always pick the next activity whose finish time is the least among the remaining activities and the start time is more than or equal to the finish time of the previously selected activity. We can sort the activities according to their finishing time so that we always consider the next activity as the minimum finishing time activity.* Follow are the given steps to solve the problem:

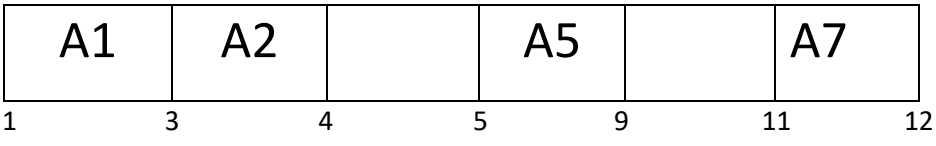
    1) Sort the activities according to their finishing time
    2) Select the first activity from the sorted array and print it
    3) Do the following for the remaining activities in the sorted array
        - If the start time of this activity is greater than or equal to the finish time of the previously selected activity then select this activity and print it

    You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time.
    start[] = {1, 3, 2,0,5,8,11}, finish[] = {3,4,5,7,9,10,12};

Sol. Sort in increasing order

Activity	Start	Finish
A1	1	3
A2	3	4
A3	2	5
A4	0	7
A5	5	9
A6	8	10
A7	11	12

Maximum Activities = 4



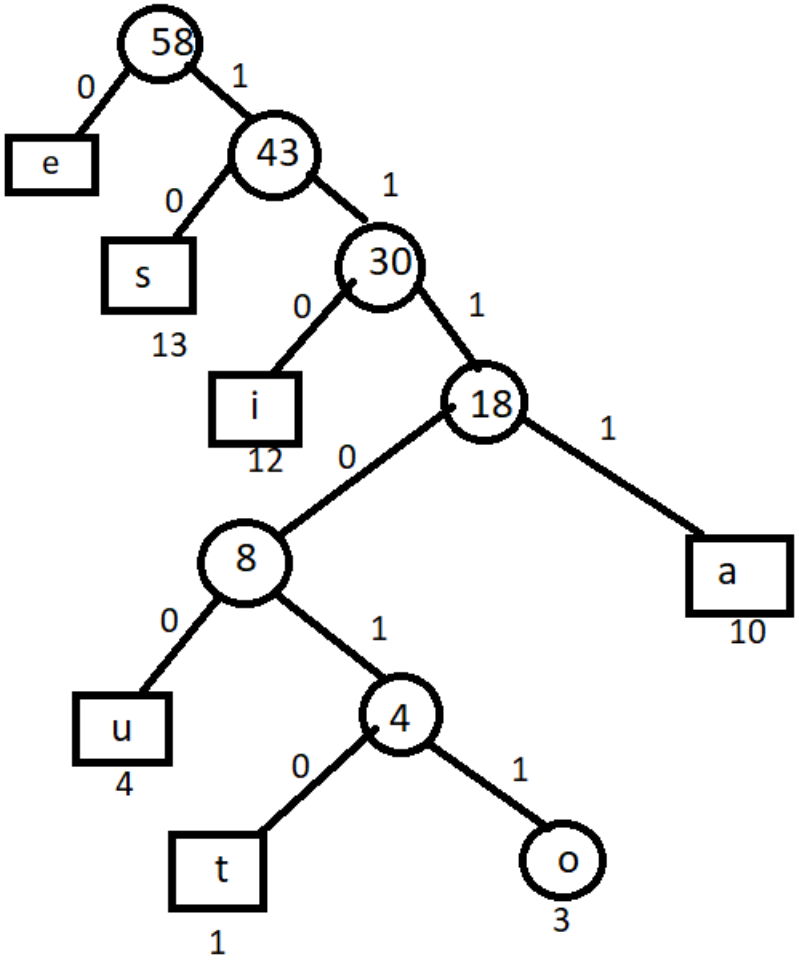
4. *Huffman coding as greedy algorithm:* Huffman Coding is a greedy technique to obtain an optimal solution to a problem. Greedy algorithm is an algorithm that follows the problem solving mechanism of making the locally optimal solution at each stage with thinking of finding a global optimum solution for the problem and In Huffman coding in every stage we try to find the prefix free binary code and try to minimize expected code word for optimum solution for compress the data.

   Steps to build Huffman Tree
   Input is an array of unique characters along with their frequency of occurrences and output is Huffman Tree.

1. Create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. The value of frequency field is used to compare two nodes in min heap. Initially, the least frequent character is at root)
2. Extract two nodes with the minimum frequency from the min heap.
3. Create a new internal node with a frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and the other extracted node as its right child. Add this node to the min heap.
4. Repeat steps#2 and #3 until the heap contains only one node. The remaining node is the root node and the tree is complete.

Determine Huffman coding for the following:

Characters	Frequencies
a	10
e	15
i	12
o	3
u	4
s	13
t	1



e = 0 = 15*1 = 15

s = 10 = 13 * 2 = 26

i = 110 = 12 * 3 = 36

a = 1111 = 10 * 4 = 40

u = 11100 = 4 * 5 = 20

t = 111010 = 1 * 6 = 6

o = 111011 = 3 * 6 = 18

= 161 bits

Avg bits = 161/58 = 2.77 bits/characters

Soumya Dubey

E21CSEU0760

1.Counting sort:

Counting sort is a sorting algorithm that sorts the elements of an array by counting the number of occurrences of each unique element in the array. The count is stored in an auxiliary array and the sorting is done by mapping the count as an index of the auxiliary array.

```
countingSort(array, size)
max <- find largest element in array
initialize count array with all zeros
for j <- 0 to size
find the total count of each unique element and
store the count at jth index in count array
 for i <- 1 to max
find the cumulative sum and store it in count array itself
for j <- size down to 1
restore the elements to array
decrease count of each element restored by 1
```

Use counting sort to sort elements in given array.

S= {2, 5, 4, 2, 3, 4, 2, 0}

Count sort: Stable Sorting Algorithm

1	0	3	1	2	1
0	1	2	3	4	5

0	2	2	2	3	4	4	5

2. Radix sort: Radix sort is a linear sorting algorithm that sorts the elements by first grouping the individual digits of the same place value. Then, sort the elements according to their increasing/decreasing order.

Suppose we have an array of 8 elements. First, we will sort elements based on the value of the unit place. Then, we will sort elements based on the value of the tenth place. This process goes on until the last significant place.

Algorithm:

1. radixSort(arr)
2. max = largest element in the given array
3. d = number of digits in the largest element (or, max)
4. Now, create d buckets of size 0 - 9
5. for i -> 0 to d
6. sort the array elements using counting sort (or any stable sort) according to the digits at the i^{th} place

Use radix sort to sort elements in given array.

A= [682,244,73,6,535,123]

Radix sort: Least Significant Digit

682	6	8	2
244	2	4	4
073	0	7	3
006	0	0	6
535	5	3	5
123	1	2	3

First, find digit in max elements (3)

Arranging in increasing order of zero place we will sort unit place in increasing order

682

073

123

244

535

006

Arranging in increasing order of tenth place

006

123

535

244

073

682

Arranging in increasing order of hundredth place

006

073

123

244

535

682

Hence final sorted list is:

6	73	123	244	535	682

3. Bucket Sort: Bucket Sort is a sorting algorithm that divides the unsorted array elements into several groups called buckets. Each bucket is then sorted by using any of the suitable sorting algorithms or recursively applying the same bucket algorithm. Finally, the sorted buckets are combined to form a final sorted array.

Scatter Gather Approach

The process of bucket sort can be understood as a scatter-gather approach. Here, elements are first scattered into buckets then the elements in each bucket are sorted. Finally, the elements are gathered in order.

Use bucket sort to sort elements in given array.

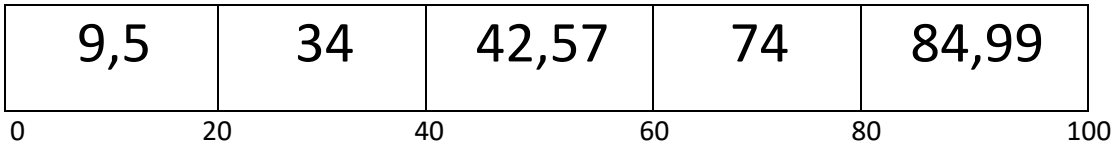
A={34,42,74,57,99,84,9,5}

= Highest number of element in the array/Total no. of elements in the array

= 99/8

=12.3

5,9		34	42	57		74,84		99
0-12	12-24	24-36	36-48	48-60	60-72	72-84	84-96	96-108


9,5
34
42,57
74
84,99
0
20
40
60
80
100

Tutorial 6

Soumya Dubey

E21CSEU0760

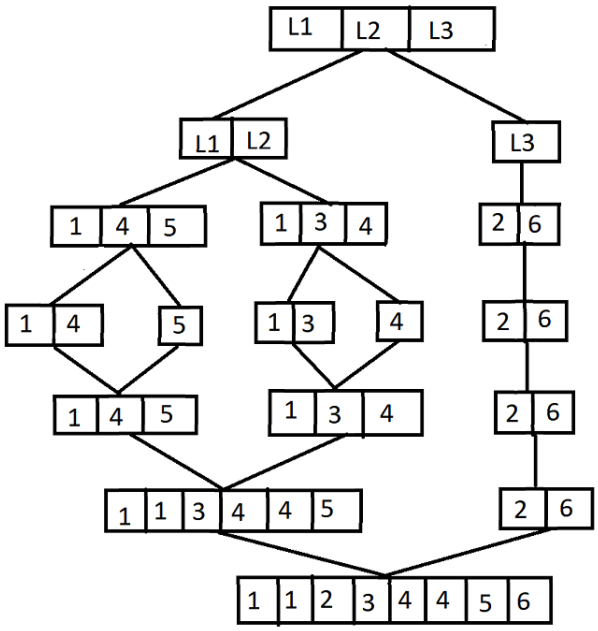
1- Consider 'k' array if given where the array is sorted in ascending order. You need to merge all the array into one sorted array using divide and conquer approach. The array is given as follows:
Given that k=3,
Arr1 = [1, 4, 5]
Arr2 = [1, 3, 4]
Arr3 = [2, 6]
Output= [1, 1, 2, 3, 4, 4, 5, 6]
Hint: Use merge procedure from merge sort: Since there are k arrays that are already sorted, merge the k arrays. Create a recursive function which will take k arrays and divide them into two parts and call the function recursively with each half. The base cases are when the value of k is less than 3.
Steps to follow:
A. Initialize the output array with the size n*k.
B. Call the function divide. Let L and R represent the range of arrays that are to be merged and thus vary between 0 to k-1.
C. At each step, we call the left and right half of the range recursively so that, they will be sorted and stored in the output array.
D. After that, we merge the left and right half. For merging, we need to determine the range of indexes for the left and right halves in the output array. We can easily find that.
    a. Left part will start from the index L * n of the output array.
    b. Similarly, right part will start from the index ((L + R) / 2 + 1) * n of the output array.

Sol. k =3

Arr1 = [1,4,5]

Arr2 = [1,3,5]

Arr3 = [2,6]

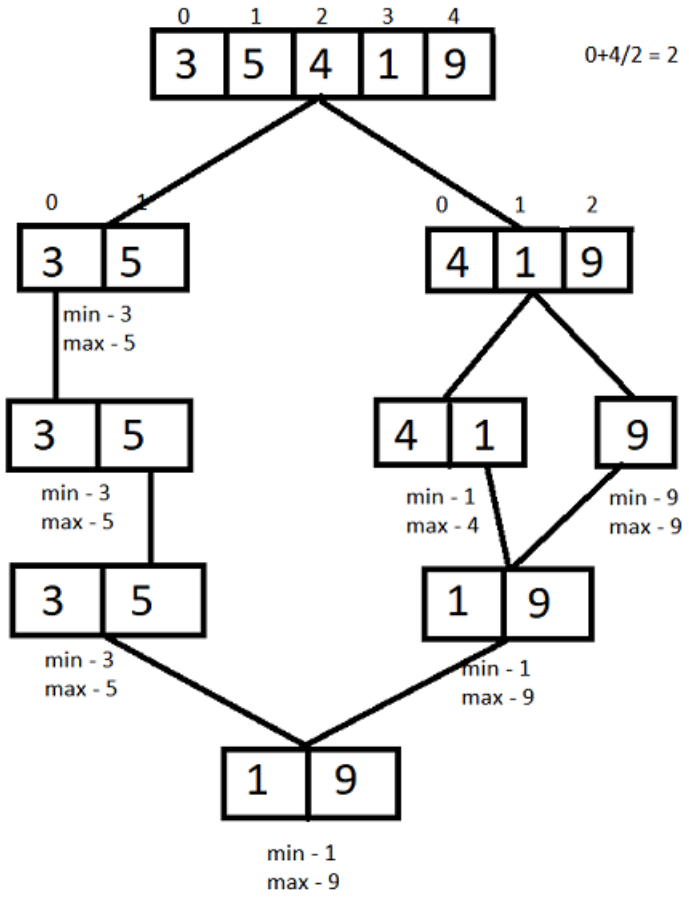


2- Consider a problem where list of elements are given in an unsorted order. The task is to find out the maximum and minimum element using the divide and conquer approach.

The given list is: [3, 5, 4, 1, 9]

Hint:

```
Pair MaxMin(array, array_size)
  if array_size = 1
    return element as both max and min
  else if arry_size = 2
    one comparison to determine max and min
     return that pair
  else   /* array_size  > 2 */
    recur for max and min of left half
    recur for max and min of right half
    one comparison determines true max of the two candidates
    one comparison determines true min of the two candidates
    return the pair of max and min
```



3- Consider two sorted array arr1 and arr2 with size n. Find out the median of the two sorted array so that the algorithm taken O(logn) complexity.

For example, consider the following arrays:
num1 = {1, 12, 15, 26, 38}
num2 = {2, 13, 17, 30, 45}
med1 = 15 and med2 = 17 for num1[0 to 4] and num2[0 to 4]
med1 < med2. So median is present in {15, 26, 38} and {2, 13, 17}
now, med1 = 26 and med2 = 13
med1 > med2. So median is present in {15, 26} and {13, 17}
The size of both the subarrays are now 2. So,
median = (max(15,13) + min(26, 17))/2 = 16

Steps to follow:

1. First calculate the median of both the arrays i.e. arr1 (firstMedian), and arr2 (secondMedian).
2. If both the firstMedian and the secondMedian are equal then we have found our answer. So, return either firstMedian or secondMedian as the answer.
3. If the firstMedian is greater than the secondMedian then the required median must lie in the sub-arrays:
   - from the first element of a1 to the firstMedian.
   - or from the secondMedian to the last element of a2.
4. If the firstMedian is smaller than the secondMedian then the required median must lie in the sub-arrays:
   - from the firstMedian to the last element of a1.
   - or from the first element of a2 to the secondMedian.
5. Repeat the process until the size of the sub-arrays becomes 2.
   If the size of the sub arrays becomes 2, then median is:
   Median = (max(a1[0], a2[0]) + min(a1[1], a2[1]))/2

Solve the following to find out the median
List1= [1, 2, 9]

List2= [3, 4, 7]

Output: 3+4/2=3.5

Sol. num1 = [1,2,9]

num2 = [3,4,7]

med1 = 2 and med2 = 4 for num1[0 to 2] and num2[0 to 2]

med1 < med2. So median is present in [2,9] and [3,4]

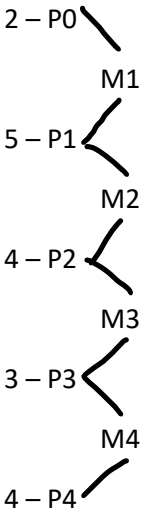
So, median = (max(2,3) + min(9,4))/2 = (3 + 4)/2 = 7/2 = 3.5

Tutorial 7

Soumya Dubey

E21CSEU0760

1. Find a minimum number of multiplications required to multiply for the given sequence {2, 5, 4, 3, 4].



M1*M2 = 2*5*5*4 = 2*5*4 = 40

M2*M3 = 5*4*4*3 = 5*4*3 = 60

M3*M4 = 4*3*3*4 = 4*3*4 = 48

	1	2	3	4
1	0	40	64	88
2		0	60	120
3			0	48
4				0

M1*M2*M3 =M (1, 3)

Min{(M(1,2) + M(3,3) + P0P2P3

= 40 + 0 + (2*4*3)

= 40 + 24

= 64

M (1,2) + M (2,3) + P0P1P3

=0 + 60 + (2*5*3)

= 60 + 30

= 90

Min {64, 90}

M (1, 3) = 64

M (2,4) = Min {M (2,3) + M (4,4) + P1P2P4

= 60 + 0 + (5*3*4)

= 60 + 60

= 120

M (2, 2) + M (3,4) + P1P2P4

= 0 + 43 + (5*4*4)

= 48 + 80

= 128

Min {120, 128}

M (2, 4) = 120

M (1,4) = M1 M2 M3 M4

(M1*M2*M3) *M4

M1*(M2*M3* M4)

(M1*M2) *(M3*M4)

M (1,4) = Min {M (1,3) + M (4,4) + P0 P3 P4

= 64 + 0 + (2*3*4)

= 64 + 24

= 88

M (1,2) + M (3, 4) + P0 P2 P4

= 0 + 120 + (2*5*4)

= 120 + 40

= 160

Min {88, 120, 160]

= 88

M (1, 4) = 88

2. Compute the optimal sequence of multiplication and cost for the matrices P[3x4], Q[4x3], R[3x5] and S[5x4].

P – 3*4

Q – 4*3

R – 3*5

S - 5*4

D0 = 3

D1 = 4

D2 = 3

D3 = 5

D4 = 4

	1	2	3	4
1	0	60	81	132
2		0	60	108
3			0	60
4				0

M(1,2) = M(1,1) + M(2,2) + D0*D1*D2

= 0 + 0 + 3*4*3

= 36

M(2,3) = M(2,2) + M(3,3) + D1*D2*D3

= 0 + 0 + 4*3*5

= 60

M (3,4) = M (3,3) + M (4,4) + D2*D1*D3

= 0 + 60 + 3*4*5

= 120

M (1,2) + M (3,3) + D0*D2*D3

= 36 + 0 + 3*3*5

= 21

Therefore, M (1,3) = 81

M(2,4) = M(2,2) + M(3,4) + D1*D2*D4

= 0 + 60 + 4*3*4

= 108

M (2,3) + m(4,4) + D2*D3*D4

= 60 + 0 + 4*5*4

=140

Therefore, M (2,4) = 108

M(1,4) = M(1,1) + M(2,4) + D0*D1*D4

0 + 108 + 3*4*4

= 156

M (1,2) = M (3,4) + D0*D2*D4

= 36 + 60 + 3*3*4

= 132

M(1,3) + M(4,4) + D0*D3*D4

= 81 + 0 + 3*5*4

= 141

Therefore, M (1,4) = 132

Minimum optimus = 132

Cost = 30 Rs

Tutorial 8

Soumya Dubey

E21CSEU0760

Problems to ponder

Problem-1

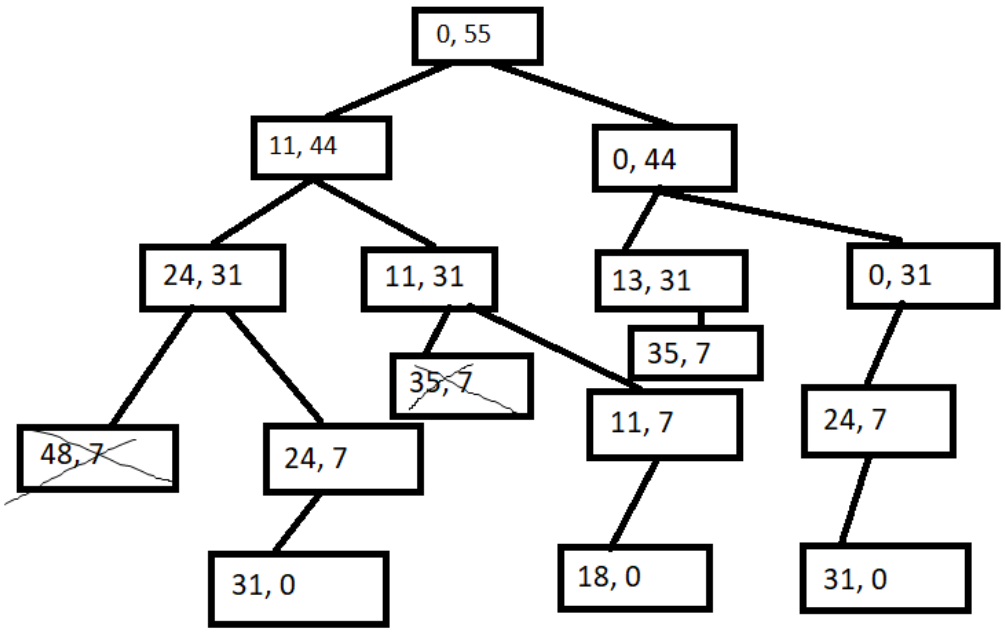
Let us consider the instance of subset sum problem with M=31, S = {11, 13, 24, 7}

a. Apply backtracking and draw the state space tree
b. Find 2 subsets of S whose element sum is M

M = 31

S = {11, 13. 24, 7}

11+13+24+7 = 55



b) {11, 13, 7}

{24, 7}

Problem-2

What is the nature of time complexity in this scenario? (Choose the correct one)

a) Exponential

b) Linear

c) Constant

d) Logarithmic

Ans a) Exponential

Problem-3

Let us consider the instance of subset sum problem with M=15, S = {3, 5,10, 12, 20}

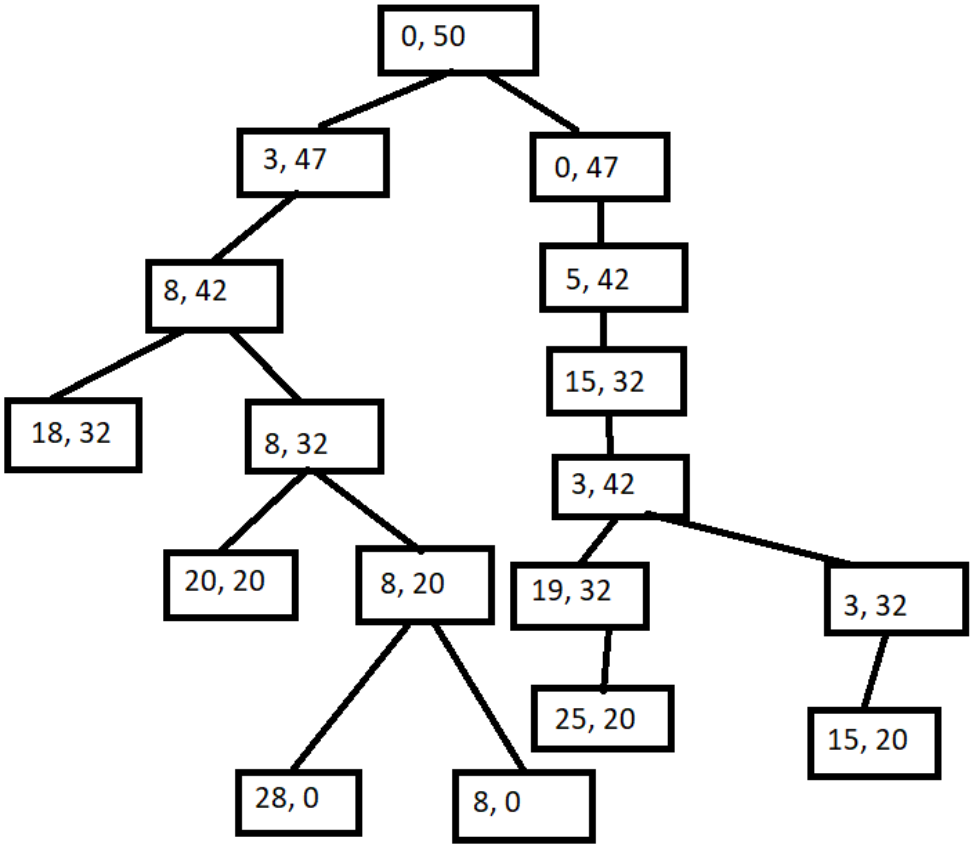
a. Apply backtracking and draw the state space tree

b. Find all subsets of S whose element sum is M

M = 15

S = {3,5,10,12,20}

3+5+10+12+20 = 50


0, 50
3, 47
0, 47
8, 42
5, 42
15, 32
18, 32
8, 32
3, 42
20, 20
8, 20
19, 32
3, 32
25, 20
15, 20
28, 0
8, 0

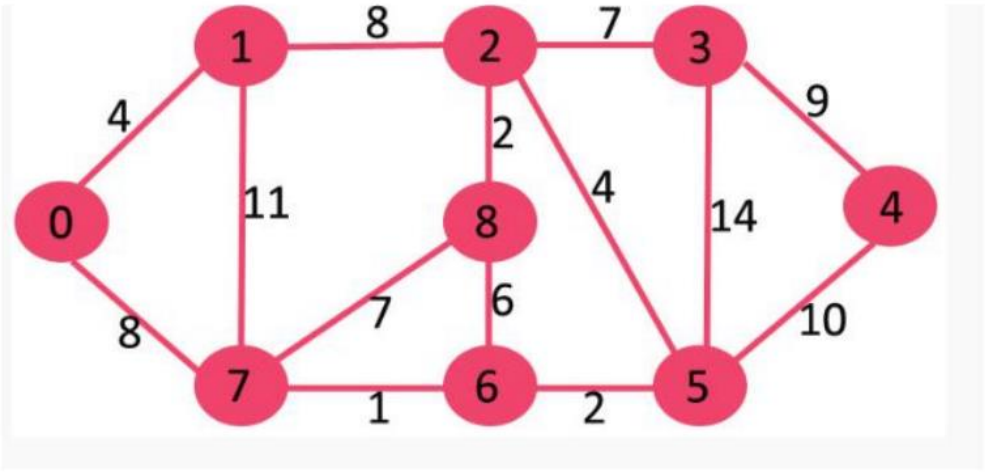
Ans. B) {5, 10}

{3, 12}

Tutorial 9

E21CSEU0760

1. Consider the following graph and compute the shortest path from source 'O' to all other vertices.



V stands for visited

D stands for distance

V

F	F	F	F	F	F	F	F	F
0	1	2	3	4	5	6	7	8

D

∞	∞	∞	∞	∞	∞	∞	∞	∞
0	1	2	3	4	5	6	7	8

V

T	F	F	F	F	F	F	F	F
0	1	2	3	4	5	6	7	8

D

0	∞	∞	∞	∞	∞	∞	∞	∞
0	1	2	3	4	5	6	7	8

V

T	T	F	F	F	F	F	T	F
0	1	2	3	4	5	6	7	8

D

0	4	∞	∞	∞	∞	∞	8	∞
0	1	2	3	4	5	6	7	8

V

T	T	T	F	F	F	F	T	F
0	1	2	3	4	5	6	7	8

D

0	4	12	∞	∞	∞	∞	8	∞
0	1	2	3	4	5	6	7	8

V

T	T	T	F	F	F	T	T	T
0	1	2	3	4	5	6	7	8

D

0	4	12	∞	∞	∞	9	8	15
0	1	2	3	4	5	6	7	8

V

T	T	T	F	F	T	T	T	T
0	1	2	3	4	5	6	7	8

D

0	4	12	∞	∞	11	9	8	15
0	1	2	3	4	5	6	7	8

V

T	T	T	F	T	T	T	T	T
0	1	2	3	4	5	6	7	8

D

0	4	12	25	21	11	9	8	15
0	1	2	3	4	5	6	7	8

V

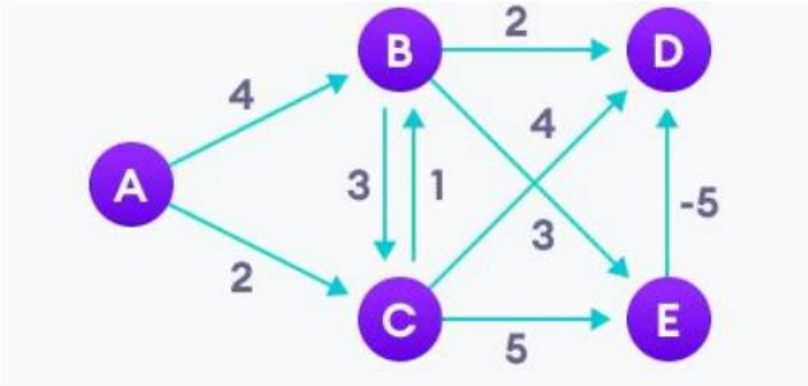
T	T	T	T	T	T	T	T	T
0	1	2	3	4	5	6	7	8

D

0	4	12	19	21	11	9	8	14
0	1	2	3	4	5	6	7	8

0 → 1 →7 → 6 → 5 → 2 → 8 → 3 → 4

2. Compute the shortest distance from source 'A' to all other vertices using Bellman Ford Algorithm.



Edge list: - (A, B), (A,C), (B,C), (B,D), (B,E), (C,B), (C,D), (C,E), (E,D).

We take A as 0 (taking it as a starting point) and rest are set to infinity.

As we know: - If (d[u] + (u,v) < d[v])

d[v] = d[u] + c[u,v]

A	B	C	D	E
0	∞	∞	∞	∞

As we can see, (A, B) = 0+4 = 4< infinity (so we update it)

(A, C) = 0+2 = 22 (so not updated)

(B, D) = 4+2 = 6< infinity (updated)

(B, E) = 4+3 = 7< 4 (updated)

(C, D) = 2+4 = 6 = 6 (same value, not updated)

(C, E) = 2+5 = 7 > 6(not updated)

(E, D) = 7 – 5 = 2 < 6 (updated)

Hence, vice-versa if (If (d[u] + (u,v) < d[v]) , d[v] = d[u] + c[u,v]), then we update the value

A	B	C	D	E
0	∞	∞	∞	∞
0	3	2	2	7
0	3	2	1	6
0	3	2	1	6

Tutorial 10

E21CSEU0760

1. Given string=xabcabzabc and the pattern=abc. Find the z array of string, z array of new string formed by concatenation of pattern and string. And tell whether pattern is present in the string or not.

Concatenated string "abc$xabcabzabc"

i = 1, L = 0, R = 0, Z[1] = 0

i = 2, L = 0, R = 0, Z[2] = 0

i = 3, L = 0, R = 0, Z[3] = 0

i = 4, L = 3, R = 6, k = 1, Z[4] = 0

i = 5, L = 3, R = 6, k = 2, Z[5] = 3

i = 6, L = 3, R = 6, k = 3, Z[6] = 0

i = 7, L = 7, R = 8, k = 0, Z[7] = 0

i = 8, L = 7, R = 8, k = 1, Z[8] = 2

i = 9, L = 7, R = 8, k = 2, Z[9] = 0

i = 10, L = 10, R = 10, k = 3, Z[10] = 0

i = 11, L = 10, R = 10, k = 0, Z[11] = 3

i = 12, L = 10, R = 10, k = 1, Z[12] = 0

i = 13, L = 13, R = 12, k = 2, Z[13] = 0

pattern found at = 8

Z = [x, 0, 0, 0, 0, 3, 0, 0, 2, 0, 0, 2,0,0]

2. Given string=aabxaaab and the pattern=bxa. Find the z array of string, z array of new string formed by concatenation of pattern and string. And tell whether pattern is present in the string or not.

Concatenated string " bxa$aabxaaab "

i = 1, L = 0, R = 0, Z[1] = 0

i = 2, L = 0, R = 0, Z[2] = 0

i = 3, L = 0, R = 0, Z[3] = 0

i = 4, L = 3, R = 6, k = 1, Z[4] = 0

i = 5, L = 3, R = 6, k = 2, Z[5] = 0

i = 6, L = 3, R = 6, k = 3, Z[6] = 3

i = 7, L = 7, R = 8, k = 0, Z[7] = 0

i = 8, L = 7, R = 8, k = 1, Z[8] = 0

i = 9, L = 7, R = 8, k = 2, Z[9] = 0

i = 10, L = 10, R = 10, k = 3, Z[10] = 0

i = 11, L = 10, R = 10, k = 0, Z[11] = 1

pattern not found

Z = [x, 0, 0, 0, 0, 0,0, 3, 0, 0, 0, 0, 1]

3. What is time complexity of Z algorithm?

The time complexity of the Z algorithm is O (n + m), where n is the length of the text string and m is the length of the pattern being searched for.

Tutorial 11

E21CSEU0760

Rabin Karp String Matching Algorithm

Problem 1:

Working modulo q = 17, how many spurious hits does the Rabin-Karp matcher encounter in the text T = 2359023141526739921 when looking for the pattern P = 7399?

h(P) = P mod q = 7399 mod 17 = 4

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index 0 to 3

2359 mod 17 = 13 not equal to 4 (not match)

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index[1: 4]

3590 mod 17 = 3 not equal to 4 (no match)

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [2 : 5]

5902 mod 17 = 3 not equal to 4(no match)

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [3 : 6]

9023 mod 17 = 13 not equal to 4 (no match)

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [4: 7]

0231 mod 17 = 10 not equal to 4 (no match)

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [5: 8]

2314 mod 17 = 2 not equal to 4 (no match]

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [6: 9]

3141 mod 17 = 13 not equal to 4 (no match)

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index[7 : 10]

1415 mod 17 = 4 but 1415 not equal to 7399 so it is spurious hit

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [8: 11]

4152 mod 17 = 4 but 4152 not equal to 7399 so it is a spurious hit

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [9 : 12]

1526 mod 17 = 13 not equal to 4 (no match)

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [10: 13]

5267 mod 17 = 13 not equal to 4 (no match)

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [11: 14]

2673 mod 17 = 4 but 2163 not equal to 7399 so it is a spurious hit

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [12: 15]

6739 mod 17 = 7 not equal to 4(no match)

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [13 : 16]

7399 mod 17 = 4 and it is an exact match.

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index [14 : 17]

3992 mod 17 = 14 not equal to 4(no match)

2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

For index[15 : 18]

9921 mod 17 = 10 not equal to 4(no match)

Problem 2:

Working modulo q = 11, how many spurious hits does the Rabin-Karp matcher encounter in the text T = 3151429563589793 when looking for the pattern P = 589

h(P) = P mod q = 589 mod 11 = 6

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [0:2]

315 mod 11 = 7 not equal to 6 (not match)

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index[1: 3]

151 mod 11 = 8 not equal to 6 (no match)

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [2 : 4]

514 mod 11 = 8 not equal to 6 (no match)

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [3 : 5]

142 mod 11 = 10 not equal to 6 (no match)

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [4: 6]

429 mod 11 = 0 not equal to 6 (no match)

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [5: 7]

295 mod 11 = 9 not equal to 6 (no match]

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [6: 8]

956 mod 11 = 10 not equal to 6 (no match)

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index[7 : 9]

563 mod 11 = 2 not equal to 6(no match)

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [8: 10]

635 mod 11 = 4 not equal to 6(no match)

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [9 : 11]

358 mod 11 = 6 but 358 not equal to 589 so it is a spurious hit

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [10: 12]

589 mod 11 = 6 equal to 6 and it is an exact match.

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [11 : 13]

897 mod 11 = 6 but 897 not equal to 589 so it is a spurious hit.

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index [12 : 14]

979 mod 11 = 0 not equal to 6(no match)

3	1	5	1	4	2	9	5	6	3	5	8	9	7	9	3

For index[13 : 15]

793 mod 11 = 1 not equal to 6(no match)

Problem 3:

What is the time complexity of Rabin Karp string matching algorithm where length of T = n and length of P =m?

The average case and best case complexity of Rabin-Karp algorithm is O (m + n) and the worst case complexity is O (m n).

Problem 4:

Suggest a good Hash function to quickly compute the hash value of substring of text T corresponding to pattern P. (**Hint**: use a Rolling Hash Function)

This rolling hash function has the property that the hash value of each substring can be computed in O(1) time, and it is very unlikely to have hash collisions for different substrings. This makes it an effective choice for the Rabin-Karp algorithm and other string matching algorithms that use hash functions.

Tutorial 12

Soumya Dubey

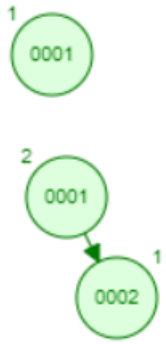
E21CSEU0760

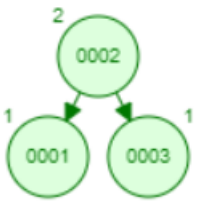
1. What is the difference between AVL and Red Black tree. Specify clearly.

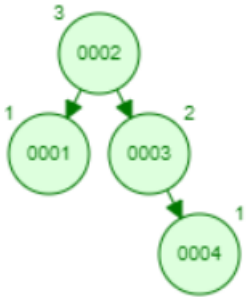
AVL Tree	Red Black Tree
Strictly height-balanced tree	Roughly height-balanced tree
Comparatively complex to implement	Easy to implement
Takes more processing for balancing	Takes less processing for balancing because a maximum of two rotations are required.
Searching operation is faster because of height-balanced structure.	Searching operation is comparatively slow
Insertion and deletion operations are slow	Insertion and deletion operations are faster
No colour for the nodes	The nodes can be either red or black
Balance factor is associated with each node	Balance factor is associated with each node
Used in databases for faster retrievals	Used in the language libraries

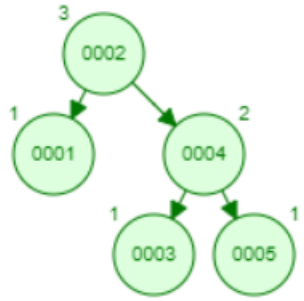
Q2)

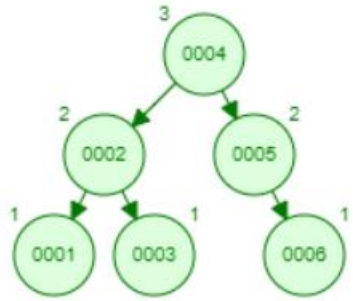
Construct an AVL Tree by inserting numbers from 1 to 8.

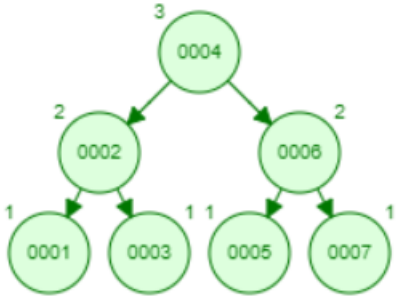


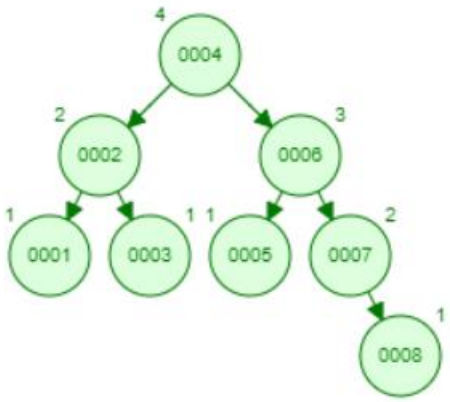

2
0002
1
0001
0003
1


3
0002
1
0001
0003
2
0004
1


3
0002
1
0001
0004
2
0003
0005
1
1

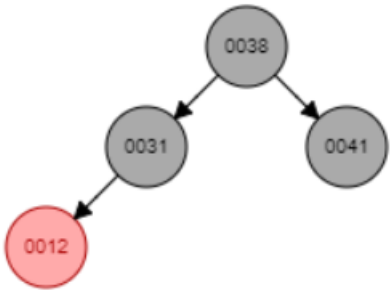

3
0004
2
0002
0005
2
1
0001
0003
1
0006
1

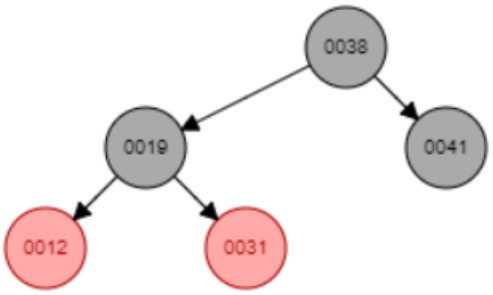

3
0004
2
0002
0006
2
1
0001
0003
1
1
0005
0007
1

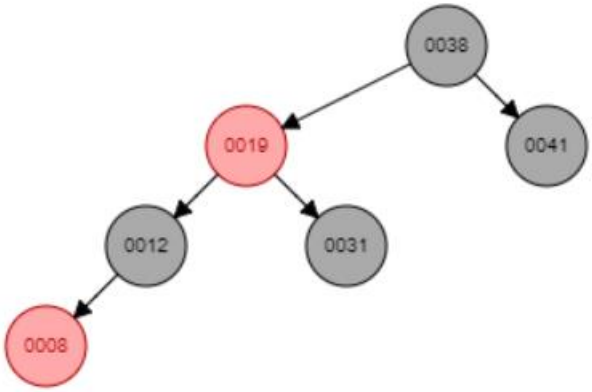


Q3)

Show the red-black trees that result after successively inserting the keys 41,38,31,12,19,8 into an initially empty red-black tree.




0038
0019
0041
0012
0031


0038
0019
0041
0012
0031
0008