

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institution Affiliated to VTU, Belagavi)

SHAVIGE MALLESHWARA HILLS, K.S.LAYOUT, BANGALORE-560078

Department of Computer Science and Engineering



2019-2020

4th Semester

DESIGN AND ANALYSIS OF ALGORITHMS

LAB MANUAL

(18CS4DL DAL)

Compiled by

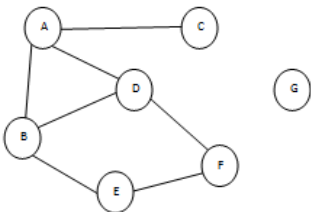
Prof. A M Prasad

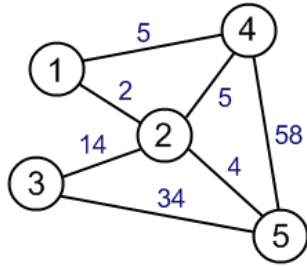
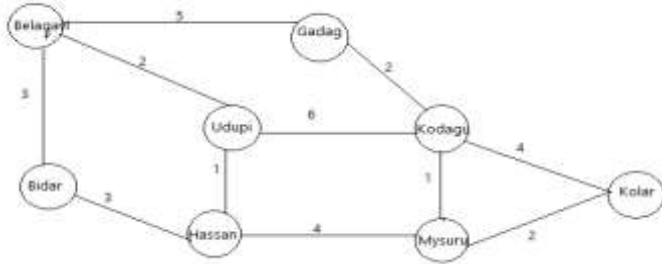
Prof. Harish Kumar N

Prof. Annapoorna B R

Prof. Sahana D

Prof. Kusuma H

Experiment No.	Contents of the Experiment	Page No:
1a.	Write a C program to implement Maximum-sub array problem where, you are given a one dimensional array that may contain both positive and negative integers, and find the sum of contiguous sub array of numbers which has the largest sum.	5-6
1b.	In a Stock market you can buy any unit of stock and then sell it at same date before the close of trading for the day or later date. If you want to learn what the price of the stock will be in the future and want to maximize your profit, use suitable algorithm design approach and implement the same.	
2a.	i) Using Decrease and Conquer strategy design and execute a program in C, to print all the nodes reachable from a given starting node in a graph using BFS method. ii) Using Decrease and Conquer strategy develop a program in C to check whether a given graph is connected or not using DFS method.	6-8
2b.	Navigation systems such as the Google Maps, which can give directions to reach from one place to another, take your location to be the source node and your destination as the destination node on the graph. Graph traversal methods are used to find if the destination entered can be reachable or not. Hint: Any city can be represented as a graph taking landmarks as nodes and roads as edges. Using an appropriate technique check if your desired destination is reachable or not considering the following graph. 	
3a.	Design and execute a program in C to search for the pattern string in given text string using Boyer-Moore String Matching algorithm.	9-10
3b.	Spam (Unsolicited and unwanted emails) is annoying, no doubt, but it can also be dangerous. Malware and phishing are hugely profitable for scammers and can be costly for mailbox providers' customers, as well as the mailbox providers who face intense market competition. To avoid such problem mailbox providers' use Spam filters. All spam filters use the concept of string matching to identify and discard the spam. Considering the above scenario use an appropriate technique to classify the mails if it's a spam or not when there are patterns like "free," "money," "help" and "prize" in the mail content.	
4a.	Given two sequences $X = \langle x_1; x_2; : : : x_m \rangle$, $Y = \langle y_1; y_2; : : : y_n \rangle$ and required to find a longest-common-subsequence , of X and Y using	10-12

	dynamic programming.	
4b.	DNA-based identity testing is extensively used in the forensic field. DNA sequences can be viewed as strings of A, C, G, and T characters, which represent nucleotides. To compare and analyze two such strings, the longest subsequence is necessary. With an appropriate approach print the longest subsequence for two DNA sequences.	
5a.	Implement Kruskal algorithm in a function KRUSKAL. Execute this function, giving cost adjacency matrix of a undirected graph as input and output its Minimum Spanning Tree.	12-13
5b.	<p>Laying out electrical networks to be carried in a city that minimizes the total cost of the wiring. Choose appropriate design strategy to connect all the houses so that wiring cost is minimum.</p> <p>Given : Layout input as a graph with the distances. Nodes 1,2,3,4 and 5 represent homes and the edges with cost gives the wiring cost between the homes.</p> 	
6a.	Design and execute a program in C to create a function called bellman-ford that represents the cost adjacency matrix. From a given vertex in a weighted connected graph, find shortest paths from a single source vertex to all of the other vertices using Bellman-Ford algorithm.	13-14
6b.	<p>In the google map, the navigation system always shows the shortest path to travel from source to destination. State can be represented as a weighted graph by taking districts as nodes and roads as the edges that connects the districts.</p> <p>Considering the above scenario, apply single source shortest path method to find the shortest distance between two districts for the graph.</p> 	

7a.	Design and execute a program in C to create a function called SUMOFSUBSET that represents the array of elements, and to find a subset of a given set $S = \{s_1, s_2, \dots, s_n\}$ of n positive integers whose sum is equal to a given positive integer d . For example, if $S = \{1, 2, 5, 6, 8\}$ and $d = 9$ there are two solutions $\{1, 2, 6\}$ and $\{1, 8\}$. A suitable message is to be displayed if the given problem instance doesn't have a solution.	14-15																				
7b.	<p>In an Education institution, there are faculties with varied number of teaching experience. For an upcoming semester Computer Science department requires a team of faculty with combined experience of 9 years to form a syllabus for Data Science course.</p> <p>Consider the table below with information about the teaching experience of Computer Science faculty. Apply an appropriate backtracking technique to solve the above scenario.</p> <table><tr><th>SL.No</th><th>Faculty</th><th>Teaching Experience</th></tr><tr><td>1)</td><td>Mr. John</td><td>1 yr</td></tr><tr><td>2)</td><td>Mr. Jacob</td><td>2 yrs</td></tr><tr><td>3)</td><td>Mr. Dave</td><td>3 yrs</td></tr><tr><td>4)</td><td>Mrs. Emily</td><td>6 yrs</td></tr><tr><td>5)</td><td>Mrs. Ava</td><td>7 yrs</td></tr><tr><td>6)</td><td>Ms. Jessica</td><td>8 yrs</td></tr></table>		SL.No	Faculty	Teaching Experience	1)	Mr. John	1 yr	2)	Mr. Jacob	2 yrs	3)	Mr. Dave	3 yrs	4)	Mrs. Emily	6 yrs	5)	Mrs. Ava	7 yrs	6)	Ms. Jessica
SL.No	Faculty	Teaching Experience																				
1)	Mr. John	1 yr																				
2)	Mr. Jacob	2 yrs																				
3)	Mr. Dave	3 yrs																				
4)	Mrs. Emily	6 yrs																				
5)	Mrs. Ava	7 yrs																				
6)	Ms. Jessica	8 yrs																				

1. Write a C program to implement Maximum-sub array problem where, you are given a one dimensional array that may contain both positive and negative integers, and find the sum of contiguous sub array of numbers which has the largest sum.

```
#include <stdio.h>
#include <limits.h>

// A utility function to find maximum of two integers
int max1(int a, int b) { return (a > b)? a : b; }

// A utility function to find maximum of three integers
int max2(int a, int b, int c) { return max1(max1(a, b), c); }

// Find the maximum possible sum in arr[] such that arr[m] is
// part of it
int maxCrossingSum(int arr[], int l, int m, int h)
{
    // Include elements on left of mid.
    int sum = 0;
    int left_sum = INT_MIN;
    for (int i = m; i >= l; i--)
    {
        sum = sum + arr[i];
        if (sum > left_sum)
            left_sum = sum;
    }

    // Include elements on right of mid
    sum = 0;
    int right_sum = INT_MIN;
    for (int i = m+1; i <= h; i++)
    {
        sum = sum + arr[i];
        if (sum > right_sum)
            right_sum = sum;
    }

    // Return sum of elements on left and right of mid
    return left_sum + right_sum;
}

// Returns sum of maximum sum subarray in aa[l..h]
int maxSubArraySum(int arr[], int l, int h)
{

```

```

// Base Case: Only one element
if (l == h)
    return arr[l];

// Find middle point
int m = (l + h)/2;

/* Return maximum of following three possible cases
a) Maximum subarray sum in left half
b) Maximum subarray sum in right half
c) Maximum subarray sum such that the subarray crosses the
midpoint */
return max2(maxSubArraySum(arr, l, m),
            maxSubArraySum(arr, m+1, h),
            maxCrossingSum(arr, l, m, h));
}

/*Driver program to test maxSubArraySum*/
int main()
{
    int arr[20],n,i;
    printf("enter the number of elements\n");
    scanf("%d",&n);
    printf("enter the elements\n");
    for(i=0;i<n;i++)
        scanf("%d",&arr[i]);
    int max_sum = maxSubArraySum(arr, 0, n-1);
    printf("Maximum contiguous sum is %d\n", max_sum);
    getchar();
    return 0;
}

```

2. i) Using Decrease and Conquer strategy design and execute a program in C, to print all the nodes reachable from a given starting node in a graph using BFS method.
- ii) Using Decrease and Conquer strategy develop a program in C to check whether a given graph is connected or not using DFS method.

i. #include<stdio.h>

```

void bfs(int n,int a[10][10],int source,int s[10])
{
    int i,f=0,r=-1,q[10],v;

    printf("%d--->",source);
    s[source]=1;
    q[++r]=source;

```

```

        while(f<=r)
        {
            v=q[f++];

            for(i=1;i<=n;i++)
                if(s[i]==0 && a[v][i])
                {
                    q[++r]=i;
                    printf("%d--->",i);
                    s[i]=1;
                }
        }
    }

void main()
{
    int n,a[10][10],i,j,source,s[10];

    printf("enter the number nodes in the graph\n");
    scanf("%d",&n);

    printf("enter the matrix for the graph\n");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            scanf("%d",&a[i][j]);

    printf("enter the source node\n");
    scanf("%d",&source);

    for(i=1;i<=n;i++)
        s[i]=0;

    printf("the BFS traversal is\n");

    bfs(n,a,source,s);

    printf("\nthe nodes reachable are\n");

    for(i=1;i<=n;i++)
        if(s[i])
            printf("%d\n",i);
}

```

ii. #include<stdio.h>

```
void dfs(int n,int a[10][10],int s[10],int source);
```

```
void main()
{
    int j, s[10], a[10][10], source, i, n, flag=0;

    printf("Enter the number of nodes:\n");
    scanf("%d",&n);

    printf("Enter the adjacency matrix:\n");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            scanf("%d",&a[i][j]);

    printf("Enter the source node:\n");
    scanf("%d",&source);
    for(i=1;i<=n;i++)
        s[i]=0;
```

```
dfs(n,a,s,source);
```

```
    for(i=1;i<=n;i++)
        if(s[i]==0)
            flag=1;
```

```
    if(flag==1)
        printf("graph not connected");
    else
        printf("graph is connected");
```

```
}
```

```
void dfs(int n,int a[10][10],int s[10],int source)
```

```
{
    int i;
    s[source]=1;
    printf("-->%d",source);
    for(i=1;i<=n;i++)
        if(s[i]==0 && a[source][i]==1)
            dfs(n,a,s,i);
}
```


3. Design and execute a program in C to search for the pattern string in given text string using Boyer-Moore String Matching algorithm.

```
/* Program for Bad Character Heuristic of Boyer Moore String Matching Algorithm
*/
```

```
# include <limits.h>
# include <string.h>
# include <stdio.h>
# define MAX 256
int bc[MAX];

// A utility function to get maximum of two integers
int max(int a, int b)
{
    return (a > b) ? a : b;
}

// The preprocessing function for Boyer Moore's bad character heuristic
void bctable(char p[])
{
    int i,m,j;
    m=strlen(p);
    // Initialize all occurrences as -1
    for (i = 0; i < MAX; i++)
        bc[i] = m;
    // Fill the actual value of last occurrence of a character
    for (j = 0; j < m-1; j++)
        bc[p[j]] = m-1-j;
}

int boyer(char txt[], char p[])
{
    int m = strlen(p);
    int n = strlen(txt);
    // s is shift of the pattern with respect to text
    int i=m-1;
    while (i<n)
    {
        int k=0;
        while (k<m && p[m-1-k] == txt[i-k])
            k++;
        if (k==m)
            return(i-m+1);
        else
            i=i+ max(1, bc[txt[i]]-k);
    }
}
```

```

return -1;
}

/* Driver program to test above funtion */
int main()
{
    int pos;
    char txt[] = "BESS KNEW ABOUT BAOBABS";
    char pat[] = "BAOBAB";
    bctable(pat);
    pos=boyer(txt, pat);
    if(pos>=0)
        printf("the pattern found at %d position\n", pos+1);
    else
        printf("pattern not found\n");

    return 0;
}

```

4. Given two sequences $X = \langle x_1; x_2; \dots; x_m \rangle$, $Y = \langle y_1; y_2; \dots; y_n \rangle$ and required to find a longest-common-subsequence, of X and Y using dynamic programming.

```

/* Dynamic Programming implementation of LCS problem */
#include<stdio.h>
#include<string.h>

int max(int a, int b)
{
    if(a>b) return a;
    return b;
}

/* Returns length of LCS for X[0..m-1], Y[0..n-1] */
void lcs( char *X, char *Y, int m, int n )
{
    int L[m+1][n+1];

    /* Following steps build L[m+1][n+1] in bottom up
    fashion. Note
        that L[i][j] contains length of LCS of X[0..i-1] and
        Y[0..j-1] */
    for (int i=0; i<=m; i++)
    {
        for (int j=0; j<=n; j++)
        {
            if (i == 0 || j == 0)

```

```

        L[i][j] = 0;
    else if (X[i-1] == Y[j-1])
        L[i][j] = L[i-1][j-1] + 1;
    else
        L[i][j] = max(L[i-1][j], L[i][j-1]);
    }
}

// Following code is used to print LCS
int index = L[m][n];

// Create a character array to store the lcs string
char lcs[index+1];
lcs[index] = '\0'; // Set the terminating character

// Start from the right-most-bottom-most corner and
// one by one store characters in lcs[]
int i = m, j = n;
while (i > 0 && j > 0)
{
    // If current character in X[] and Y are same, then
    // current character is part of LCS
    if (X[i-1] == Y[j-1])
    {
        lcs[index-1] = X[i-1]; // Put current character
                               // in result
        i--; j--; index--; // reduce values of i, j
                           // and index
    }

    // If not same, then find the larger of two and
    // go in the direction of larger value
    else if (L[i-1][j] > L[i][j-1])
        i--;
    else
        j--;
}

// Print the lcs
printf("LCS of %s and %s is %s", X, Y, lcs);
}

/* Driver program to test above function */
int main()
{
    char X[20], Y[20];
    printf("\nEnter String 1\n");

```

```

scanf("%s",X);
printf("\nEnter String 2\n");
scanf("%s",Y);
int m = strlen(X);
int n = strlen(Y);
lcs(X, Y, m, n);
return 0;
}

```

5. Design, develop, and execute a program in C to create a function called **KRUSKAL** that represents the cost adjacency matrix, Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm.

```

#include<stdio.h>
#include<conio.h>
int parent[20]={0},min,mincost=0,ne=1,n,cost[20][20];
int a,b,i,j,u,v;
void kruskal(void);
void main()
{
    clrscr();
    printf("Enter the number of nodes:");
    scanf("%d",&n);
    printf("Enter the cost matrix:");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    kruskal();
    getch();
}
void kruskal()
{
    while(ne<n)
    {
        for(i=1,min=999;i<=n;i++)
            for(j=1;j<=n;j++)
                if(cost[i][j]<min)
                {
                    min=cost[i][j];
                    a=u=i;
                    b=v=j;
                }
        while(parent[u])

```

```

        u=parent[u];
        while(parent[v])
        v=parent[v];
        if(u!=v)
        {
            printf("%d\\tedge\\t(%d,%d)=%d\\n",ne++,a,b,min);
            mincost+=min;
            parent[v]=u;
        }
        cost[a][b]=cost[b][a]=999;
    }
    printf("The minimum cost=%d\\n",mincost);
}

```

- 6. Design and execute a program in C to create a function called bellman-ford that represents the cost adjacency matrix. From a given vertex in a weighted connected graph, find shortest paths from a single source vertex to all of the other vertices using Bellman-Ford algorithm.**

```

#include <stdio.h>
#include <stdlib.h>
int Bellman_Ford (int G[20][20] , int V, int E, int edge[20][2])
{
    int i, u, v, k, distance[20],parent[20],S, flag=1;
    for(i=0;i<V; i++)
        distance[i] = 1000 , parent[i] = -1 ;
    printf ("Enter source: ");
    scanf ("%d", &S);
    distance[S-1]=0 ;
    for (i=0;i<V-1;i++)
    {
        for (k=0;k<E; k++)
        {
            u = edge[k][0] , v = edge[k][1] ;
            if(distance[u]+G[u][v] < distance[v])
                distance[v] = distance[u] + G[u][v] , parent[v]=u ;
        }
    }
    for(k=0;k<E; k++)
    {
        u = edge[k][0] , v = edge[k][1] ;
        if (distance[u]+G[u][v] < distance[v])
            flag = 0 ;
    }
    if (flag)

```

```

        for (i=0; i<V; i++)
printf ("Vertex %d -> cost = %d parent %d\n", i+1, distance[i],parent[i]+1);

        return flag;
    }
int main()
{
    int V, edge[20][2],G[20][20],i, j, k=0;
    printf ("BELLMAN FORD\n");
    printf ("Enter no. of vertices: ");
    scanf ("%d",&V);
    printf ("Enter graph in matrix form:\n");
    for(i=0;i<V; i++)
        for(j=0;j<V; j++)
        {
            scanf("%d", &G[i][j]);
            if(G[i][j]!=0)
                edge[k][0]=i, edge[k++][1]=j;
        }

    if (Bellman_Ford(G,V,k,edge))
        printf ("\n No negative weight cycle\n");
    else
        printf ("\n Negative weight cycle exists\n");
    return 0;
}

```

- 7. Design and execute a program in C to create a function called SUMOFSUBSET that represents the array of elements, and to find a subset of a given set $S = \{s_1, s_2, \dots, s_n\}$ of n positive integers whose sum is equal to a given positive integer d . For example, if $S = \{1, 2, 5, 6, 8\}$ and $d = 9$ there are two solutions $\{1, 2, 6\}$ and $\{1, 8\}$. A suitable message is to be displayed if the given problem instance doesn't have a solution.**

```

#include<stdio.h>
int total=0,n,i,p,r,x[10],s[10],count=0;
void subset(int k,int sum,int rem)
{
    x[k]=1;
    if(s[k]+sum==total)
    {
        count++;
        printf("\n");
    }
    for(i=1;i<=k;i++)

```

```

{
If(x[i]==1)
printf("%d\t",s[i]);
}
}
else if(s[k]+s[k+1]+sum<=total)
subset(k+1,sum+s[k],rem-s[k]);
if( (sum+rem-s[k]>=total) && (sum+s[k+1]<=total) )
{
x[k]=0;
subset(k+1,sum,rem-s[k]);
}
}
void main()
{
int rem=0;
printf("Enter the number of elements in the set\n");
scanf("%d", &n);
printf("Enter the elements in increasing order\n");
for(i=1;i<=n;i++)
{
scanf("%d",&s[i]);
rem=rem+s[i];
x[i]=0;
}

printf("Enter Max subset value\n");
scanf("%d",&total);
if(total>rem)
printf("Subset not possible\n");
else
{
subset(1,0,r);
if(count==0)
printf("Solution does not exist\n");
else
printf("\nNumber of subsets is %d\n",count);
}
}
}

```