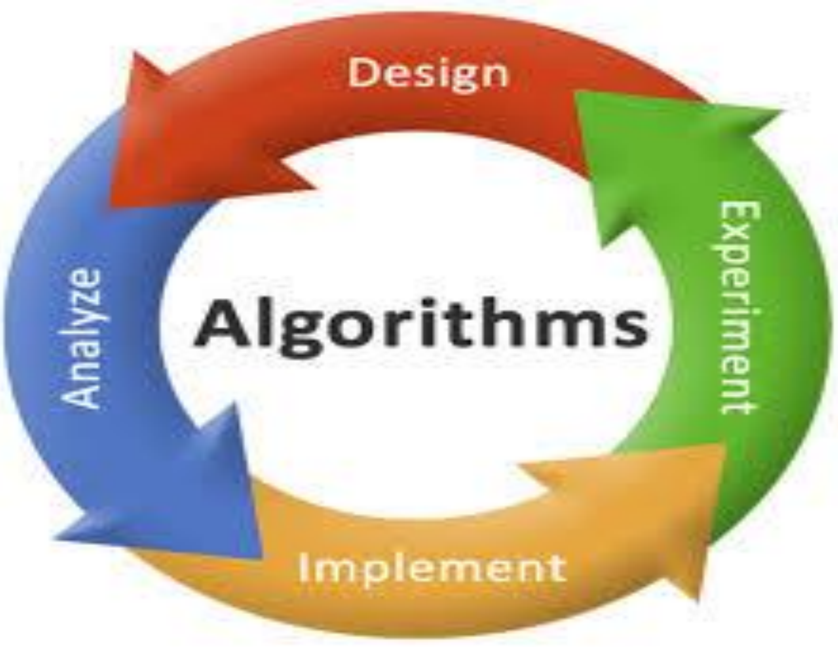
Design and Analysis of Algorithm Lab Manual

Course Code: DSC5-Lab



Prepared By
Mr. Varun K S
Assistant Professor
GMS Academy, Davanagere

1. **Sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void quicksort(int a[], int l, int h)
{
   int i=l, j=h, pivot=a[(l+h)/2], t;
   while(i<=j)
 {
       while(a[i]<pivot) i++;
       while(a[j]>pivot) j--;
       if(i<=j) {
           t=a[i]; a[i]=a[j]; a[j]=t;
           i++; j--;
       }
   }
   if(l<j) quicksort(a,l,j);
   if(i<h) quicksort(a,i,h);
}
int main()
{
   int n,i; clock_t s,e;
   printf("Enter n: ");
   scanf("%d",&n);
   int a[n];
   for(i=0;i<n;i++) a[i]=rand()%1000;
   s=clock();
   quicksort(a,0,n-1);
   e=clock();
   printf("Sorted:\n");
   for(i=0;i<n;i++)
    printf("%d ",a[i]);
   printf("\nTime = %f sec",(double)(e-s)/CLOCKS_PER_SEC);
   return 0;
}
```

Output:

```
Enter n: 5
Sorted:
383 777 793 886 915
Time = 0.000002 sec
```

2. **Implement a Merge Sort algorithm to sort a given set of elements and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted. The elements can be read from a file or can be generated using the random number generator.**

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
void merge(int a[], int l, int m, int r)
 {
   int i=l, j=m+1, k=0, b[1000];
   while(i<=m && j<=r) b[k++] = (a[i]<a[j]) ? a[i++] : a[j++];
   while(i<=m) b[k++] = a[i++];
   while(j<=r) b[k++] = a[j++];
   for(i=0;i<k;i++) a[l+i]=b[i];
}
void mergesort(int a[], int l, int r)
 {
   if(l<r)
 {
      int m=(l+r)/2;
      mergesort(a,l,m);
      mergesort(a,m+1,r);
      merge(a,l,m,r);
   }
}
void main()
{
   int n,i,a[1000];
   clock_t start,end;
   clrscr();
   printf("Enter number of elements: ");
   scanf("%d",&n);
   randomize();
   for(i=0;i<n;i++) a[i]=rand()%1000;
   printf("\nUnsorted:\n");
   for(i=0;i<n;i++)
   printf("%d ",a[i]);
   start=clock();
   mergesort(a,0,n-1);
   end=clock();
   printf("\n\nSorted:\n");
   for(i=0;i<n;i++)
  printf("%d ",a[i]);
   printf("\n\nTime = %lf sec",(double)(end-start)/CLK_TCK);
   getch();
}
```

Output:

```
Enter number of elements: 5
Unsorted:
756 262 38 739 401
Sorted:
38 262 401 739 756
Time taken = 0.000003 sec
```

3. **Write a program to sort a list of N elements using Selection Sort Technique.**

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int a[50], n, i, j, min, temp;
   clrscr();
   printf("Enter number of elements: ");
   scanf("%d", &n);
   printf("Enter %d elements:\n", n);
   for(i=0; i<n; i++)
    scanf("%d", &a[i]);
   for(i=0; i<n-1; i++)
{
      min = i;
      for(j=i+1; j<n; j++)
         if(a[j] < a[min])
            min = j;
      temp = a[i];
      a[i] = a[min];
      a[min] = temp;
   }
   printf("\nSorted array:\n");
   for(i=0; i<n; i++)
      printf("%d ", a[i]);
   getch();
}
```

Output:

```
Enter number of elements: 5
Enter 5 elements:
10
30
20
50
40
Sorted array:
10 20 30 40 50
```

4. **Write a program to perform Travelling Salesman Problem.**

```
#include <stdio.h>
#include <conio.h>
#define MAX 10
#define INF 9999
int n, cost[MAX][MAX], visited[MAX], min = INF;
void tsp(int city, int count, int sum)
 {
   int i;
   if (count == n && cost[city][0] > 0)
 {
       if (sum + cost[city][0] < min)
           min = sum + cost[city][0];
       return;
   }
   for (i = 0; i < n; i++) {
       if (!visited[i] && cost[city][i] > 0)
 {
           visited[i] = 1;
           tsp(i, count + 1, sum + cost[city][i]);
           visited[i] = 0;
       }
   }
}
void main()
 {
   int i, j;
   clrscr();
   printf("Enter number of cities: ");
   scanf("%d", &n);
   printf("Enter cost matrix:\n");
   for (i = 0; i < n; i++)
       for (j = 0; j < n; j++)
           scanf("%d", &cost[i][j]);
   for (i = 0; i < n; i++)
       visited[i] = 0;
   visited[0] = 1;
   tsp(0, 1, 0);
   printf("\nMinimum cost = %d", min);
   getch();
}
```

Output:

```
Enter number of cities: 4
Enter cost matrix:
0 10 15 20
10 0 35 25
```

```
15 35 0 30
20 25 30 0
Minimum cost = 80
```

5. **Write a program to perform Knapsack Problem using Greedy Solution.**

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int n, i, j, cap, wt[10], pr[10], temp;
   float ratio[10], x[10], tp = 0;
   clrscr();
   printf("Enter number of items: ");
   scanf("%d", &n);
   printf("Enter knapsack capacity: ");
   scanf("%d", &cap);
   for(i = 0; i < n; i++) {
      printf("Profit and Weight of item %d: ", i+1);
      scanf("%d %d", &pr[i], &wt[i]);
      ratio[i] = (float)pr[i] / wt[i];
   }
   // Sort by ratio (descending)
   for(i = 0; i < n; i++)
 {
      for(j = i+1; j < n; j++)
{
         if(ratio[i] < ratio[j])
{
            temp = pr[i]; pr[i] = pr[j]; pr[j] = temp;
            temp = wt[i]; wt[i] = wt[j]; wt[j] = temp;
            ratio[i] = (float)pr[i] / wt[i];
         }
      }
   }
   for(i = 0; i < n; i++) x[i] = 0.0;
   int u = cap;
   for(i = 0; i < n; i++)
{
      if(wt[i] > u) break;
      x[i] = 1.0;
      tp += pr[i];
      u -= wt[i];
   }
   if(i < n)
{
      x[i] = (float)u / wt[i];
```

```
        tp += pr[i] * x[i];
    }
    printf("\nMaximum Profit = %.2f", tp);
    getch();
}
```

Output:

```
Enter number of items: 3
Enter knapsack capacity: 20
Profit and Weight of item 1: 25 15
Profit and Weight of item 2: 24 20
Profit and Weight of item 3: 12 35
Maximum Profit = 31.00
```

6. Write program to implement the DFS and BFS algorithm for a graph.

```
#include <stdio.h>
#include <conio.h>
#define MAX 10
int a[MAX][MAX], vis[MAX], n;
// DFS
void dfs(int v)
{
    int i;
    printf("%d ", v);
    vis[v] = 1;
    for(i = 0; i < n; i++)
        if(a[v][i] && !vis[i])
            dfs(i);
}
// BFS
void bfs(int v) {
    int q[MAX], f=0, r=0, i;
    vis[v] = 1;
    q[r] = v;
    while(f <= r)
 {
        v = q[f++];
        printf("%d ", v);
        for(i = 0; i < n; i++)
            if(a[v][i] && !vis[i])
                q[++r] = i, vis[i] = 1;
    }
}
void main()
 {
    int i,j,start;
    clrscr();
    printf("Enter no. of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);
    printf("Enter start vertex (0-%d): ", n-1);
    scanf("%d",&start);
    // BFS
    for(i=0;i<n;i++) vis[i]=0;
    printf("\nBFS: "); bfs(start);
    // DFS
    for(i=0;i<n;i++) vis[i]=0;
    printf("\nDFS: "); dfs(start);
    getch();
}
```

Output:

```
Enter no. of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter start vertex (0-3): 0

BFS: 0 1 2 3
DFS: 0 1 3 2
```

7. Write a program to find minimum and maximum value in an array using divide and conquer

```
#include <stdio.h>
void minmax(int a[], int low, int high, int *min, int *max)
{
    int mid, min1, max1, min2, max2;
    if (low == high)
  {
        *min = *max = a[low];
    }
  else if (high == low + 1)
  {
        if (a[low] < a[high])
  {
            *min = a[low];
            *max = a[high];
        }
else
```

```
 {
           *min = a[high];
           *max = a[low];
       }
   }
 else
 {
       mid = (low + high) / 2;
       minmax(a, low, mid, &min1, &max1);
       minmax(a, mid + 1, high, &min2, &max2);
       *min = (min1 < min2) ? min1 : min2;
       *max = (max1 > max2) ? max1 : max2;
   }
}
int main()
{
    int a[50], n, i, min, max;
    printf("Enter size of array: ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++)
     scanf("%d", &a[i]);
    minmax(a, 0, n - 1, &min, &max);
    printf("Minimum = %d\n", min);
    printf("Maximum = %d\n", max);
    return 0;
}
```

Output:

```
Enter size of array: 5
Enter 5 elements:
10
2
3
6
8
Minimum = 2
Maximum = 10
```

8. Write a test program to implement divide and conquer strategy: ex; quicksort algorithm for sorting list of integers in ascending order

```
#include <stdio.h>
#include <conio.h>
void swap(int *a, int *b)
{
   int t = *a; *a = *b; *b = t;
}
int partition(int a[], int low, int high)
{
   int pivot = a[high], i = low - 1, j;
   for (j = low; j < high; j++)
{
      if (a[j] <= pivot)
{
         i++;
         swap(&a[i], &a[j]);
      }
   }
   swap(&a[i+1], &a[high]);
   return i+1;
}
void quicksort(int a[], int low, int high)
 {
   if (low < high)
 {
      int pi = partition(a, low, high);
      quicksort(a, low, pi-1);
      quicksort(a, pi+1, high);
   }
}
void main()
{
   int i, n, a[20];
   clrscr();
   printf("Enter size of array: ");
   scanf("%d", &n);
   printf("Enter %d elements:\n", n);
   for (i=0; i<n; i++) scanf("%d", &a[i]);
   quicksort(a, 0, n-1);
   printf("Sorted array:\n");
   for (i=0; i<n; i++) printf("%d ", a[i]);
   getch();
}
```

Output:

```
Enter size of array: 5
Enter 5 elements:
10
60
30
20
50
Sorted array:
10 20 30 50 60
```

9. write a program to implement merge sort algorithm for sorting list of integers in ascending order

```
#include <stdio.h>
#include <conio.h>
void merge(int a[], int l, int m, int r)
{
   int i=l, j=m+1, k=0, temp[50];
   while(i<=m && j<=r) temp[k++] = (a[i]<a[j]) ? a[i++] : a[j++];
   while(i<=m) temp[k++] = a[i++];
   while(j<=r) temp[k++] = a[j++];
   for(i=l, j=0; i<=r; i++, j++) a[i]=temp[j];
}
void mergesort(int a[], int l, int r)
 {
   if(l<r)
 {
      int m=(l+r)/2;
      mergesort(a,l,m);
      mergesort(a,m+1,r);
      merge(a,l,m,r);
   }
}
void main()
{
   int a[50], n, i;
   clrscr();
   printf("Enter number of elements: ");
   scanf("%d",&n);
   printf("Enter elements:\n");
   for(i=0;i<n;i++) scanf("%d",&a[i]);
   mergesort(a,0,n-1);
   printf("Sorted array:\n");
   for(i=0;i<n;i++) printf("%d ",a[i]);
   getch();
}
```

Output:

```
Enter number of elements: 5
Enter elements:
20
60
10
50
40
Sorted array:
10 20 40 50 60
```