1BM22CS279

INDEX

Name Sindhuja Narasimhan Std Sec 4F

Roll No. Subject School/College

School/College Tel. No. Parents Tel. No.

Sl. No.	Date	Title	Page No.	Teacher Sign / Remarks
1.	03/05/24	Linear, Binary, Bubble Sort, Select Sort.	10	3/5/24
2	31/05/24	CRCD, TOH, Lomuto, TSTDFS.	10	7/6/24
3.	7/06/24	Merge Sort, Quick sort	10	
4.	13/06/24	Warshall's, Floyd's, Presorting, Knapsack Problem	10	21/6/24
5.	21/06/24	HorSpool's, Heap Sort.		
6.	05/07/24	Fractional Knapsack Kruskal's Dijkstra's Prims	10	5/7/24
7.	12/07/24	N-Queens.	10	19/7/24

03/05/24

Date __/__/__
Page _____

01 Linear Search

```
#include<stdio.h>
int linearsearch(int * arr, int size, int key)
{
  for(int i=0; i<size; i++){
    if(arr[i]==key){
      return i;
    }
  }
  return -1;
}
int main()
{
  int arr[10], n, i, key;
  printf("enter no of elements\n");
  scanf("%d", &n);
  printf("enter elements:\n");
  for(i=0; i<n; i++){
    scanf("%d", &arr[i]);
  }
  printf("Enter key");
  scanf("%d", &key);
  int index = linearsearch(arr, n, key);
  if(index==-1){
    printf("Element not found");
  }
  else{
    printf("Element is present at arr %d", index+1);
  }
  return 0;
}
```

Date / /
Page

O/p:

Case 1: Enter no of elements
4
Enter elements
1 3 5 6
Enter key 5
The element is present at 3 index.

Case 2: enter no of elements
4
enter elements
1 3 5 6
Enter key 7
Element not present.

⇒ Binary Search

```
#include <stdio.h>
int binarysearch(int array[], int x, int low, int high){
  while(low <= high)
    {
    int mid = low+(high-low)/2;
    if(array[mid] == x)
      return mid;
    if(array[mid] < x)
      low = mid+1;
    else
    high = mid-1;
  }
  return -1;
}
```

Date __/__/__
Page _____

```
int main(void){
int arr[10], n, key, i;
printf("enter no of elements \n");
scanf("%d", &n);
printf("enter elements");
for (i=0; i<n; i++) {
    scanf("%d", &arr[i]);
}
printf("Enter key");
scanf("%d", &key);
for (i=0; i<n; i++)
    {
    for (int j = i+1; j<n; j++)
    {
        int temp, arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
}
}
int result = binarySearch (arr, key, 0, n-1);
if (result == -1)
    printf("Not found");
else
    printf("Element is found at index %d", result);
return 0;
}
```

Op: enter no of elements
4
Enter elements
6 4 31 2
Enter key 2
Element found at 1.

Date / /
Page

case1: enter no of elements
4
enter elements
6 4 31 2
enter key 7
Element not found.

⇒ Bubble Sort.

```
#include <stdio.h>
void bubblesort(int arr[], int n){
int i, j;
for (i=0; i<n-1; i++){
  for (j=0; j<n-1-i; j++){
    if (arr[j] > arr[j+1]){
      int temp = arr[j];
      arr[j] = arr[j+1];
      arr[j+1] = temp;
    }
  }
}
void main(){
  int num;
  int i, arr[20];
  printf("Enter number of element : ");
  scanf("%d", &num);
  printf("Enter all element");
  for(i=0; i<num; i++)
  {
   scanf("%d", &arr[i]);
}
```

Date __/__/__
Page _____

```
bubblesort(arr, num)
printf("Sorted array");
for(i=0; i<num; i++)
  {
  printf("%d", arr[i])
  }
}
```

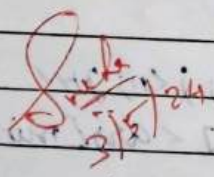
O/p: Enter number of elements: 4
Enter elements : 4 2 7 1
Sorted array : 1 2 4 7

=> Select Sort.

```
#include <stdio.h>
void swap(int *xp, int *yp){
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
void selectionSort(int arr[], int n){
   int i, j, min_idx;
   for(i=0; i<n; i++){
     min_idx = i;
     for(j=i+1; j<n; j++)
       if(arr[j] < arr[min_idx])
           min_idx = j;
     swap(&arr[min_idx], &arr[i]);
   }
}
void printArray(int arr[], int size){
   int i;
  for(i=0; i<size; i++)
     printf("%d", arr[i]);
     printf("\n");
}
```

```
int main()
{
  int arr[10], n, key, i;
  printf("enter no of elements");
  scanf("%d", &n);
  printf("enter elements");
  for(i=0; i<n; i++)
  {
    scanf("%d", &arr[i]);
  }
  selectionsort(arr, n);
  printf("Sorted array");
  PrintArray(arr, n);
  return 0;
}
```

O/p: enter no of elements
5
enter elements
2 3 6 1 7
Sorted array
1 2 3 6 7



30/05/24

Date __/__/__
Page _____

1. GCD of two nos:

```
#include<stdio.h>
int gcd(int a, int b){
    if(b==0)
        return a;
    return gcd(b, a%b);
}
int main(){
    int num1,num2;
    Printf("Enter two integers:");
    Scanf("%d %d", &num1, &num2);
    int result = gcd(num1, num2);
    Printf("GCD of %d and %d is %d\n", num1, num2,
                                        result);
    return 0;
}
```

O/P
```
Enter two integers
8 16
GCD of 8 and 16 is 8.
```

2. Towers of Hanoi

```
#include<stdio.h>
void moveDisk(int n, char source, char target){
    printf("Move disk %d from %c to %c", n,
                                source, target);
}
void toh(int n, char source, char target, char auxiliary)
{
    if(n==1){
        moveDisk(n, source, target);
        return;
    }
```

```
toh(n-1, source, auxiliary, target);
moveDisk(n, source, target);
toh(n-1, auxiliary, target, source);
}
int main() {
    int n;
    printf("Enter the no of disks: ");
    scanf("%d", &n);
    toh(n, 'A', 'C', 'B');
    return 0;
}
```

O/p: Enter the number of disks: 3
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C.

3 Computing Median & Quick sort.

```
#include<stdio.h>
void swap(int*, int*);
int computePartition(int a[], int, int);
int quickSelect(int[], int, int, int);
int main()
{
int a[10], l, u, n, i, res;
Printf("Enter the size");
scanf("%d", &n);
Printf("Enter the array elements:");
for(i=0; i<n; i++)
```

Date __/__/__
Page _____

```
    {
    swap("%d", &a[i]);
}
    res = quickSelect(a, 0, n-1, n/2);
    printf("Median is %d", res);
    return 0;
}
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int lomutoPartition(int a[], int l, int r) {
    int p, s, i;
    p = a[l];
    s = l;
    for(i = l+1; i < r; i++) {
        if(a[i] < p) {
            s = s+1;
            swap(&a[s], &a[i]);
        }
    }
    swap(&a[s], &a[l]);
    return s;
}
int quickSelect(int a[], int l, int r, int k) {
    int s = lomutoPartition(a, l, r);
    if(s == k) {
        return a[s];
    } else if(s < k) {
        quickSelect(a, s+1, r, k);
    } else {
        quickSelect(a, l, s-1, k);
    }
}
```

Date / /
Page

O/p Enter the size: 9
Enter the array elements: 4 1 10 6 7 12 9 12 15
Median is 7.

4. Enter the number of vertices: 4
Enter the adjacency matrix
0 1 1 0
0 0 0 1
0 0 0 1
0 0 0 0
Topological sort: 0 2 1 3.

5. TFS using source removal

Enter the number of vertices: 4
Enter the adjacency matrix
0 1 1 0
0 0 0 1
0 0 0 0
0 0 0 0
Topological sort: 0 1 3 2.

Date __/__/__
Page _____

01/06/24

1. Merge sort.

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
void merge(int a[], int l, int h, int mid){
  int i, j, k;
  i = l;
  j = mid+1;
  k = l;
  int b[MAX];
  while(i<=mid && j<=h)
  {
    if(a[i]<=a[j])
    {
      b[k] = a[i];
      i++;
      k++;
    }
    else{
      b[k] = a[j];
      j++;
      k++;
    }
  }
while(i<=mid)
  {
    b[k] = a[i];
    i++;
    k++;
  }
while(j<=h)
  {
```

```
    b[k] = a[j];
    j++;
    k++;
  }
  for(int v= l; v <= h; v++)
    {
      a[v] = b[v];
  }
}
void mergesort(int a[], int l, int h) {
    if (l<h)
      {
        int mid = (l+h)/2;
        mergesort (a, l, mid);
        mergesort (a, mid+1, h);
        merge(a, l, h, mid);
    }
}

int main() {
    int n;
    printf("Enter the size of the array:");
    scanf("%d", &n);
    if (n>MAX)
      {
        printf("the array is exceeded \n");
        return 1;
      }
  int a[MAX];
  for (int i =0; i<n; i++)
    {
    scanf("%d", &a[i]);
  }
```

```
int l=0;
int h=n-1;
mergesort(a, l, h);
printf("the sorted array is:");
for(int i=0; i<n; i++){
    printf("%d\t", a[i]);
}
}
```

O/p: Enter the size of Array: 4
Enter array elements
5 1 7 3
the sorted array is: 1 3 5 7

2. Quick Sort.

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

int part(int a[], int low, int high){
  int piv = a[low], i = low, j = high+1;
  while(i<j){
    do{
      i++;
    }while(a[i]<=piv && i<=high);
  do{
    --j;
  }while(a[j]>piv && j>=low);
  if(i<j){
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
  }
}
```

```
int temp = a[j];
a[j] = a[low];
a[low] = temp;
return j;
}

void quicksort(int a[], int low, int high){
if(low < high){
int mid = part(a, low, high);
quicksort(a, low, mid-1);
quicksort(a, mid+1, high);
}
}

int main(){
int a[100], n, i;
printf("Enter the no of elements: ");
scanf("%d", &n);
printf("Enter the elements:");
for(i=0; i<n; i++){
scanf("%d", &a[i]);
}
quicksort(a, 0, n-1);
printf("Sorted array");
for(i=0; i<n; i++){
printf("%d\t", a[i]);
}
return 0;
}
```

Date __/__/__
Page ____

Op Enter the number of elements: 8
Enter the elements: 2 6 8 3 9 1 7 4
Sorted array: 1 2 3 4 6 7 8 9

7/6/24

13/06/24

1. Warshall's algorithm

```
#include<stdio.h>
int a[10][10], p[10][10], n;
void warshall(int [][10], int);
int main()
{
  printf("Enter no of vertices");
  scanf("%d",&n);
  printf("Enter the adjacent matrix\n");
  int i,j;
  for(i=0; i<n; i++) {
        for(j=0; j<n; j++){
          scanf("%d", &a[i][j]);
    }
  }
  warshall(a,n);
  printf("Transitive Matrix ");
  for(i=0; i<n; i++) {
      for(j=0; j<n; j++) {
          printf("%d", p[i][j]);
    }
    return 0;
  }

void warshall(int a[][10], int n){
    int i,j,k;
    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            p[i][j] = a[i][j];
    }
  }
```

Date __/__/__
Page _____

```
for (k=0; k<n; k++){
    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            if (P[i][j]==0 && (P[i][k]==1
                && P[k][j]==1)){
                P[i][j]=1;
    }
}
}
}
```

Output:

Enter the no of vertices: 4

Enter the adjacent matrix:

0 1 0 0
0 0 0 1
0 0 0 0
1 0 1 0

Transitive Matrix:

1 1 1 1
1 1 1 1
0 0 0 0
1 1 1 1

Date / /
Page

2. Floyd's algorithm.

```
#include <stdio.h>
int a[10][10], D[10][10], n;
void floyd(int [][10], int);
int min(int, int);
int main()
{
 Printf("Enter the no of vertices");
 scanf("%d", &n);
 Printf("Enter the cost adjacency matrix \n");
  int i, j;
  for(i=0; i<n; i++){
      for(j=0; j<n; j++){
          scanf("%d", &a[i][j]);
      }
   }
  floyd(a, n);
  Printf("Distance Matrix \n");
  for(i=0; i<n; i++){
     for(j=0; j<n; j++){
        Printf("%d", D[i][j]);
  }
 }
  return 0;
}
void floyd(int a[][10], int n){
    int i, j, k;
    for(i=0; i<n; i++){
       for(j=0; j<n; j++){
          D[i][j]=a[i][j];
    }
  }
```

Date __/__/__
Page _____

```
for (k=0; k<n; k++){
    for (i=0; i<n; i++){
        for (j=0; j<n; j++){
            D[i][j] = min(D[i][j], D[i][k]+D[k][j]);
        }
    }
}
}

int min(int a, int b){
    if (a<b){
        return a;
    }else{
        return b;
    }
}
```

O/p: Enter the no. of vertices 4
Enter the cost adjacency matrix:

0	99	3	99
2	0	99	99
99	6	0	1
7	99	99	0

Distance Matrix:

0	9	3	4
2	0	5	6
8	6	0	1
7	16	10	0

Date __/__/__
Page _____

3. Presorting

```
#include <stdio.h>
int a[20], n;
void simple_sort (int [], int, int, int);
void merge_sort (int [], int, int);

int main()
{
    int i, result;
    printf("Enter the no. of elements ");
    scanf("%d", &n);
    printf("Enter the array elements: ");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    result = presorting(a, n);
    if(result == 1)
        printf("Unique Array elements ");
    else
        printf("Duplicate Array elements ");
    return 0;
}
void merge_sort(int a[], int low, int high)
{
    if(low < high)
    {
        int mid = (low + high)/2;
        merge_sort(a, low, mid);
        merge_sort(a, mid+1, high);
        simple_sort(a, low, mid, high);
    }
}
```

Date __/__/__
Page _____

```
void simple_sort(int a[], int low, int mid, int
                                        high){
    int i=low, j=mid+1, k=low;
    int c[n];
    while(i<=mid && j<=high){
        if(a[i]<a[j]){
            c[k++]=a[i];
            i++;
        }else{
            c[k++]=a[j];
            j++;
        }
    }
    while(i<=mid){
        c[k++]=a[i];
        i++;
    }
}
int presorting(int a[], int n){
    merge_sort(a, 0, n-1);
    int i;
    for(i=0; i<=n-2; i++){
        if(a[i]>=a[i+1]){
            return 0;
        }
    }
    return 1;
}
```

Date / /
Page

Output:

Case 1: Enter the no. of elements: 8
Enter the array elements:
25 03 96 78 65 10 65 25
Duplicate Array elements

Case 2: Enter the no of elements: 5
Enter the array elements: 12 56 98 74 32
Unique Array elements.

4. Knapsack Problem.

```
#include<stdio.h>
int n, m, w[10], p[10], v[10][10];
int main()
{
 int i, j;
 Printf("Enter the no. of items");
 Scanf("%d", &n);
 Printf("Enter the capacity of knapsack");
 Scanf("%d", &m);
 Printf("Enter weights:");
 printf("Enter profits");
  for(i=0; i<n; i++)
      Scanf("%d", &p[i]);
 }.
 Knapsack(n, m, w, p);
 printf("Optimal Solution");
 for(i=0; i<n; i++)
     for(j=0; j<n; j++)
         Printf("%d", v[i][j]);
  }
}
```

Date __/__/__
Page

```
void knapsack (int n, int m, int w[], int p[]) {
  int i, j;
  for (i=0; i<n; i++) {
    for (j=0; j<m; j++) {
      if (i==0 || j==0) {
        v[i][j] = 0;
      }
      else if (w[i] > j) {
        v[i][j] = v[i-1][j];
      } else {
        v[i][j] = max(v[i-1][j], (v[i-1][j-w[i]]
                                  + p[i]));
      }
    }
  }
}
int max (int a, int b) {
  if (a > b) {
    return a;
  } else {
    return b;
  }
}
```

Output:

Enter the no of items: 4
Enter the capacity of knapsack: 5
Enter weights: 2 1 3 2
Enter profits: 12 10 20 15
Optimal Sol: 0 0 0 0
0 10 10 10
0 10 10 20
0 10 15 25

Date __/__/__
Page _____

21/06/24

1 Horspool's

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

char P[100], T[100];
int S[200];

int main(){
    printf("Enter the text string: ");
    fgets(T, sizeof(T), stdin);
    T[strcspn(T, "\n")] = 0;
    printf("Enter the pattern string: ");
    fgets(P, sizeof(P), stdin);
    P[strcspn(P, "\n")] = 0;
    int result = horSpool(T, P);
    if (result == -1){
        printf("Pattern not found\n");
    } else {
        printf("Pattern found at position: %d\n", result
    }
    return 0;
}

void ShiftTable(char P[], int S[]){
    int m = strlen(P);
    int i;
    for(i=0; i<100; i++){
        S[i] = m;
    }
    for(i=0; i<m-1; i++){
        S[(int)P[i]] = m-1-i;
    }
}
```

Date __/__/__
Page _____

```
int horSpool(char T[], char P[]){
    shiftTable(P, S);
    int n = strlen(T);
    int m = strlen(P);
    int i = m-1;
    while(i<=n-1){
        int k = 0;
        while(k<=m-1 && T[i-k]==P[m-1-k]){
            k++;
        }
        if(k==m){
            return i-m+1;
        }
        i = i+S[(int)T[i]];
    }
    return -1;
}
```

Op: Enter the text string jim saw me in barber shop
Enter the pattern string: barber
Pattern found at position : 16.

Date __/__/__
Page _____

2. Heap Sort.

```
#include<stdio.h>
int a[10], n;
int main(){
    printf("Enter the no of array element:");
    scanf("%d",&n);
    int i;
    printf("Enter array element ");
    for(i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    heapify(a,n);
    printf("Array elements ");
    for(i=0;i<n;i++){
        printf("%d",a[i]);
    }
    return 0;
}
void heapify(int a[], int n){
    int k;
    for(k=1;k<n;k++){
        int key=a[k];
        int c=k;
        int p=(c-1)/2;
        while(c>0 && key>a[p]){
            a[c]=a[p];
            c=p;
            p=(c-1)/2;
        }
        a[c]=key;
    }
}
```

Date __/__/__
Page _____

Op Enter the number of array elements : 5
Enter array elements : 7 9 1 6 2
Array elements : 9 7 1 6 2.

21/6/24

05/07/24

Date __/__/__
Page _____

1 Fractional knapsack Pbo

```
#include <stdio.h>
#define MAX 100
#define MAXWEIGHT 100

typedef struct {
    int weight;
    int cost;
    float ratio;
} Item;

void fKMaxProfit(Item items[], int n, int capacity) {
    int i, w;
    float totalCost = 0.0;
    for (i=0; i<n; i++) {
        for (int j = i+1; j<n; j++) {
            if (items[i].cost < items[j].cost) {
                Item temp = items[i];
                items[i] = items[j];
                items[j] = temp;
            }
        }
    }
    printf("Total cost (MaxProfit) : %.2f\n", totalCost);
}

void fractionalKnapsackMinWeight(Item items[], int n,
                                 int capacity) {
    int i, w;
    float totalCost = 0.0;
```

```
for(i=0; i<; i++) {
  if(items[i].weight <= capacity) {
      capacity -= items[i].weight;
      totalcost += item[i].cost;
  } else {
      float fraction = (float) capacity/items[i].weight;
      totalcost += items[i].cost * fraction;
      break;
  }
}
Printf("Total Cost (Min weight): %f\n", totalcost);

void fractionalKnapsackProfitByWeightRatio(Item items[],
                                           int n, int capacity) {
  int i, u;
  float totalCost = 0.0;
  for(i=0; i<n; i++) {
      if(items[i].weight <= capacity) {
          capacity -= items[i].weight;
          totalcost += items[i].cost;
      } else {
          float fraction = (float) capacity/items[i].weight;
          totalCost += items[i].cost * fraction;
          break;
      }
  }
int main() {
  int n, capacity;
  Item items[MAX_ITEMS];
  printf("Enter the number of items: ");
  scanf("%d", &n);
  printf("Enter the capacity of knapsack");
  scanf("%d", &capacity);
```

Date __/__/__
Page _____

```
fractionalKnapsackMaxProfit(items, n, capacity);
fractionalKnapsackMinWeight(items, n, capacity);
fractionalKnapsack(items, n, capacity);

return 0;
}
```

O/p:

```
Enter the number of items: 7
Enter the capacity of the knapsack: 15
Enter the cost and weight of each item
15 1
10 3
15 5
7 4
8 1
9 3
4 2
```

```
Total cost(MaxProfit): 47.25
Total cost(MinWeight): 46.00
Total cost(P/W ratio): 51.
```

Date __/__/__
Page _____

2 Prims Algorithm:

```
#include <stdio.h>
int cost[10][10], n, t[10][2], sum;
void prims(int cost[10][10], int n);
int main()
{
  int i,j;
  printf("Enter the number of vertices");
  scanf("%d", &n);
  printf("Enter cost adjacency matrix");
  scanf("%d", &cost[i][j]);
  }
}

prims(cost, n);
printf("Edges of minimal spanning tree");
for(i=0; i<n; i++){
    printf("(%d %d)", t[i][0], t[i][1]);
}
return 0;
}
void prims(int cost[10][10], int n){
   int i, j, u, v;
   int min, source;
   int d[10], a[10], s[10];
   min = 999;
   sour = 0;

   s[source] = 1;
   sum = 0;
   int k = 0;
   for(i=0; i<n-1; i++){
     for(j=0; j<n; j++){
       if(s[j]==0 && d[j]<min){
```

```
            min=d[j];
            u=j;
        }
    }
    if(u!=-1){
        t[k][0]=u;
        t[k][1]=p[u];
        k++;
        sum+=cost[u][p[u]];
        s[u]=1;
        for(v=0; v<n; v++){
            if(s[v]==0 && cost[u][v]<d[v]){
                d[v]=cost[u][v];
                p[v]=u;
            }
        }
    }
    }
}
```

O/p: Enter the number of vertices: 6
Enter the cost adjacency matrix:

```
0 3 0 0 6 5
3 0 1 0 0 4
0 1 0 6 0 4
0 0 6 0 0 4
0 0 6 0 8 5
6 0 0 6 0 2
```

Edges of min spanning tree: (2,0) (3,0) (1,3) (4,2) (5,4)

Date __/__/__
Page _____

3. Kruskal's Algorithm

```
#include<stdio.h>
int cost[20][20], n, t[20][2], sum;
int main()
{
  int i, j;
  printf("Enter no of vertices");
  scanf("%d", &n);
  kruskal(cost, n);
  printf("Edges of min");
  for(i=0; i<n; i++)
  {
    printf("%d-%d", t[i][0], t[i][1]);
  }
  return 0;
}
void kruskal(int cost[20][20], int n)
{
  int min, u, v, count, k;
  int parent[20];
  k=0;
  sum=0;
  for(int i=0; i<n; i++)
  {
    parent[i]=i;
  }
  count=0;
  while(count<n-1)
  {
    min=999;
    for(int i=0; i<n; i++)
    {
      for(int j=0; j<n; j++)
      {
        if(find(parent, i)!=find(parent, j) &&
           cost[i][j]<min)
        {
          min=cost[i][j];
          u=i;
          v=j;
```

```
int root_u = find(parent, u);
int root_v = find(parent, v);
if (root_u != root_v)
    parent[root_u] = root_v;
    t[k][0] = u;
    count++;
}
}
}
```

O/P: Enter the number of vertices: 4
Enter the cost adjacency mat

```
0 1 2 0
1 0 3 0
2 3 0 1
0 0 1 0
```

Edges of minimal spanning tree
(0, 3) (1, 3) (2, 3).

5/7/24

Date __/__/__
Page _____

4. Dijkstra's

```
#include<stdio.h>
void dijkstra(int n, int s, int c[10][10])
  for(i=0; i<n; i++){
    d[i] = int_max;
    visited[i] = 0;
    p[i] = -1;
}
  d[s] = 0;
  for(i=0; i<n-1; i++){
    min = int_max;
    for(j=0; j<n; j++){
      if(visited[j] == 0 && d[j] < min){
        min = d[j];
        u = j;
      }
    }
    visited[u] = 1;
    for(v=0; v<n; v++){
      if(!visited[v] && c[u][v] && d[u] !=
                                  int max)
      d[v] = d[u] + c[u][v];
      p[v] = u;
    }
  }
}

for(i=0; i<n; i++){
  if(i != s){
    printf("Shortest path from %d to %d is %d", s, i,
                                        d[i]);
  }
}
}
```

Date / /
Page

12/07/24

1.

```
int main()
{
  int n, s, c[10][10], i, j;
  printf("Enter the no of vertices");
  scanf("%d", &n);
  printf("Enter the adjacency matrix");
  for(i=0; i<n; i++)
  {
    for(j=0; j<n; j++)
    {
      scanf("%d", &c[i][j]);
    }
  }
  printf("Enter the source vertex");
  scanf("%d", &s);
  dijkstra(n, s, c);
  return 0;
}
```

O/P:

```
Enter the number of vertices: 5
Enter the adjacency matrix:
0 3 0 4 0
3 0 4 2 0
0 4 0 5 6
7 2 5 0 4
0 0 6 4 0
Enter the source vertex: 0
Shortest path from 0 to 1 is 3
Shortest path from 0 to 2 is 7
shortest path from 0 to 3 is 5
Shortest path from 0 to 4 is 9
```

Date __/__/__
Page _____

2/01/24

1. N-Queen's Problem.

```
#include<stdio.h>
#include<stdbool.h>
bool place(int[], int);
void printSolution(int[], int);
void nQueens(int);
int main()
{
 int n;
 printf("Enter the no of queens: ");
 scanf("%d", &n);
 nQueens(n);
 return 0;
}
void nQueens(int n){
 int x[10];
 int count = 0;
 int k = 1;
 while(k != 0){
  x[k] = x[k]+1;
 while (x[k] <= n && !place(x,k)){
    x[k] = x[k]+1;
 }
  if (x[k] <= n){
   if (k == n){
    printSolution(x, n);
    printf("Solution found\n");
     count++;
  } else{
    k++;
   x[k] = 0;
  }
```

```
}
else {
  k++;
  x[k]=0;
}} else {
  k++;
  x[k]=0;
}}
  else {
    k--;
}
}
printf("Total Solutions: %d\n", count);
}
bool place(int x[10], int k){
  int i;
  for(i=1; i<k; i++){
  if((x[i]==x[k] || (i-x[i]==k-x[k]) ||
       (i+x[i]==k+x[k]))){
  return false;
}
}
  return true;
}
void printSolution(int x[10], int n){
  int i;
  for(i=1; i<=n; i++){
  printf("%d", x[i]);
}
  printf("\n");
}
```

Date __/__/__
Page _____

O/P: Enter the number of queens: 6

2 4 6 1 3 5

Solution found

3 6 2 5 1 4

Solution found

4 1 5 2 6 3

Solution found

5 3 1 6 4 2

Solution found

Total Solutions : 4

19/2/24