1. mergesort

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void merge(int *a, int l, int m, int h) {
   int i = l, j = m+1, k = 0, *b = malloc((h-l+1)*sizeof(int));
   while(i <= m && j <= h) b[k++] = a[i] <= a[j] ? a[i++] : a[j++];
   while(i <= m) b[k++] = a[i++];
   while(j <= h) b[k++] = a[j++];
   for(i = l, k = 0; i <= h; i++) a[i] = b[k++];
   free(b);
}
void sort(int *a, int l, int h) {
   if(l < h) {
      int m = (l+h)/2;
      sort(a, l, m);
      sort(a, m+1, h);
      merge(a, l, m, h);
   }
}
int main() {
   int n; scanf("%d", &n);
   int *a = malloc(n*sizeof(int));
   for(int i=0; i<n; i++) a[i] = rand()%100;
   clock_t s = clock();
   sort(a, 0, n-1);
   clock_t e = clock();
   for(int i=0; i<n; i++) printf("%d ", a[i]);
   printf("\n%.3e ms\n", (double)(e-s)*1000/CLOCKS_PER_SEC);
   free(a);
   return 0;
```

```
}
```

2. quicksort

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void qs(int *a,int l,int h){
   if(l<h){
      int p=a[l],i=l+1,j=h,t;
      while(1){
         while(i<=h && a[i]<=p) i++;
         while(a[j]>p) j--;
         if(i>=j) break;
         t=a[i];a[i]=a[j];a[j]=t;
      }
      t=a[l];a[l]=a[j];a[j]=t;
      qs(a,l,j-1);
      qs(a,j+1,h);
   }
}
int main(){
   int n,*a; clock_t s,e;
   scanf("%d",&n);
   a=malloc(n*sizeof(int));
   for(int i=0;i<n;i++) a[i]=rand()%100;
   s=clock();
   qs(a,0,n-1);
   e=clock();
   for(int i=0;i<n;i++) printf("%d ",a[i]);
   printf("\nTime: %.3e ms\n",(double)(e-s)*1000/CLOCKS_PER_SEC);
   free(a);
   return 0;
```

```
}
```

3. dfs

```
#include <stdio.h>
int a[10][10], visited[10], res[10], n, idx;
void DFS(int u) {
   visited[u] = 1;
   for(int v=0; v<n; v++)
      if(a[u][v] && !visited[v])
         DFS(v);
   res[idx++] = u;
}
int main() {
   scanf("%d", &n);
   for(int i=0; i<n; i++)
   for(int j=0; j<n; j++)
      scanf("%d", &a[i][j]);
    idx = 0;
   for(int i=0; i<n; i++)
      if(!visited[i]) DFS(i);
    for(int i=idx-1; i>=0; i--)
      printf("%d ", res[i]);
   return 0;
}
```

4a.floyd's

```
#include <stdio.h>
#define INF 999
int main() {
   int a[10][10], n;
   scanf("%d", &n);
```

```
    for(int i=0; i<n; i++)
    for(int j=0; j<n; j++)
    scanf("%d", &a[i][j]);
    for(int k=0; k<n; k++)
    for(int i=0; i<n; i++)
       for(int j=0; j<n; j++)
         if(a[i][k] + a[k][j] < a[i][j])
           a[i][j] = a[i][k] + a[k][j];
    for(int i=0; i<n; i++) {
    for(int j=0; j<n; j++)
       printf("%d ", a[i][j]);
     printf("\n");
    }
    return 0;
}
```

4b. warshall's

```
#include <stdio.h>
int a[10][10], n;
void warshall() {
    for (int k = 0; k < n; k++)
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                a[i][j] = a[i][j] || (a[i][k] && a[k][j]);
}

int main() {
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
```

```
            scanf("%d", &a[i][j]);
    warshall();
    printf("Transitive closure:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            printf("%d ", a[i][j]);
        printf("\n");
    }
    return 0;
}
```

5. prims

```
#include <stdio.h>
#define INF 999
int main() {
    int n, cost[10][10], vis[10] = {0}, i, j, u, v, min, total = 0, edges = 0;
    printf("Enter number of nodes: ");
    scanf("%d", &n);
    printf("Enter cost matrix (999 = no edge):\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &cost[i][j]);
    vis[0] = 1;
    printf("\nMST Edges:\n");
    while (edges < n - 1) {
        min = INF;
        for (i = 0; i < n; i++) {
            if (vis[i]) {
                for (j = 0; j < n; j++) {
                    if (!vis[j] && cost[i][j] < min) {
                        min = cost[i][j];
```

```
                u = i;
                v = j;
            }
        }
    }
    printf("(%d - %d) Cost: %d\n", u, v, min);
    total += min;
    vis[v] = 1;
    edges++;
  }
  printf("Total Cost: %d\n", total);
  return 0;
}
```

6. kruskal

```
#include <stdio.h>
#define INF 999
int parent[10];
int find(int i) {
  while (parent[i] != -1) i = parent[i];
  return i;
}
void kruskal(int a[10][10], int n) {
  int count = 0, mincost = 0;
  for (int i = 0; i < n; i++) parent[i] = -1;
  while (count < n - 1) {
    int min = INF, u, v;
    for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
        if (a[i][j] < min) min = a[i][j], u = i, v = j;
```

```
        int x = find(u), y = find(v);
        if (x != y) {
            printf("%d - %d : %d\n", u, v, min);
            parent[y] = x;
            mincost += min;
            count++;
        }
        a[u][v] = a[v][u] = INF;
    }
    printf("Min Cost: %d\n", mincost);
}
int main() {
    int a[10][10], n;
    printf("Vertices: ");
    scanf("%d", &n);
    printf("Cost matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) {
            scanf("%d", &a[i][j]);
            if (a[i][j] == 0) a[i][j] = INF;
        }
    kruskal(a, n);
    return 0;
}
```

7. dijkstra

```c
#include<stdio.h>
#define INF 999
int main(){
 int n,i,j,count=0,u,min,cost[10][10],dist[10],visited[10];
 printf("Number of vertices:");
 scanf("%d",&n);
 printf("Enter cost matrix (999 for no edge):\n");
 for(i=0;i<n;i++)for(j=0;j<n;j++)scanf("%d",&cost[i][j]);
 int source;
 printf("Source vertex (0 to %d):",n-1);
 scanf("%d",&source);
 for(i=0;i<n;i++){dist[i]=cost[source][i];visited[i]=0;}
 dist[source]=0;visited[source]=1;
 while(count<n-1){
  min=INF;u=-1;
  for(i=0;i<n;i++)if(!visited[i]&&dist[i]<min){min=dist[i];u=i;}
  if(u==-1)break;
  visited[u]=1;
  for(i=0;i<n;i++)
   if(!visited[i]&&cost[u][i]!=INF&&dist[u]+cost[u][i]<dist[i])dist[i]=dist[u]+cost[u][i];
  count++;
 }
 printf("Shortest distances from vertex %d:\n",source);
 for(i=0;i<n;i++)
  if(dist[i]==INF)printf("No path to %d\n",i);
  else printf("Distance to %d=%d\n",i,dist[i]);
 return 0;
}
```

8. knapsack

```c
#include <stdio.h>

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

void knapsack(int n, int w, int wt[], int val[]) {
    int dp[10][10] = {0};

    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= w; j++)
            if (wt[i - 1] <= j)
                dp[i][j] = max(val[i - 1] + dp[i - 1][j - wt[i - 1]], dp[i - 1][j]);
            else
                dp[i][j] = dp[i - 1][j];

    printf("Maximum value: %d\n", dp[n][w]);
}

int main() {
    int n, w, wt[10], val[10];
    printf("Enter number of items and knapsack capacity: ");
    scanf("%d %d", &n, &w);
    printf("Enter weights:\n");
    for (int i = 0; i < n; i++) scanf("%d", &wt[i]);
    printf("Enter values:\n");
    for (int i = 0; i < n; i++) scanf("%d", &val[i]);
    knapsack(n, w, wt, val);
    return 0;
}
```

9. nqueens

```c
#include<stdio.h>
int x[10],n;

int place(int r,int c){
 for(int i=1;i<r;i++)
  if(x[i]==c || abs(x[i]-c)==abs(i-r)) return 0;
 return 1;
}

void nqueens(int r){
 if(r>n){
  for(int i=1;i<=n;i++) printf("(%d,%d) ",i,x[i]);
  printf("\n"); return;
 }
 for(int c=1;c<=n;c++)
  if(place(r,c)){ x[r]=c; nqueens(r+1); }
}

int main(){
 printf("Enter n: ");
 scanf("%d",&n);
 nqueens(1);
 return 0;
}
```

10. sum of subsets

```c
#include <stdio.h>

int w[10], x[10], d, count = 0;

void subsetSum(int s, int k, int n) {
    if (s == d) {
        printf("Subset %d: ", ++count);
        for (int i = 0; i < k; i++)
            if (x[i]) printf("%d ", w[i]);
        printf("\n");
        return;
    }
    for (int i = k; i < n; i++) {
        if (s + w[i] <= d) {
            x[i] = 1;
            subsetSum(s + w[i], i + 1, n);
            x[i] = 0;
        }
```

```
    }
}
int main() {
    int n, sum = 0;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter elements (in increasing order): ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &w[i]);
        sum += w[i];
    }
    printf("Enter the required sum: ");
    scanf("%d", &d);

    if (sum < d) printf("No subset possible\n");
    else subsetSum(0, 0, n);
    return 0;
}
```