**BFS**

#include<stdio.h>

int g[20][20], q[20], color[20], prev[20], d[20];

int n, i, j, front = 0, rear = 0;

void BFS();

void enqueue(int s);

int dequeue();

int main()

{

while(scanf("%d", &n) != EOF)

{

printf("Graph Input\n");

for(i = 0; i < n; i++)

for(j = 0; j < n; j++)

scanf("%d", &g[i][j]);

for(i = 0; i < n; i++)

{

color[i] = 0;

prev[i] = -1;

d[i] = -1;

}

BFS();

}

return 0;

}

void BFS()

{

int s;

printf("Enter the starting node: ");

scanf("%d", &s);

color[s] = 1;

prev[s] = 0;

d[s] = 0;

enqueue(s);

while(front != rear)

{

int u;

u = dequeue();

for(i = 0; i < n; i++)

{

if(g[u][i] == 1)

{

if(color[i] == 0)

{

color[i] = 1;

prev[i] = u;

d[i] = d[u] + 1;

enqueue(i);

}

}

}

color[u] = 2;

}

printf("\n\nColor:");

for(i = 0; i < n; i++)

printf(" %d", color[i]);

printf("\n\nPrev:");

for(i = 0; i < n; i++)

printf(" %d", prev[i]);

printf("\n\nDistance:");

for(i = 0; i < n; i++)

printf(" %d", d[i]);

printf("\n\nStep for Source to C node: %d", d[2]);

}

void enqueue(int i)

{

q[rear] = i;

rear++;

}

int dequeue()

{

int v;

v = q[front];

front++;

return v;

}

**DFS & TOPOLOGICAL SORT**

#include<iostream>

#include<algorithm>

using namespace std;

int n, i, j, g[10][10], color[10], prev[10], d[10], t, f[10], zz[10], x = 0;

void DFS\_visit(int u);

void DFS()

{

t = 0;

for(i = 0; i < n; i++)

if(color[i] == 0)

DFS\_visit(i);

}

void DFS\_visit(int u)

{

color[u] = 1;

t++;

d[u] = t;

int v;

for(v = 0; v < n; v++)

{

if(g[u][v] == 1)

{

if(color[v] == 0)

{

prev[v] = u;

DFS\_visit(v);

}

}

}

zz[x] = u;

x++;

color[u] = 3;

t++;

f[u] = t;

}

int main()

{

scanf("%d", &n);

printf("Graph Input\n");

for(i = 0; i < n; i++)

for(j = 0; j < n; j++)

scanf("%d", &g[i][j]);

for(i = 0; i < n; i++)

{

color[i] = 0;

prev[i] = -1;

d[i] = -1;

}

DFS();

printf("\n\nColor: ");

for(i = 0; i < n; i++)

printf("%c = %d, ", 97+i,color[i]);

printf("\n\nDistant: ");

for(i = 0; i < n; i++)

printf("%c = %d, ", 97+i, d[i]);

printf("\n\nPrev: ");

for(i = 0; i < n; i++)

printf("%c => %d, ", 97+i, prev[i]);

printf("\n\nFinishing time: ");

for(i = 0; i < n; i++)

printf("%c => %d, ", 65+i, f[i]);

printf("\n\nTopological sort: ");

for(i = n-1; i >= 0; i--)

printf("%c, ", 65+zz[i]);

return 0;

}

**Quick Sort**

#include<iostream>

#include<algorithm>

using namespace std;

int a[100];

void QuickSort(int left, int right);

int partition(int left, int right);

int main()

{

int n, i;

printf("Enter number of elements: ");

scanf("%d", &n);

for(i = 0; i < n; i++)

scanf("%d", &a[i]);

QuickSort(0, n-1);

printf("Here's the output: ");

for(i = 0; i < n; i++)

printf("%d ", a[i]);

return 0;

}

void QuickSort(int left, int right)

{

if(left < right)

{

int pivot = partition(left, right);

QuickSort(left, pivot-1);

QuickSort(pivot+1, right);

}

}

int partition(int left, int right)

{

int i = left-1, p, j;

p = a[right];

for(j = left; j < right; j++)

{

if(a[j] <= p)

{

i = i + 1;

swap(a[i], a[j]);

}

}

swap(a[i+1], a[right]);

return (i+1);

}

**Merge Sort**

#include<iostream>

#include<algorithm>

#include<cmath>

using namespace std;

int a[100];

void Merge\_Sort(int left, int right);

void Merge(int left, int center, int right);

int main()

{

int n, i;

printf("Enter number of elements: ");

scanf("%d", &n);

for(i = 1; i <=n; i++)

scanf("%d", &a[i]);

Merge\_Sort(1, n);

for(i = 1; i < n+1; i++)

printf("%d ", a[i]);

return 0;

}

void Merge\_Sort(int left, int right)

{

if(left < right)

{

int center = (left+right) / 2;

Merge\_Sort(left, center);

Merge\_Sort(center + 1, right);

Merge(left, center, right);

}

}

void Merge(int left, int center, int right)

{

int n1 = center - left + 1; // To know the number of element

int n2 = right - center;

int L[n1+1], R[n2+1];

for(int i = 1; i <= n1; i++)

L[i] = a[left+i-1];

for(int i = 1; i <= n2; i++)

R[i] = a[center + i];

L[n1+1] = 12345;

R[n2+1] = 12345;

int i = 1, j = 1;

for(int k = left; k <= right; k++)

{

if(L[i] <= R[j])

{

a[k] = L[i];

i++;

}

else

{

a[k] = R[j];

j++;

}

}

}

**Greedy-Coin Change**

#include<iostream>

#include<algorithm>

#include<cmath>

using namespace std;

int main()

{

int n, a[n];

printf("Enter the number of note: ");

scanf("%d", &n);

printf("Enter notes: ");

for(int i = 0; i < n; i++)

scanf("%d", &a[i]);

sort(a, a+n);

int tk;

printf("\nEnter the amount: ");

scanf("%d", &tk);

printf("\n\n\n");

int note, count = 0;

int i = n-1;

while(tk > 0)

{

note = tk / a[i];

tk = tk % a[i];

if(note != 0)

printf("RS %d --> %d\n\n", a[i], note);

i--;

count += note;

}

printf("Total Note: %d\n", count);

return 0;

}