Lecture 20 Backtracking

1. Quick Sort

#include <iostream>

using namespace std;

int partition(int arr[], int s, int e)

{

int i = s-1;

int j = s;

int pivot = arr[e];

for(j=s; j<e; j++)

{

if(arr[j] <= pivot)

{

i++; // expanding samller part

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

}

}

swap(arr[i+1], arr[e]);

return i+1;

}

void quick\_sort(int arr[], int s, int e)

{

if(s>=e)

{

return;

}

int p = partition(arr, s, e);

quick\_sort(arr, s, p-1); // left part

quick\_sort(arr, p+1, e); // right part

}

int main() {

int n;

cin>>n;

int arr[n];

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

{

cin>>arr[i];

}

quick\_sort(arr, 0, n-1);

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

{

cout<<arr[i]<<", ";

}

return 0;

}

1. Counting Sort

#include <iostream>

using namespace std;

void counting\_sort(int arr[], int n)

{

int largest = -1;

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

{

largest = max(largest, arr[i]);

}

int \*freq = new int[largest+1] {0};

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

{

freq[arr[i]]++;

}

// putting element back into array

int j = 0;

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

{

while(freq[i]>0)

{

arr[j] = i; // copy

freq[i]--; // decre

j++; // index

}

}

}

int main() {

int arr[] = {4, 3, 3, 1, 10, 30, 2, 8};

int n = sizeof(arr)/sizeof(int);

counting\_sort(arr, n);

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

{

cout<<arr[i]<<", ";

}

return 0;

}

1. Rat In A Maze

#include <iostream>

using namespace std;

bool ratInAMaze(char maze[10][10], int soln[10][10], int i, int j, int m, int n)

{

// base case

if(i==m-1 && j==n-1)

{

soln[i][j] = 1;

// print path

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

{

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

{

cout<<soln[i][j]<<" ";

}

cout<<endl;

}

cout<<endl;

return true;

}

// rat should be inside the grid

if(i > m || j>n)

{

return false;

}

// valid path only

if(maze[i][j]=='X')

{

return false;

}

soln[i][j] = 1;

// recursive case

bool rightSuccess = ratInAMaze(maze, soln, i, j+1, m, n);

bool downSuccess = ratInAMaze(maze, soln, i+1, j, m, n);

// backtrack

soln[i][j] = 0;

if(rightSuccess || downSuccess)

{

return true;

}

return false;

}

int main() {

char maze[10][10] = {

"0000",

"00X0",

"000X",

"0X00",

};

int soln[10][10] = {0};

int m = 4, n = 4;

bool ans = ratInAMaze(maze, soln, 0, 0, m, n);

if(ans==false)

{

cout<<"path doesnt exist";

}

return 0;

}