Statement of the problem **:** Implementing 16 puzzle problem using branch and bound.

branch and bound(puzzle)

input : Initial configuration of the puzzle with one misplaced position.

output : goal configuration of the puzzle with all numbers in their respective places.

Step 0 : Possible movements (left ,right ,up ,down) of the empty box in the initial configuration are checked.

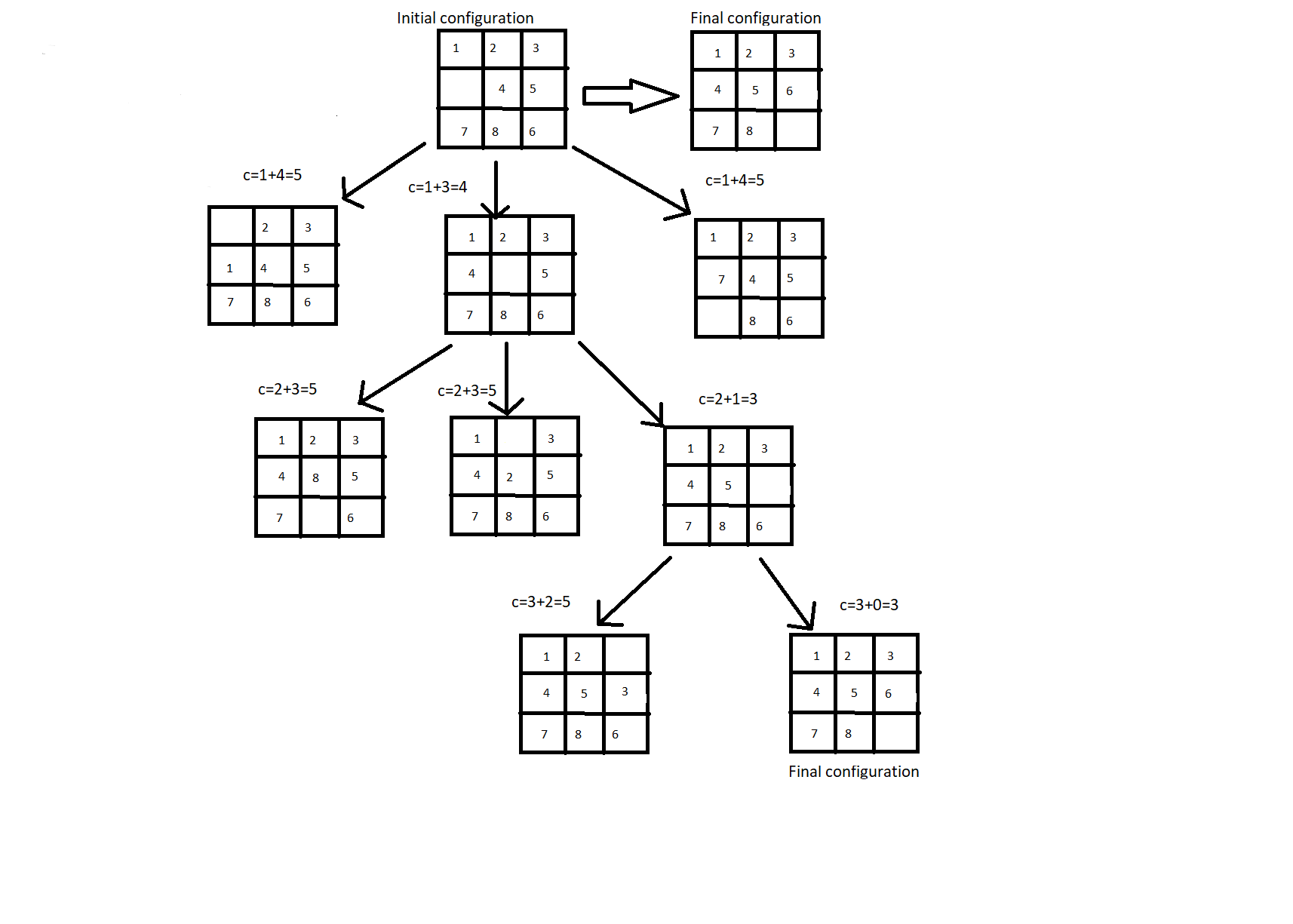
Step 1: Apply heuristics, c(x) = f(x) + g(x),to check the estimate cost towards the solution .where f(x) = distance from initial path, g(x) = no of misplaced positions.

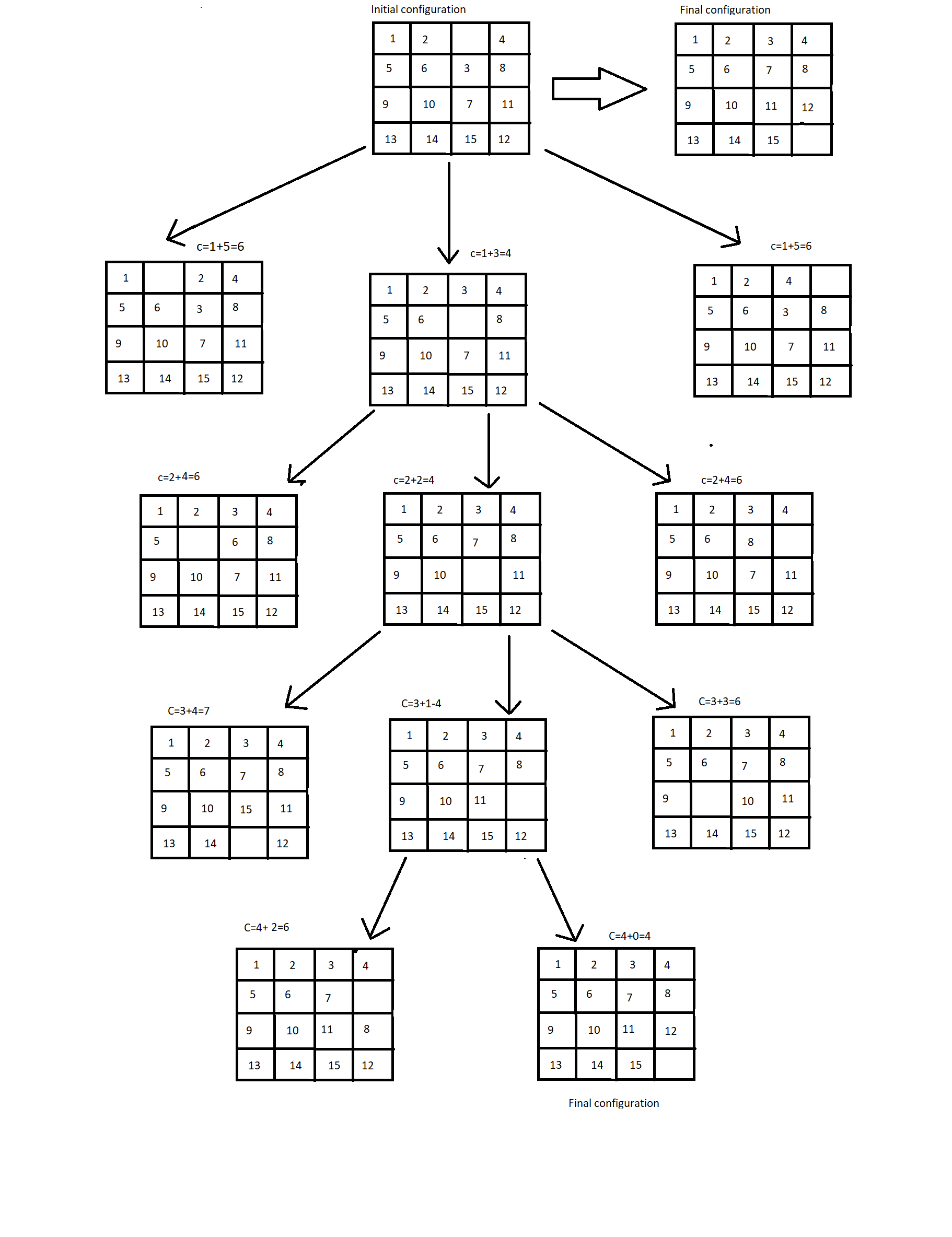
Step 2 : Choose the heuristic with minimum cost and branch the configuration and bound all other remaining configurations.

Step 3 : Repeat the heuristics on the branched state if the goal configuration is not achieved.

Step 4 : While applying the heuristics don’t consider the counter movements.

Step 5 : If g(x)=0,the no of misplaced solutions are zero , then we arrived at the final configuration of the puzzle.





C++ CODE

#include <cstdlib>

#include <iostream>

#include <stdio.h>

#include <math.h>

using namespace std;

int main()

{ srand(time(0));

int m[4][4],i,j,i1,j1,bi,bj,nbi,nbj,tbi,tbj,hu,nhu,temp,flag,randi,randj;

int st[5000][2],top=1; st[0][0]=-1;st[0][1]=-1;st[1][0]=-1;st[1][1]=-1;

// To generate initial random board.

for(i=0;i<4;i++){ for(j=0;j<4;j++){ m[i][j]=4\*i+j; }}

for(i=0;i<4;i++){ for(j=0;j<4;j++){ randi = rand()%4; randj = rand()%4;

swap( m[i][j], m[randi][randj] );

}}

cout<<"The initial board generated is"<<endl;

for(i=0;i<4;i++){ for(j=0;j<4;j++){cout<< m[i][j];cout<<" ";}cout<<endl;}

// To locate the initial zero within the board as blank cell//

for(i=0;i<4;i++){ for(j=0;j<4;j++){ if(m[i][j]==0){ bi = i; bj= j;} }}

flag=1;

while(flag)

{ hu=9999;

for(i=0;i<2;i++){ for(j=0;j<2;j++){

if(j%2){nbi=bi+(1-2\*i);nbj=bj;}

else{ nbj=bj+(1-2\*i);nbi=bi;}

// This module generates all 4 possible moves.

// The following module finds if the possibility is legal i.e not going out of the board and for each possible move calculates a heuristic value.

// The heuristic is the distance of the possible board form the goal board.

// Out of the maximum 4 possible boards which ever has minimum heuristic value that will be selected.

// Since all possibilities from a state are generated, this is BFS.

// As we apply constraints to find a solution (BFS+ constraints) this is a sample of Branch-and-bound scheme.

if(nbi<4&&nbi>=0&&nbj<4 &&nbj>=0)

{

nhu=0;

for(i1=0;i1<4;i1++){ for(j1=0;j1<4;j1++){

nhu=nhu+pow((m[i1][j1]-(4\*i1+j1+1)),2);

}};

printf("Value of the heuristic = %d\n",nhu);

if( nhu==0) flag=0;

// If the heuristic value is 0 the current board is the solution//

if(nhu<hu){ if(nbi!=st[top-1][0]||nbj!=st[top-1][1])

{ hu=nhu;tbi=nbi,tbj=nbj; }

}//if

}//if

}}//for

st[++top][0]=tbi;st[top][1]=tbj;

printf("considering %d and %d\n", tbi,tbj);

system("PAUSE");

temp=m[bi][bj]; m[bi][bj]=m[tbi][tbj];m[tbi][tbj]=temp; bi=tbi;bj=tbj;

//show the puzzle layout

for(i=0;i<4;i++)

{

for(j=0;j<4;j++)

if(m[i][j] == 0) printf(" ");

else printf("%3d ",m[i][j]);

printf("\n");

}

}//while

printf("got solution");

system("PAUSE");

return EXIT\_SUCCESS;

}