**N queen 8x8 backtracking**

#include <iostream>

using namespace std;

class CNode {

public:

char X[8][8];

int r;

CNode\* pNext;

};

class CStack {

public:

CNode\* PHead;

CStack() {

PHead = NULL;

}

~CStack()

{

CNode\* pTrav;

pTrav = PHead;

while (PHead != NULL)

{

PHead = pTrav->pNext;

pTrav->pNext = NULL;

delete pTrav;

pTrav = PHead;

}

}

void Push(CNode\* pnn)

{

pnn->pNext = PHead;

PHead = pnn;

}

CNode\* Pop()

{

if (PHead == NULL)

{

return NULL;

}

CNode\* pTrav = PHead;

PHead = PHead->pNext;

pTrav->pNext = NULL;

return pTrav;

}

};

void Copy(CNode\* pnn, CNode\* pCurr)

{

for (int r = 0; r < 8; r++)

{

for (int c = 0; c < 8; c++)

{

pnn->X[r][c] = pCurr->X[r][c];

}

}

pnn->r = pCurr->r;

}

CNode\* initialize()

{

CNode\* pnn = new CNode;

pnn->pNext = NULL;

pnn->r = 0;

for (int r = 0; r < 8; r++)

{

for (int c = 0; c < 8; c++)

{

pnn->X[r][c] = ' ';

}

}

return pnn;

}

void Expand(CNode\* pCurr, CStack& S)

{

CNode\* pnn;

int check = 0, ct = 0;

for (int i = 0; i < 8; i++)

{

check = 0;

pnn = new CNode;

Copy(pnn, pCurr);

pnn->X[pnn->r][i] = 'Q';

//row

ct = 0;

for (int r = 0; r < 8; r++)

{

if (pnn->X[r][i] == 'Q')

{

ct++;

if (ct > 1)

{

check = 1;

break;

}

}

}

//column

ct = 0;

for (int c = 0; c < 8 && pnn->r < 8; c++)

{

if (pnn->X[pnn->r][c] == 'Q')

{

ct++;

if (ct > 1)

{

check = 1;

break;

}

}

}

//diagonal check

ct = 0;

for (int r = (pnn->r) - 1, c = i - 1; r >= 0 && pnn->r < 8 && c >= 0; r--, c--) //left up

{

if (pnn->X[r][c] == 'Q')

{

ct++;

if (ct == 1)

{

check = 1;

break;

}

}

}

for (int r = (pnn->r) + 1, c = i + 1; r < 8 && pnn->r < 8 && c < 8; r++, c++) //left down

{

if (pnn->X[r][c] == 'Q')

{

ct++;

if (ct == 1)

{

check = 1;

break;

}

}

}

ct = 0;

for (int r = (pnn->r) - 1, c = i + 1; r >= 0 && c < 8; r--, c++) //right up

{

if (pnn->X[r][c] == 'Q')

{

ct++;

if (ct == 1)

{

check = 1;

break;

}

}

}

for (int r = (pnn->r) + 1, c = i - 1; r < 8 && c >= 0; r++, c--) //right down

{

if (pnn->X[r][c] == 'Q')

{

ct++;

if (ct == 1)

{

check = 1;

break;

}

}

}

if (check == 0)

{

if (pnn->r < 8)

{

(pnn->r)++;

}

S.Push(pnn);

}

}

}

void BackTrack(CStack& S)

{

CNode\* pCurr;

CNode\* pTrav = S.PHead;

int check = 0;

pCurr = initialize();

int i = 0;

S.Push(pCurr);

while (S.PHead != NULL)

{

if (S.PHead->r < 8)

{

pCurr = S.Pop();

}

else {

pTrav = S.PHead;

while (pTrav->pNext != NULL)

{

if (pTrav->pNext->r < 8)

{

pCurr = pTrav->pNext;

pTrav->pNext = pTrav->pNext->pNext;

break;

}

pTrav = pTrav->pNext;

}

if (pTrav->pNext == NULL)

{

check = 1;

}

}

if (!check)

{

Expand(pCurr, S);

}

else

{

break;

}

}

}

int main() {

CStack S;

BackTrack(S);

int ct = 0;

CNode\* pTrav = S.PHead;

while (pTrav != NULL)

{

for (int r = 0; r < 8; r++)

{

for (int c = 0; c < 8; c++)

{

cout << pTrav->X[r][c] << " | ";

}

cout << endl;

cout << "----------------------------------" << endl;

}

ct++;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

pTrav = pTrav->pNext;

}

cout << ct << endl;

return 0;

}