ЗВІТ

до лабораторної роботи №3

на тему “ Рішення задачі «гра у 8» за допомогою методу оціночної функції”

Мета роботи: Ознайомитись з інформативними методами пошуку рішення задач в просторі станів, типовим представником яких є алгоритм А*. Навчитись застосовувати алгоритм А* на практиці”

Індивідуальне завдання:

1)Розв’язати задачу «гра у 8» за допомогою методу оціночної функції до третього рівня дерева пошуку (початковий стан –перший рівень) ручним способом. У звіті навести дерево пошуку.

2)Розробити алгоритм і програму рішення задачі «гра у 8» за допомогою методу оціночної функції.

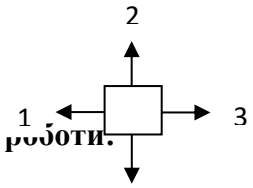
3) У звіті навести загальну кількість згенерованих станів, кількість станів занесених в базу станів, кількість відкинутих станів, глибину дерева пошуку, на якій знайдено рішення.

6) У звіті навести блок-схему алгоритму і роздрук програми.

Заданий стан:

5	8	6
3	1	4
7		2

Алгоритм пересування пустої фішки:



Хід роботи:

Еврістична 4 кція h-сума відстаней кожного числа від поточного до кінцевого стану

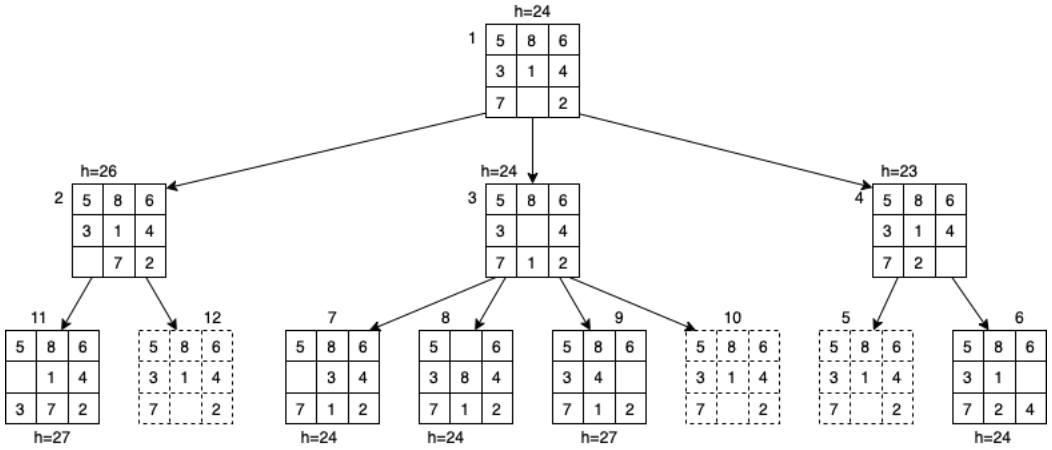


Рисунок 1. Дерево пошуку

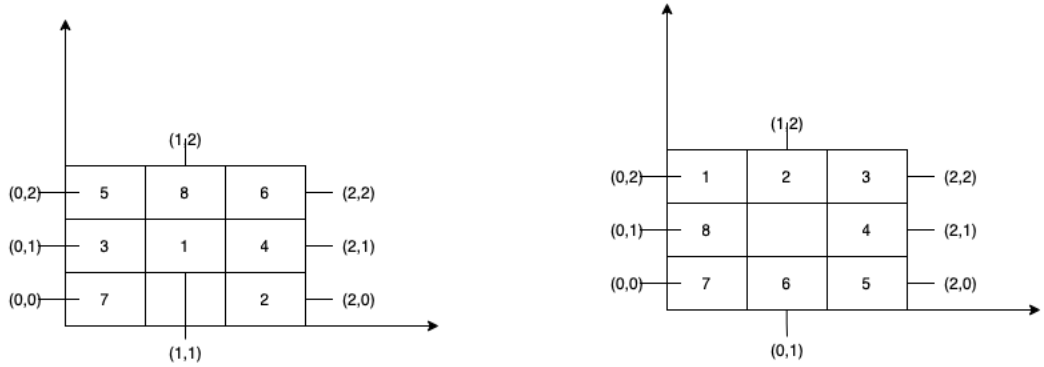


Рисунок 2. Початковий та кінцевий стан на координатній площині

Розрахунок евристичної функції:

$h = d + n$, де $d -$ мангеттенська відстань, $n -$ кількість значень не на своєму місці

Розрахунок Мангеттенської відстані:

$d(n) = |x_i - x_j| + |y_i - y_j|$, де $n -$ значення комірки,

$x_i, y_i -$ координати комірки в початковому стані,

$x_j, y_j -$ координати комірки в кінцевому

1)

$d(5) = |0 - 2| + |2 - 0| = 4$

$d(8) = |1 - 0| + |2 - 1| = 2$

$d(6) = |2 - 0| + |2 - 1| = 3$

$d(3) = |0 - 2| + |1 - 2| = 3$

$d(1) = |1-0| + |1-2| = 2$

$d(4) = |2-2| + |1-1| = 0$

$d(7) = |0-0| + |0-0| = 0$

$d(2) = |2-1| + |0-2| = 3$

$d = \sum d(n) = 4+2+3+3+2+0+0+3 = 17$

$n = 7$

$h = 17 + 7 = 24$

2)

$d(5) = |0-2| + |2-0| = 4$

$d(8) = |1-0| + |2-1| = 2$

$d(6) = |2-0| + |2-1| = 3$

$d(3) = |0-2| + |1-2| = 3$

$d(1) = |1-0| + |1-2| = 2$

$d(4) = |2-2| + |1-1| = 0$

$d(7) = |0-0| + |0-1| = 1$

$d(2) = |2-1| + |0-2| = 3$

$d = \sum d(n) = 4+2+3+3+2+0+1+3 = 18$

$n = 8$

$h = 18 + 8 = 26$

3)

$d(5) = |0-2| + |2-0| = 4$

$d(8) = |1-0| + |2-1| = 2$

$d(6) = |2-0| + |2-1| = 3$

$d(3) = |0-2| + |1-2| = 3$

$d(1) = |1-0| + |0-2| = 3$

$d(4) = |2-2| + |1-1| = 0$

$d(7) = |0-0| + |0-0| = 0$

$d(2) = |2-1| + |0-2| = 3$

$d = \sum d(n) = 4+2+3+3+3+0+0+3 = 18$

$n = 6$

$h = 18 + 6 = 24$

4)

$d(5) = |0-2| + |2-0| = 4$

$d(8) = |1-0| + |2-1| = 2$

$d(6) = |2-0| + |2-1| = 3$

$d(3) = |0-2| + |1-2| = 3$

$d(1) = |1-0| + |1-2| = 2$

$d(4) = |2-2| + |1-1| = 0$

$d(7) = |0-0| + |0-0| = 0$

$d(2) = |1-1| + |0-2| = 2$

$d = \sum d(n) = 4+2+3+3+2+0+0+2 = 16$

$n = 7$

$h = 17 + 6 = 23$

Код програми:

```
#include <iostream>
#include<vector>
#include<queue>
#include<stack>
#include <unordered_set>
#include<fstream>
#define N 3 //number of row/coll
using namespace std;
vector<vector<int>> result={{1,2,3},{8,0,4},{7,6,5}}; // result matrix, 0-
```

```
represent empty position


long long matrix_to_ll(vector<vector<int>> &matrix)
{
    long long res=matrix[0][0];
    for(int i=0;i<matrix.size();i++)
    {
        for(int j=0;j<matrix.size();j++)
        {
            res=res<<4;
            res=res | matrix[i][j];
        }
    }
    return res;
}


vector<vector<int>> ll_to_matrix(long long var)
{
    vector<vector<int>> res(N,vector<int>(N));
    int multiplayer=0;
    for(int i=res.size()-1;i>=0;i--)
    {
        for(int j=res.size()-1;j>=0;j--)
        {
            res[i][j]=(var>>(4*multiplayer)) & 0xf; //0xf-read 4-bite 0x7- read
3-bite 0x3- read 2-bite
            multiplayer++;
        }
    }

    return res;
}




struct Comp
{
    bool operator()(pair<int, int> a,pair<int, int>  b)
    {
        return a.second > b.second;
    }
};


int h1(vector<vector<int>>&matrix ) //sum(distance of blocks)
{

    int sum=0;
    for(int i=0;i<matrix.size();i++)
    {
        for(int j=0;j<matrix.size();j++)
        {
            if(matrix[i][j]==0)
                continue;
            else
            {
                for(int m=0;m<result.size();m++)
                {
                    for(int n=0;n<result.size();n++)
                    {
                        if(matrix[i][j]==result[m][n])
                            sum+=abs(i-m)+abs(j-n);
```

```
                    }
                }
            }
        }
    }
    return sum;
}


int h2(vector<vector<int>>&matrix)
{
    int counter=0;
    for(int i=0;i<matrix.size();i++)
    {
        for(int j=0;j<matrix.size();j++)
        {
            if(matrix[i][j]!=result[i][j])
                counter++;
        }
    }
    return counter;
}


vector<pair<long long,int>> find_position_heuristic(vector<vector<int>> matrix)
{
    vector<pair<long long,int>> res;

    int ind1,ind2;
    for(int i=0;i<matrix.size();i++)
    {
        for(int j=0;j<matrix.size();j++)
        {
            if(matrix[i][j]==0)
            {
                ind1=i;
                ind2=j;
                break;
            }

        }
    }
    if(ind2-1>=0) //check left
    {
        vector<vector<int>> temp=matrix;
        temp[ind1][ind2]=temp[ind1][ind2-1];
        temp[ind1][ind2-1]=0;
        res.push_back(make_pair(matrix_to_ll(temp),h1(temp)+h2(temp)));
    }
    if(ind1-1>=0)  //check up
    {
        vector<vector<int>> temp=matrix;
        temp[ind1][ind2]=temp[ind1-1][ind2];
        temp[ind1-1][ind2]=0;
        res.push_back(make_pair(matrix_to_ll(temp),h1(temp)+h2(temp)));
    }
    if(ind2+1<=matrix.size()-1) //check right
    {
        vector<vector<int>> temp=matrix;
        temp[ind1][ind2]=temp[ind1][ind2+1];
        temp[ind1][ind2+1]=0;
        res.push_back(make_pair(matrix_to_ll(temp),h1(temp)+h2(temp)));
    }
    if(ind1+1<=matrix.size()-1) //check down
    {
```

```
        vector<vector<int>> temp=matrix;
        temp[ind1][ind2]=temp[ind1+1][ind2];
        temp[ind1+1][ind2]=0;
        res.push_back(make_pair(matrix_to_ll(temp),h1(temp)+h2(temp)));
    }


    return res;
}

vector<long long> a_star(vector<vector<int>>& start)
{
    priority_queue<pair<long long,int>,vector<pair<long long,int>>,Comp> pq;
    unordered_set<long long> visited;
    vector<long long> vis;
    ofstream f("ai_3.txt", ios::app);
    pq.push(make_pair(matrix_to_ll(start),0));
    visited.insert(matrix_to_ll(start));
    int iter=0;
    int ind=0;
    while(!pq.empty())
    {

        long long cur=pq.top().first;
        pq.pop();
        vector<vector<int>> temp= ll_to_matrix(cur);
        vis.push_back(cur);
        f<<"Iteration number "<<ind++<<endl;
        for(int i=0;i<temp.size();i++)
        {
            for(int j=0;j<temp.size();j++)
            {
                f<<temp[i][j];
            }
            f<<endl;
        }
        f<<endl;
        if(temp==result)
        {
            break;
        }
        vector<pair<long long,int>> next= find_position_heuristic(temp);
        for(auto& elem:next)
        {
            auto it=visited.find(elem.first);
            if(it==visited.end())
            {
                visited.insert(elem.first);
                pq.push(elem);
            }
            else
            {
                iter++;
            }
        }
    }
    vis.push_back(static_cast<long long>(iter));
   return vis;

}


void Menu(vector<vector<int>>&matrix )
{
    int key;
```

```
vector<long long> visited=a_star(matrix);
auto it=visited.begin();
while(true)
{
    cout<<"1-Print next state"<<endl;
    cout<<"2-Print last state"<<endl;
    cout<<"3-Exit"<<endl;
    cin>>key;
    if(key==1)
    {
        vector<vector<int>> temp(ll_to_matrix(*it));
        cout<<"Value of heuristic function: "<<h1(temp)+h2(temp)<<endl;
        for(int i=0;i<temp.size();i++)
        {
            for(int j=0;j<temp.size();j++)
            {
                cout<<temp[i][j]<<" ";
            }
            cout<<endl;
        }
        cout<<endl;
        it++;
        if(it++==visited.end())
            break;
        else
            it--;
    }
    else if(key==2)
    {
        auto it_end=visited.end();
        it_end--;
        it_end--;
        vector<vector<int>> temp(ll_to_matrix(*it_end));
        for(int i=0;i<temp.size();i++)
        {
            for(int j=0;j<temp.size();j++)
            {
                cout<<temp[i][j]<<" ";
            }
            cout<<endl;
        }
        cout<<endl;
        if(temp==result)
        {
            cout<<"Final state achieved"<<endl;
        }
        else
        {
            vector<pair<long long,int>> next= find_position_heuristic(temp);
            cout<<"It is impossible to achieve the final state"<<endl;
            cout<<"Next states are already in the state database"<<endl;
            for(auto elem:next)
            {
                auto fnd=find(visited.begin(),visited.end(),elem.first);
                int ind=fnd-visited.begin();
                cout<<"Index in the set database: "<<ind<<endl;
                vector<vector<int>> cur= ll_to_matrix(elem.first);
                for(int i=0;i<cur.size();i++)
                {
                    for(int j=0;j<cur.size();j++)
                    {
                        cout<<cur[i][j]<<" ";
                    }
                    cout<<endl;
                }
```

```
                    cout<<endl;
                }
                cout<<"Number of generated state: "<<visited.size()-
1+visited[visited.size()-1]<<endl;
                cout<<"Number of state added to database: "<<visited.size()-
1<<endl;
                cout<<"Number of rejected state: "<<visited[visited.size()-
1]<<endl;
            }

        }
        else
            return;
    }
}


int main() {
    vector<vector<int>> matrix={{5,8,6},{3,1,4},{7,0,2}}; //start matrix, 0-
represent empty position
    a_star(matrix);

}
```

З даного початкового стану неможливо досягнути кінцевого стану. Тому в результаті роботи алгоритму генерується 181440 станів і додаються в базу станів після чого закінчуються можливі переходи і алгоритм закінчується.

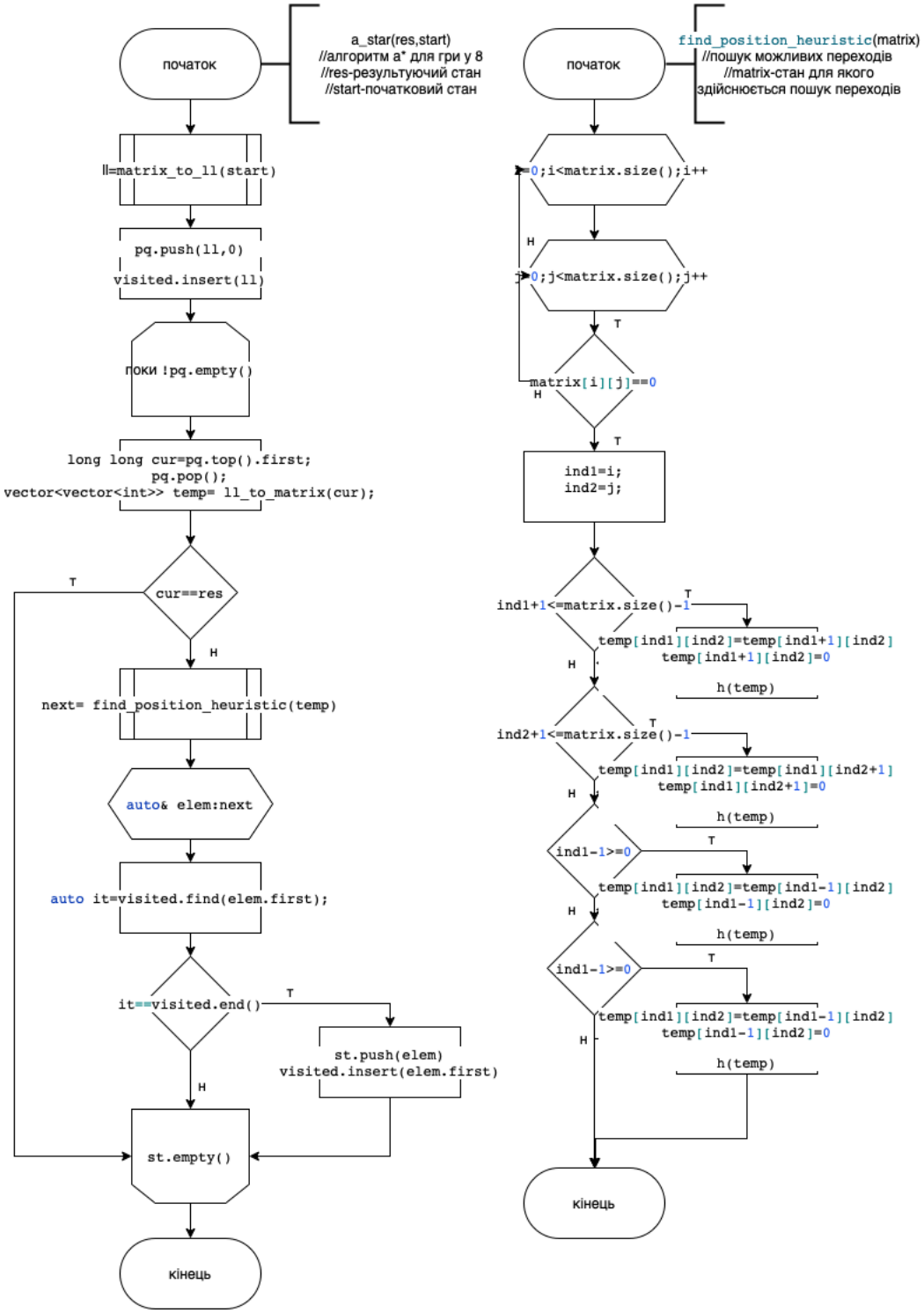


Рисунок 2. Блок-схема алгоритму

Вивід кількох перших станів:

```
Iteration number 1
5 8 6
3 1 4
7 0 2

Iteration number 2
5 8 6
3 1 4
7 2 0

Iteration number 3
5 8 6
3 0 4
7 1 2

Iteration number 4
5 8 6
0 3 4
7 1 2

Iteration number 5
0 8 6
5 3 4
7 1 2

Iteration number 6
8 0 6
5 3 4
7 1 2
```

```
1-Print next state
2-Print last state
3-Exit
2
4 5 7
6 0 1
3 2 8

It is impossible to achieve the final state
Next states are already in the state database
Index in the set database: 180936
4 5 7
0 6 1
3 2 8

Index in the set database: 181360
4 0 7
6 5 1
3 2 8

Index in the set database: 181377
4 5 7
6 1 0
3 2 8

Index in the set database: 180974
4 5 7
6 2 1
3 0 8

Number of generated state: 483841
Number of state added to database: 181440
Number of rejected state: 302401
```

Висновок: В результаті виконання даної лабораторної роботи я ознайомився з інформативними методами пошуку рішення задач в просторі станів, типовим представником яких є алгоритм A*. Навчився застосовувати алгоритм A* на практиці”