**Intelligent System - Project 1**

|  |
| --- |
| **Team**: Utkarsha Gurjar (801149356)  Shreya Vinodh (801149383)  Afreen (801136632) |

### **Problem Formulation**:

Initial state:

An n\*n matrix with -1 entries and 1 blank. For 8-puzzle problem, n = 3.

Goal state:

An n\*n matrix with -1 entries and 1 blank, to be reached from initial state. For 8-puzzle problem, n = 3.

Path cost:

1 per move

Here, we have used f(n) = g(n) + h(n), where g(n) is step cost and h(n) is heuristic cost.

Operators:

Move the blank left, right, up or down.

Successor function - “generate\_child”:

In this function, child nodes are created by moving the blank space in one of the 4 directions i.e. left, right, up, down.

Goal Test :

When the heuristic (h) of a state is zero, that state is considered to be the goal.

### **Program structure:**

An object of class grid created (p)

Class ‘*grid*’ assigns the size, creates two empty lists – gen (for storing generated nodes) and exp(for storing expanded nodes)

p calls main()

User is asked for a choice of heuristic to be used: input()

‘1’ for Misplaced tiles method, ‘2’ for Manhattan distance method.

User is asked to enter initial and goal states for the problem: get()

Initial state initialized with data, g(cost), f value(initially zero)- obj created

Calculate h value for the initial state obj.heuristic()

Calculate f value f()

Initial state added to list gen gen.append()

While loop

Sort the list gen by the f value of the states added in list gen.

Get the first element in the list(least f value)

Generate successor nodes of the state with least f generate\_child()

Add the expanded states to list exp exp.append()

Break condition: break when a state with h = 0 is found

Trace back from the goal state to initial state with the help of a dictionary using the unique key assigned to each state on generation.

Add the traced states to a list, sort the list in ascending order of key.

Print the states.

Class ‘*state*’ assigns data, g, f, and h values, a unique key to the state, and the parent state’s unique key.

Calculates h value according the user’s input choice.

### **Global variables:**

Four global variables are present:

inp - it is used to choose the heuristic method i.e. Misplaced as 1 and Manhattan as 2

goal - it contains the goal state

track - a dictionary to store unique key for each state

key - unique value given to each state generated

### **Functions:**

There are 2 constructors and 8 functions:

1. The constructor of class State initializes the values of f,g,h and the data to be stored in the state.
2. The function “heuristic” calculates the heuristic based on the user input. Here, if the user inputs 1 then Misplaced method will be performed and if the user inputs 2 then Manhattan method will be performed.
3. The function “find” is used to find the indices of a number in goal state.
4. The constructor of class Grid initializes the size of the grid, two empty lists for storing expanded states, and generated states.
5. The function “get” takes initial and goal states from a user
6. The function “copy” copies the parent state and generate successor states.
7. The function “swap\_pos” swaps the positions of the space('0') and the adjacent number.
8. The function “f” calculates the f value by adding the values of step cost and heuristic.
9. The function “generate\_child” is the successor function used to generate successor states.
10. The function “main” where the user is asked for initial state, goal state, the choice of heuristic, and generates successive states

### **Inputs and Outputs:**

|  |  |
| --- | --- |
| **Misplaced** | **Manhattan** |
| **CASE 1:**  Which heuristic?  1. Misplaced Tiles  2. Manhattan Distance  1  Enter numbers from 1-8 for the initial state in 3x3 format(3 values in a row; 0 for blank)  4 1 3  0 2 6  7 5 8  Enter numbers from 1-8 for the goal state in 3x3 format(3 values in a row; 0 for blank)  1 2 3  4 5 6  7 8 0  |4|1|3|  |0|2|6|  |7|5|8|  ---------------------  h = 5, g = 0, f = 5  ---------------------  |  V  |0|1|3|  |4|2|6|  |7|5|8|  ---------------------  h = 4, g = 1, f = 5  ---------------------  |  V  |1|0|3|  |4|2|6|  |7|5|8|  ---------------------  h = 3, g = 2, f = 5  ---------------------  |  V  |1|2|3|  |4|0|6|  |7|5|8|  ---------------------  h = 2, g = 3, f = 5  ---------------------  |  V  |1|2|3|  |4|5|6|  |7|0|8|  ---------------------  h = 1, g = 4, f = 5  ---------------------  |  V  |1|2|3|  |4|5|6|  |7|8|0|  ---------------------  h = 0, g = 5, f = 5  ---------------------  Number of nodes generated: 16  Number of nodes expanded: 6 | **CASE 1:**  Which heuristic?  1. Misplaced Tiles  2. Manhattan Distance  2  Enter numbers from 1-8 for the initial state in 3x3 format(3 values in a row; 0 for blank)  4 1 3  0 2 6  7 5 8  Enter numbers from 1-8 for the goal state in 3x3 format(3 values in a row; 0 for blank)  1 2 3  4 5 6  7 8 0  |4|1|3|  |0|2|6|  |7|5|8|  ---------------------  h = 5, g = 0, f = 5  ---------------------  |  V  |0|1|3|  |4|2|6|  |7|5|8|  ---------------------  h = 4, g = 1, f = 5  ---------------------  |  V  |1|0|3|  |4|2|6|  |7|5|8|  ---------------------  h = 3, g = 2, f = 5  ---------------------  |  V  |1|2|3|  |4|0|6|  |7|5|8|  ---------------------  h = 2, g = 3, f = 5  ---------------------  |  V  |1|2|3|  |4|5|6|  |7|0|8|  ---------------------  h = 1, g = 4, f = 5  ---------------------  |  V  |1|2|3|  |4|5|6|  |7|8|0|  ---------------------  h = 0, g = 5, f = 5  ---------------------  Number of nodes generated: 16  Number of nodes expanded: 6 |
| **CASE 2**:  Which heuristic?  1. Misplaced Tiles  2. Manhattan Distance  1  Enter numbers from 1-8 for the initial state in 3x3 format(3 values in a row; 0 for blank)  2 8 1  3 4 6  7 5 0  Enter numbers from 1-8 for the goal state in 3x3 format(3 values in a row; 0 for blank)  3 2 1  8 0 4  7 5 6  |2|8|1|  |3|4|6|  |7|5|0|  ---------------------  h = 5, g = 0, f = 5  ---------------------  |  V  |2|8|1|  |3|4|0|  |7|5|6|  ---------------------  h = 4, g = 1, f = 5  ---------------------  |  V  |2|8|1|  |3|0|4|  |7|5|6|  ---------------------  h = 3, g = 2, f = 5  ---------------------  |  V  |2|0|1|  |3|8|4|  |7|5|6|  ---------------------  h = 3, g = 3, f = 6  ---------------------  |  V  |0|2|1|  |3|8|4|  |7|5|6|  ---------------------  h = 2, g = 4, f = 6  ---------------------  |  V  |3|2|1|  |0|8|4|  |7|5|6|  ---------------------  h = 1, g = 5, f = 6  ---------------------  |  V  |3|2|1|  |8|0|4|  |7|5|6|  ---------------------  h = 0, g = 6, f = 6  ---------------------  Number of nodes generated: 21  Number of nodes expanded: 8 | **CASE 2:**  Which heuristic?  1. Misplaced Tiles  2. Manhattan Distance  2  Enter numbers from 1-8 for the initial state in 3x3 format(3 values in a row; 0 for blank)  2 8 1  3 4 6  7 5 0  Enter numbers from 1-8 for the goal state in 3x3 format(3 values in a row; 0 for blank)  3 2 1  8 0 4  7 5 6  |2|8|1|  |3|4|6|  |7|5|0|  ---------------------  h = 6, g = 0, f = 6  ---------------------  |  V  |2|8|1|  |3|4|0|  |7|5|6|  ---------------------  h = 5, g = 1, f = 6  ---------------------  |  V  |2|8|1|  |3|0|4|  |7|5|6|  ---------------------  h = 4, g = 2, f = 6  ---------------------  |  V  |2|0|1|  |3|8|4|  |7|5|6|  ---------------------  h = 3, g = 3, f = 6  ---------------------  |  V  |0|2|1|  |3|8|4|  |7|5|6|  ---------------------  h = 2, g = 4, f = 6  ---------------------  |  V  |3|2|1|  |0|8|4|  |7|5|6|  ---------------------  h = 1, g = 5, f = 6  ---------------------  |  V  |3|2|1|  |8|0|4|  |7|5|6|  ---------------------  h = 0, g = 6, f = 6  ---------------------  Number of nodes generated: 18  Number of nodes expanded: 7 |
| **CASE 3:**  Which heuristic?  1. Misplaced Tiles  2. Manhattan Distance  1  Enter numbers from 1-8 for the initial state in 3x3 format(3 values in a row; 0 for blank)  1 3 4  8 6 2  0 7 5  Enter numbers from 1-8 for the goal state in 3x3 format(3 values in a row; 0 for blank)  1 2 3  8 0 4  7 6 5  |1|3|4|  |8|6|2|  |0|7|5|  ---------------------  h = 5, g = 0, f = 5  ---------------------  |  V  |1|3|4|  |8|6|2|  |7|0|5|  ---------------------  h = 4, g = 1, f = 5  ---------------------  |  V  |1|3|4|  |8|0|2|  |7|6|5|  ---------------------  h = 3, g = 2, f = 5  ---------------------  |  V  |1|3|4|  |8|2|0|  |7|6|5|  ---------------------  h = 3, g = 3, f = 6  ---------------------  |  V  |1|3|0|  |8|2|4|  |7|6|5|  ---------------------  h = 2, g = 4, f = 6  ---------------------  |  V  |1|0|3|  |8|2|4|  |7|6|5|  ---------------------  h = 1, g = 5, f = 6  ---------------------  |  V  |1|2|3|  |8|0|4|  |7|6|5|  ---------------------  h = 0, g = 6, f = 6  ---------------------  Number of nodes generated: 21  Number of nodes expanded: 8 | **CASE 3:**  Which heuristic?  1. Misplaced Tiles  2. Manhattan Distance  2  Enter numbers from 1-8 for the initial state in 3x3 format(3 values in a row; 0 for blank)  1 3 4  8 6 2  0 7 5  Enter numbers from 1-8 for the goal state in 3x3 format(3 values in a row; 0 for blank)  1 2 3  8 0 4  7 6 5  |1|3|4|  |8|6|2|  |0|7|5|  ---------------------  h = 6, g = 0, f = 6  ---------------------  |  V  |1|3|4|  |8|6|2|  |7|0|5|  ---------------------  h = 5, g = 1, f = 6  ---------------------  |  V  |1|3|4|  |8|0|2|  |7|6|5|  ---------------------  h = 4, g = 2, f = 6  ---------------------  |  V  |1|3|4|  |8|2|0|  |7|6|5|  ---------------------  h = 3, g = 3, f = 6  ---------------------  |  V  |1|3|0|  |8|2|4|  |7|6|5|  ---------------------  h = 2, g = 4, f = 6  ---------------------  |  V  |1|0|3|  |8|2|4|  |7|6|5|  ---------------------  h = 1, g = 5, f = 6  ---------------------  |  V  |1|2|3|  |8|0|4|  |7|6|5|  ---------------------  h = 0, g = 6, f = 6  ---------------------  Number of nodes generated: 18  Number of nodes expanded: 7 |
| **CASE 4:**  Which heuristic?  1. Misplaced Tiles  2. Manhattan Distance  1  Enter numbers from 1-8 for the initial state in 3x3 format(3 values in a row; 0 for blank)  1 2 3  7 4 5  6 8 0  Enter numbers from 1-8 for the goal state in 3x3 format(3 values in a row; 0 for blank)  1 2 3  8 6 4  7 5 0  |1|2|3|  |7|4|5|  |6|8|0|  ---------------------  h = 5, g = 0, f = 5  ---------------------  |  V  |1|2|3|  |7|4|0|  |6|8|5|  ---------------------  h = 5, g = 1, f = 6  ---------------------  |  V  |1|2|3|  |7|0|4|  |6|8|5|  ---------------------  h = 4, g = 2, f = 6  ---------------------  |  V  |1|2|3|  |7|8|4|  |6|0|5|  ---------------------  h = 4, g = 3, f = 7  ---------------------  |  V  |1|2|3|  |7|8|4|  |0|6|5|  ---------------------  h = 4, g = 4, f = 8  ---------------------  |  V  |1|2|3|  |0|8|4|  |7|6|5|  ---------------------  h = 3, g = 5, f = 8  ---------------------  |  V  |1|2|3|  |8|0|4|  |7|6|5|  ---------------------  h = 2, g = 6, f = 8  ---------------------  |  V  |1|2|3|  |8|6|4|  |7|0|5|  ---------------------  h = 1, g = 7, f = 8  ---------------------  |  V  |1|2|3|  |8|6|4|  |7|5|0|  ---------------------  h = 0, g = 8, f = 8  ---------------------  Number of nodes generated: 118  Number of nodes expanded: 39 | **CASE 4:**  Which heuristic?  1. Misplaced Tiles  2. Manhattan Distance  2  Enter numbers from 1-8 for the initial state in 3x3 format(3 values in a row; 0 for blank)  1 2 3  7 4 5  6 8 0  Enter numbers from 1-8 for the goal state in 3x3 format(3 values in a row; 0 for blank)  1 2 3  8 6 4  7 5 0  |1|2|3|  |7|4|5|  |6|8|0|  ---------------------  h = 8, g = 0, f = 8  ---------------------  |  V  |1|2|3|  |7|4|0|  |6|8|5|  ---------------------  h = 7, g = 1, f = 8  ---------------------  |  V  |1|2|3|  |7|0|4|  |6|8|5|  ---------------------  h = 6, g = 2, f = 8  ---------------------  |  V  |1|2|3|  |7|8|4|  |6|0|5|  ---------------------  h = 5, g = 3, f = 8  ---------------------  |  V  |1|2|3|  |7|8|4|  |0|6|5|  ---------------------  h = 4, g = 4, f = 8  ---------------------  |  V  |1|2|3|  |0|8|4|  |7|6|5|  ---------------------  h = 3, g = 5, f = 8  ---------------------  |  V  |1|2|3|  |8|0|4|  |7|6|5|  ---------------------  h = 2, g = 6, f = 8  ---------------------  |  V  |1|2|3|  |8|6|4|  |7|0|5|  ---------------------  h = 1, g = 7, f = 8  ---------------------  |  V  |1|2|3|  |8|6|4|  |7|5|0|  ---------------------  h = 0, g = 8, f = 8  ---------------------  Number of nodes generated: 27  Number of nodes expanded: 10 |

### **Summary Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Initial State | Goal State | Number of nodes generated  Misplaced Manhattan | | Number of nodes expanded  Misplaced Manhattan | |
| |4|1|3|  |0|2|6|  |7|5|8| | |1|2|3|  |4|5|6|  |7|8|0| | 16 | 16 | 6 | 6 |
| |2|8|1|  |3|4|6|  |7|5|0| | |3|2|1|  |8|0|4|  |7|5|6| | 21 | 18 | 8 | 7 |
| |1|3|4|  |8|6|2|  |0|7|5| | |1|2|3|  |8|0|4|  |7|6|5| | 21 | 18 | 8 | 7 |
| |1|2|3|  |7|4|5|  |6|8|0| | |1|2|3|  |8|6|4|  |7|5|0| | 118 | 27 | 39 | 10 |

### **Source code:**

class state:

#intializes data to each state, cost(g), and initial value to f for each state

def \_\_init\_\_(self,data, g, f, h, p\_key):

self.data = data

self.g = g

self.f = f

self.h = h

#key of parent node to trace back the solution path

self.p\_key = p\_key

#key of the current node

self.s\_key = key

#function to calculate heuristic based on the input from the user

def heuristic(self, inp, initial, goal):

#for calculating misplaced tiles

if inp == '1':

h = 0

for i in range(len(self.data)):

for j in range(len(self.data)):

#check if position of a value(not space) in initial state and goal state is different and increment h

if self.data[i][j] != goal[i][j] and self.data[i][j] != '0':

h+=1

return h

#for calculating Manhattan distance

elif inp == '2':

h=0

for i in range(len(self.data)):

for j in range(len(self.data)):

if self.data[i][j] != '0':

x,y = self.find(goal,self.data[i][j])

#to find how many blocks away is the position of a value in initial state from the position in goal state

if i == x and j!=y:

h += abs(y-j)

if i!=x and j == y:

h += abs(x-i)

if i!=x and j!=y:

h += abs(x-i) + abs(y-j)

return h

#helper function to find the indices of a number in goal state

def find(self,goal,num):

for i in range(len(self.data)):

for j in range(len(self.data)):

if goal[i][j] == num:

return i,j

class grid:

#intializes size, a list for storing generated states, and a list for storing expanded states

def \_\_init\_\_(self,size):

self.size = size

self.gen = []

self.exp = []

#function to take input and goal states from a user

def get(self,n):

grid = []

for i in range(n):

row = input().split(" ")

grid.append(row)

return grid

#helper function to copy the parent state and form a successor state

def copy(self, parent):

copy\_node = []

for i in parent.data:

elem = []

for j in i:

elem.append(j)

copy\_node.append(elem)

return copy\_node

#function to swap positions of the space('0') and the adjacent number

def swap\_pos(self,parent,sp\_x,sp\_y,val\_x,val\_y):

p\_copy = self.copy(parent)

temp = p\_copy[sp\_x][sp\_y]

p\_copy[sp\_x][sp\_y] = p\_copy[val\_x][val\_y]

p\_copy[val\_x][val\_y] = temp

return p\_copy

#function to calculate f

def f(self,h, node):

return (h + node.g)

#function to generate successor states

def generate\_child(self,node):

global key

for i in range(len(node.data)):

for j in range(len(node.data)):

if node.data[i][j] == '0':

x=i

y=j

break

adj\_pos = [[x-1, y], [x+1,y], [x,y-1], [x,y+1]]

children = []

for i in adj\_pos:

#check if adjacent position is a valid position

if i[0]>=0 and i[0]<len(node.data) and i[1]>=0 and i[1]<len(node.data):

child = self.swap\_pos(node,x,y,i[0],i[1])

#increment key before assigning a new state as value in the dictionary

key = key + 1

child\_state = state(child, node.g+1,0,0,node.s\_key)

child\_state.h = child\_state.heuristic(inp,self,goal)

child\_state.f = self.f(child\_state.h, child\_state)

#list of successor states from the parent state

children.append(child\_state)

#add the new child state to dictionary with a unique key

track[key]=child\_state

return children

#Asks the user for intial state, goal state, the choice of heuristic, and generates successive states

def main(self):

global inp

global goal

#unique value as key for a dictionary

global key

#a dictionary to store a unique key for each state

global track

key = 0

track={}

print("Which heuristic?")

print("1. Misplaced Tiles")

print("2. Manhattan Distance")

inp = input()

if inp!='1' and inp!='2':

print("Invalid choice")

return

print("Enter numbers from 1-8 for the initial state in 3x3 format(3 values in a row; 0 for blank)")

initial = self.get(p.size)

print("Enter numbers from 1-8 for the goal state in 3x3 format(3 values in a row; 0 for blank)")

goal = self.get(p.size)

#increment key before assigning a new state as value in the dictionary

key = key + 1

initial\_state = state(initial,0,0,0,0)

initial\_state.h = initial\_state.heuristic(inp,initial,goal)

initial\_state.f = self.f(initial\_state.h, initial\_state)

#add initial state to dictionary

track[key] = initial\_state

count\_gen = 1

self.gen.append(initial\_state)

print()

while True:

#sort the list that has generated states in ascending order of 'f' value

self.gen.sort(key = lambda x:x.f, reverse=False)

#get the first state(with least f value)

first = self.gen[0]

if first.h == 0:

val = first.s\_key

break

successors = self.generate\_child(first)

count\_gen += len(successors)

self.exp.append(first)

del self.gen[0]

for child in successors:

self.gen.append(child)

#trace back from goal state to initial state using key of the parent state

pr = []

key = val

while key!= 0:

cur\_node = track[key]

pr.append(cur\_node)

key = cur\_node.p\_key

#sort the list in ascending order of the key

pr.sort(key = lambda x:x.s\_key, reverse=False)

for k in pr:

if k.s\_key!=1:

print(" | ")

print(" V ")

for i in k.data:

print("|", end='')

for j in i:

print(j, end='|')

print()

print("---------------------")

print("h =", k.h, end=', ')

print("g =", k.g, end=', ')

print("f =", k.f)

print("---------------------")

print("Number of nodes generated: ", end='')

print(count\_gen)

print("Number of nodes expanded: ", end='')

print((len(self.exp)+1))

p = grid(3)

p.main()