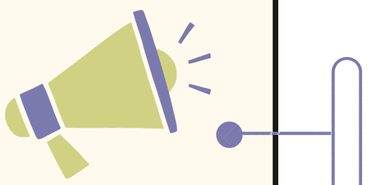
KECERDASAN BUATAN F

Heuristic

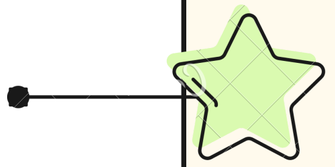
Misplaced

INFORMED SEARCH

Presented by Keluarga Berencana



Manhattan



8-puzzle

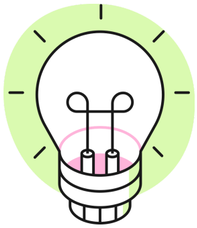
5025211129

Farrela Ranku Mahhisa

5025211134

Faizah Nurdianti Maghfirah

5025211222



Shafa Nabilah Hanin

INFORMED SEARCH

Algoritma pencarian yang memanfaatkan pengetahuan khusus atau informasi tambahan untuk memberikan oetunjuk untuk solusi masalah.

HEURISTIC FUNCTION

Menjadi jalan pintas untuk memecahkan masalah yang tidak memiliki solusi yang tepat dan memakan waktu yang lama dalam mendapatkan solusi.

MANHATTAN DISTANCE

Menghitung jarak yang akan ditempuh untuk berpindah dari satu titik data ke titik lain

MISPLACED TILES

Dimana petak yang tidak pada tempatnya harus dipindahkan setidaknya satu kali untuk mengaturnya ke keadaan tujuan

MANHATTAN DISTANCE

```python
# print matrix hasil akhir
def print_in_format(matrix):
    for i in range(9):
        if i%3==0 and i>0:
            print("")
        print(str(matrix[i])+" ", end = "")

# transfer inputan jadi bentuk matriks
def convert(s):
    mat = []
    a = []
    b = []
    c = []
    for i in range(9):
        if i<3:
            a.append(s[i])
        if i>=3 and i<=5:
            b.append(s[i])
        if i>5:
            c.append(s[i])

    mat.append(a)
    mat.append(b)
    mat.append(c)
    return mat
```

```python
# mencari hasil manhattan distance
def ideal_distFind(val):
    x1 = 999
    y1 = 999
    ideal = [[1, 2, 3],
             [4, 5, 6],
             [7, 8, 0]]

    for i in range(3):
        for j in range(3):
            if ideal[i][j]==val:
                x1 = i
                y1 = j
                break
    return x1, y1

def count(initial_state):
    inits = initial_state.copy()
    inicon = convert(inits)
    x1 = y1 = x2 = y2 = 999
    total_h = 0;

    for i in range(3):
        for j in range(3):
            x1, y1 = ideal_distFind(inicon[i][j])
            x2, y2 = i, j
            total_h += abs(x1-x2)+abs(y1-y2)

    return total_h
```

MANHATTAN DISTANCE

```python
#membuat pergerakan puzzle
def move(ar, p, st):
    rh = 9999
    store_st = st.copy()

    for i in range(len(ar)):

        dupl_st = st.copy()

        tmp = dupl_st[p]
        dupl_st[p] = dupl_st[arr[i]]
        dupl_st[arr[i]] = tmp

        trh = count(dupl_st)

        if trh<rh:
            rh = trh
            store_st = dupl_st.copy()

    #print(rh, store_st)

    return store_st, rh


state = [1, 2, 3,
         4, 5, 6,
         0, 7, 8]

h = count(state)
Level = 1

print("\n------ Level "+str(Level)+" ------")
print_in_format(state)
print("\nHeuristic Value(Manhattan Distance) : "+str(h))
```

```python
while h>0:
    pos = int(state.index(0))

    Level += 1

    if pos==0:
        arr = [1, 3]
        state, h = move(arr, pos, state)
    elif pos==1:
        arr = [0, 2, 4]
        state, h = move(arr, pos, state)
    elif pos==2:
        arr = [1, 5]
        state, h = move(arr, pos, state)
    elif pos==3:
        arr = [0, 4, 6]
        state, h = move(arr, pos, state)
    elif pos==4:
        arr = [1, 3, 5, 7]
        state, h = move(arr, pos, state)
    elif pos==5:
        arr = [2, 4, 8]
        state, h = move(arr, pos, state)
    elif pos==6:
        arr = [3, 7]
        state, h = move(arr, pos, state)
    elif pos==7:
        arr = [4, 6, 8]
        state, h = move(arr, pos, state)
    elif pos==8:
        arr = [5, 6]
        state, h = move(arr, pos, state)

    print("\n------ Level "+str(Level)+" ------")
    print_in_format(state)
    print("\nHeuristic Value(Manhattan Distance) : "+str(h))
```

OUTPUT

```
------ Level 1 ------
1 2 3
4 5 6
7 0 8
Heuristic Value(Manhattan Distance) : 2

------ Level 2 ------
1 2 3
4 5 6
7 8 0
Heuristic Value(Manhattan Distance) : 0
PS D:\SEMESTER 4\KB>
```

MISPLACED TILES

```python
def print_in_format(matrix):
    for i in range(9):
        if i%3==0 and i>0:
            print("")
        print(str(matrix[i])+" ", end = "")

def count(s):
    c = 0
    ideal = [1, 2, 3,
             4, 5, 6,
             7, 8, 0]

    for i in range(9):
        if s[i]!=0 and s[i]!=ideal[i]:
            c+=1
    return c


def move(ar, p, st):
    rh = 9999
    store_st = st.copy()

    for i in range(len(ar)):

        dupl_st = st.copy()

        tmp = dupl_st[p]
        dupl_st[p] = dupl_st[arr[i]]
        dupl_st[arr[i]] = tmp

        trh = count(dupl_st)

        if trh<rh:
            rh = trh
            store_st = dupl_st.copy()

    return store_st, rh
```

```python
state = [1, 2, 3,
         0, 5, 6,
         4, 7, 8]

h = count(state)
Level = 1

print("\n------ Level "+str(Level)+" ------")
print_in_format(state)
print("\nHeuristic Value(Misplaced) : "+str(h))
```

MISPLACED TILES

```
while h>0:
    pos = int(state.index(0))

    Level += 1

    if pos==0:
        arr = [1, 3]
        state, h = move(arr, pos, state)
    elif pos==1:
        arr = [0, 2, 4]
        state, h = move(arr, pos, state)
    elif pos==2:
        arr = [1, 5]
        state, h = move(arr, pos, state)
    elif pos==3:
        arr = [0, 4, 6]
        state, h = move(arr, pos, state)
    elif pos==4:
        arr = [1, 3, 5, 7]
        state, h = move(arr, pos, state)
    elif pos==5:
        arr = [2, 4, 8]
        state, h = move(arr, pos, state)
    elif pos==6:
        arr = [3, 7]
        state, h = move(arr, pos, state)
    elif pos==7:
        arr = [4, 6, 8]
        state, h = move(arr, pos, state)
    elif pos==8:
        arr = [5, 6]
        state, h = move(arr, pos, state)

    print("\n------ Level "+str(Level)+" ------")
    print_in_format(state)
    print("\nHeuristic Value(Misplaced) : "+str(h))
```

OUTPUT

```
------ Level 1 ------
1 2 3
0 5 6
4 7 8
Heuristic Value(Misplaced) : 3

------ Level 2 ------
1 2 3
4 5 6
0 7 8
Heuristic Value(Misplaced) : 2

------ Level 3 ------
1 2 3
4 5 6
7 0 8
Heuristic Value(Misplaced) : 1

------ Level 4 ------
1 2 3
4 5 6
7 8 0
Heuristic Value(Misplaced) : 0
```

KECERDASAN BUATAN F

THANK YOU

KELUARGA BERENCANA