λ. cost + depth

Manhattan Heuristic

1	2	3
4	5	6
7	8	0

abs $(x - x')$ + abs $(y - y')$

↗ ↗

goal ob

location

in

goal

puzzle

5 is at (1,0)

Should be at (1,1)

1	2	3
4	5	6
7	8	0

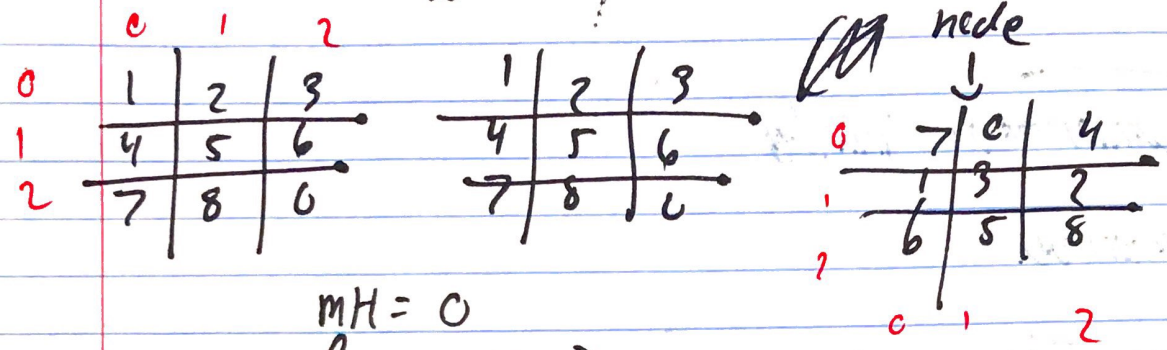
0	1	2	3
1	4	5	6
2	7	8	0

(0, 0) (2, 1)

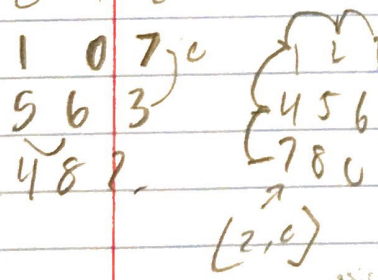
$(2 - 0) + (1 - 0)$

$= 3$

return i: [0, 0]
i := 0



mH = 0
for i: →
  for j: →
    find [i][i] in "real" // find 7 in real
    // this will return [2,0]
    // curr is [0][0]
    calculate |2-0| + |0-0|
    mH += calculated value // mH = 2



~~// find 2 in "real"~~ // ~~find element @ [i][j]~~
~~// returns [0,1]~~
~~mH += 0~~

when element at [i][i] is ZERO, skip (?)

if([i][j]) == 0
do nothing
// j will increment.

// this should work

~~count~~

```
curr = 1
for ( rows )
    for ( each val in col )
        if [i][j] != curr
            find (value) in the passed node
            increement the count;
```

Uniform cost → this looks backwards, and tries to find the best solution

→ optimal & complete

enqueue nodes by cost

g(n)

keep expanding

A* only looks backwards for uniform cost search

Uniform cost search → breath first search

Measure the cost to each node

going to need a new queueing function

and at node of... ?

a node of each state...?

[0

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

← (+1, +0) = (1, 2)

↑ (+0, -1) = (2, 1)

↓ (+1, +0) = (3, 1) X

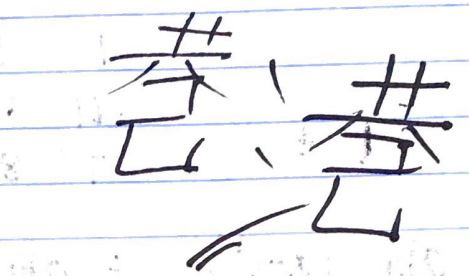
BFS → FINISHED

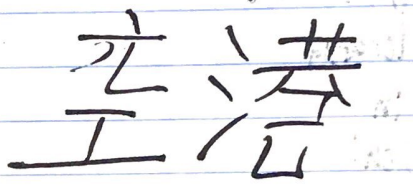
Depth

Manhattan.

Each will

→ sort fun python or sorted





I'll need to save by

how to get total number of rows & colums in python?

```
  ↓ ↓
  1 6 7
→ 5 C 3←
  4 8 2
  ↑ ↑
```

& every time you append,

Generic Search Algorithm

Initial state = given state

goal state → check if all the numbers are in order

problem → the entire problem state

problem : initial state
final state
operators

making a problem space

problem space

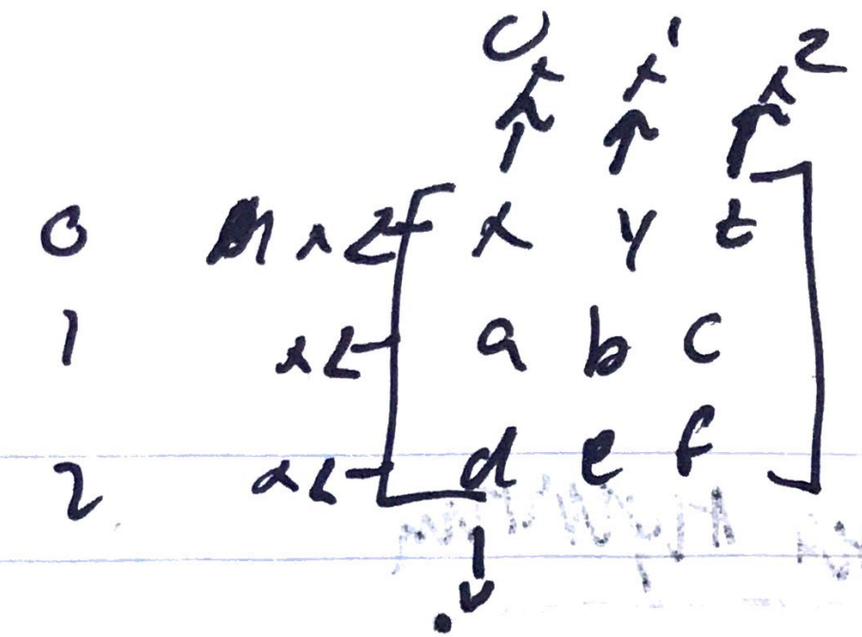
else, you got to expand and then there the things

man

move up → only illegal when index is 0

keep track of depth.

Ch 3.4



Extend

Go up it

0 is False

1 is True