

# AI heuristic search with Don't Group

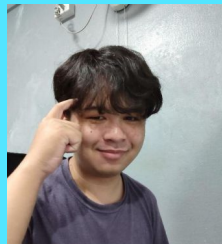
ARTIFICIAL INTELLIGENCE  
CE KMITL 2022

## TEAM เอ๋ไอ้



สุธี สาระพันธ์  
63015190

- Idea & Game



ภูษิต เลือโคร่ง  
63015137

-Input Function  
-Statistics



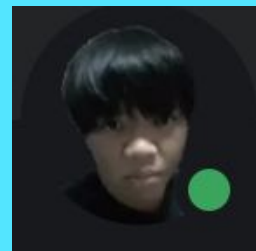
ธณวัฒน์ สุขแก้ว  
63015069

-Statistics  
-Tester



วายุ แสงพิทักษ์  
63015161

-A-star



ศรายุทธ พอค้า  
63015165

-BFS  
-Statistics

# What is เกมนี้

*Don't Group* ได้แรงบันดาลใจจากตัวเกมของ 8-puzzle ที่ให้เราเรียงเลขจาก 1 - 8 โดยได้นำไอเดียในตัวเกมนี้อาพัฒนาเป็น *Don't Group* ที่เราจะต้องเรียงสีที่ใส่เข้าไป โดยที่ไม่ให้สีนั้น ๆ เรียงติดกัน

8-puzzle

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Don't Group

Initial State

Red	Red	Blue	Blue
Yellow	Red	Green	Yellow
Red	Yellow	Blue	Yellow
Yellow	Blue	Green	Red

Goal State

Red	Blue	Red	Yellow
Yellow	Red	Green	Blue
Red	Yellow	Blue	Yellow
Yellow	Blue	Green	Red

## จุดหมายของเกมนี้


ในแต่ละโจทยจะมีความกว้างเท่ากับ  $n * n$  บล็อกและแต่ละบล็อกจะมีสีของมัน ( 4 สี )  
ซึ่งจะมีบางบล็อกที่สีเหมือนกันแล้วอยู่ติดกัน เป้าหมายของเราคือทำยังไงก็ได้ให้แต่ละ  
บล็อกไม่มีสีเดียวกันอยู่ติดกัน ( ไม่นับแนวทแยงมุม ) โดยแต่ละรอบเราสามารถสลับ  
บล็อกได้เป็นรูปบวกเท่านั้น



ลักษณะโจทย

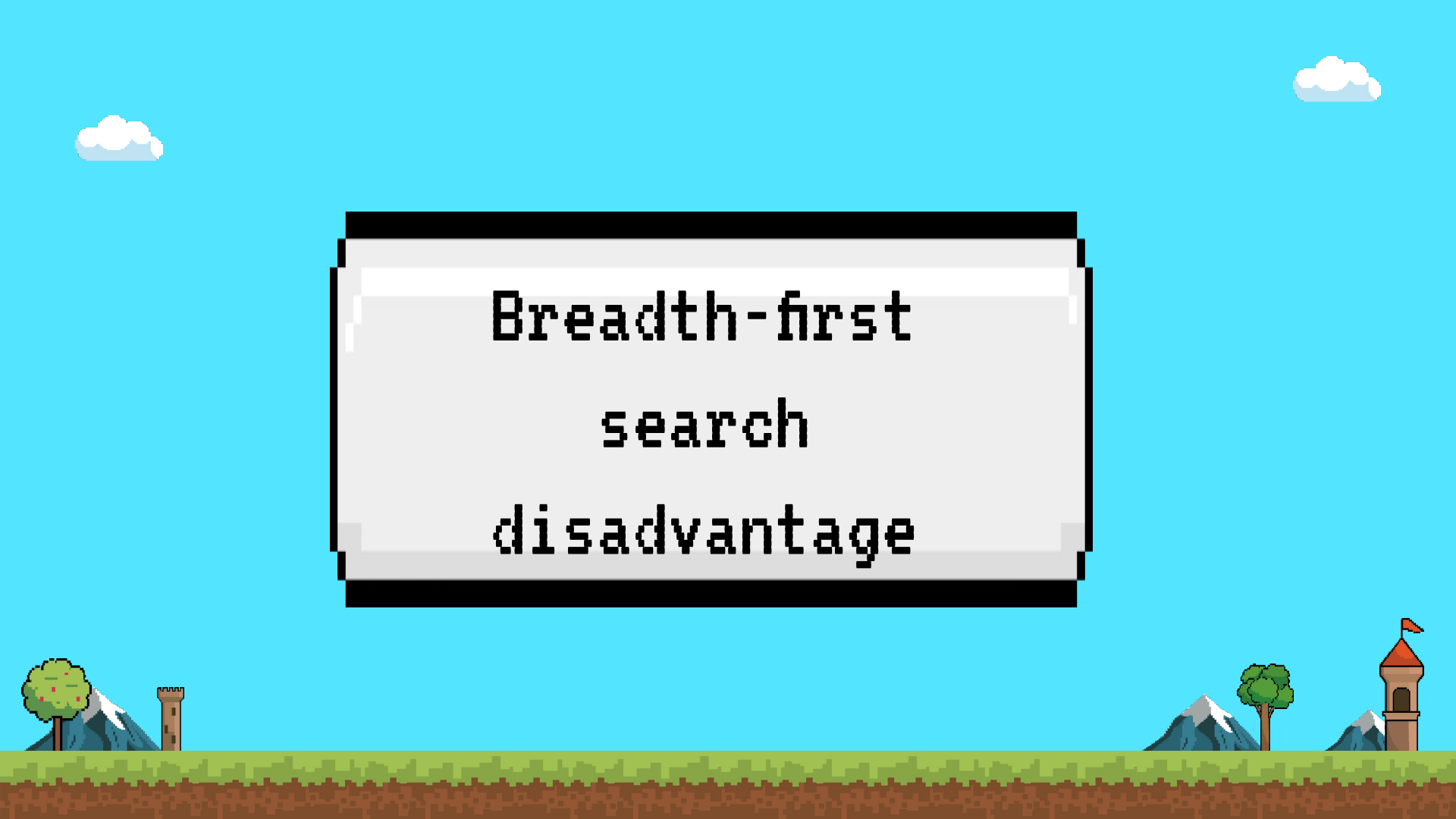


จุดหมายของโจทย



# Breadth-first search





Breadth-first  
search  
disadvantage

Copy of packai.py - Colaboratory

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

itr : 398843 depth: 6

Board state:

■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■

itr : 398844 depth: 6

Board state:

■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■

itr : 398845 depth: 6

Board state:

■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■

itr : 398846 depth: 6

Board state:

■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■
■	■	■	■	■	■

itr : 398846 depth: 6

Board state:



Executing (1h 40m 36s)

ls() > send()

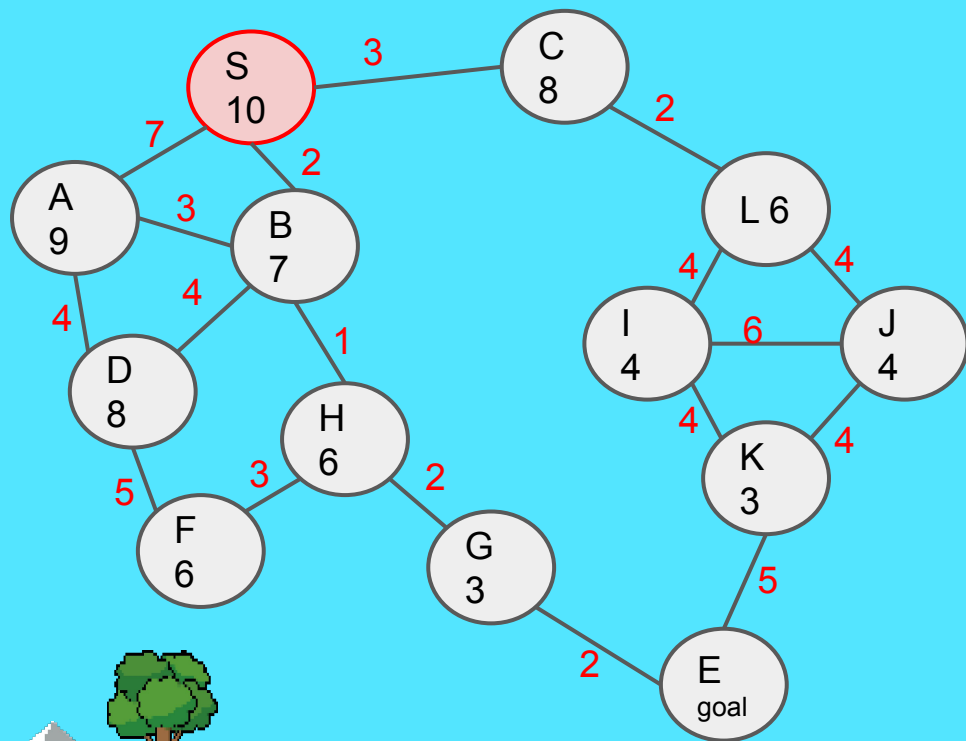






A-star search

## Example

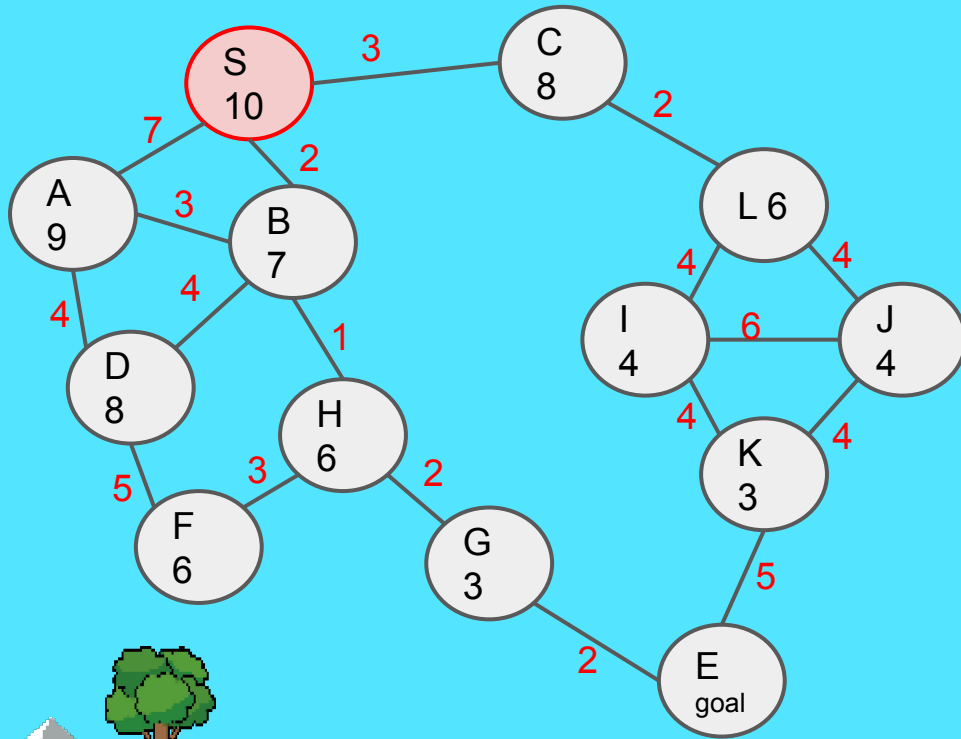


We started at S node.

S	distance	Combined heuristic
	0	10

And adds neighbor nodes.

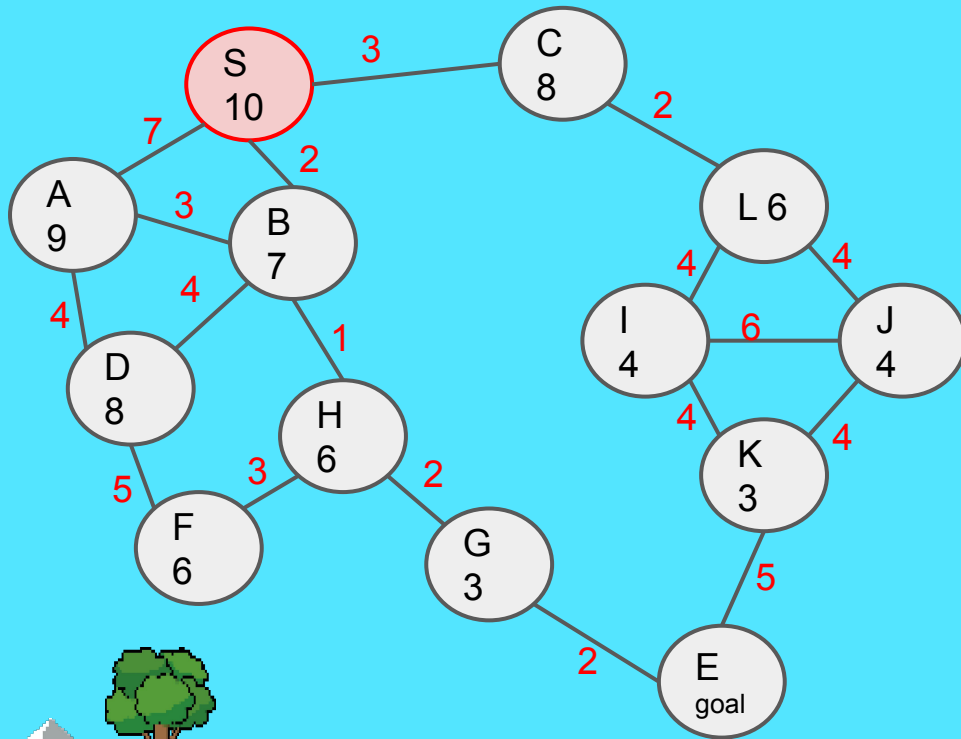
# Example



## Queue node

A	distance	Combined heuristic
	7	16

## Example



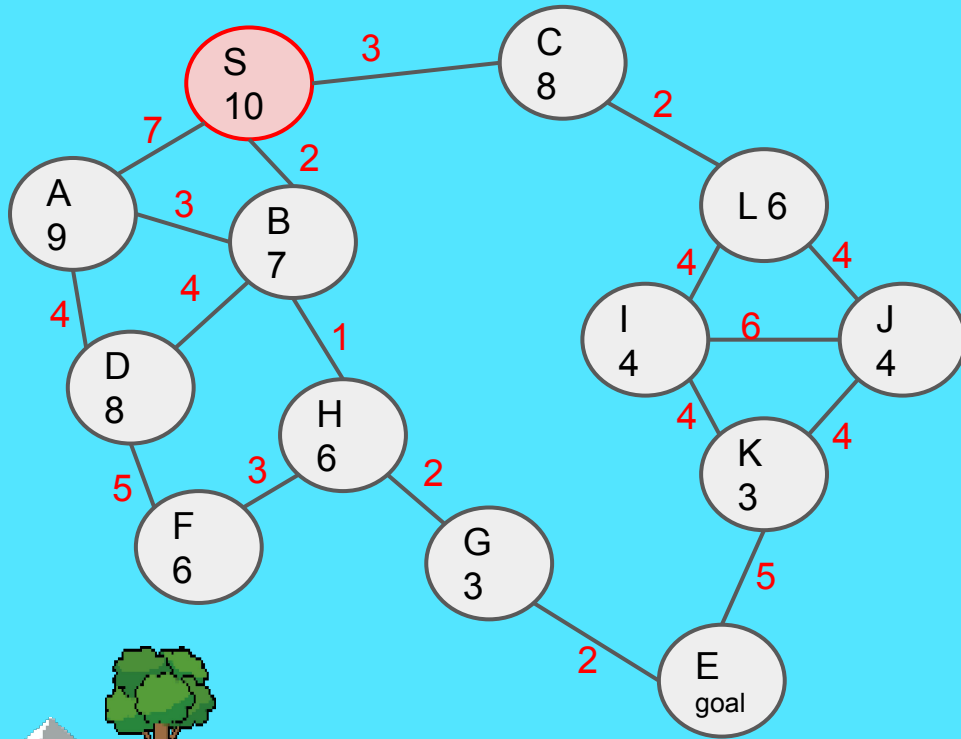
## Queue node

A	distance	Combined heuristic
	7	16

B	distance	Combined heuristic
	2	9

Then we add "B" to queue

## Example



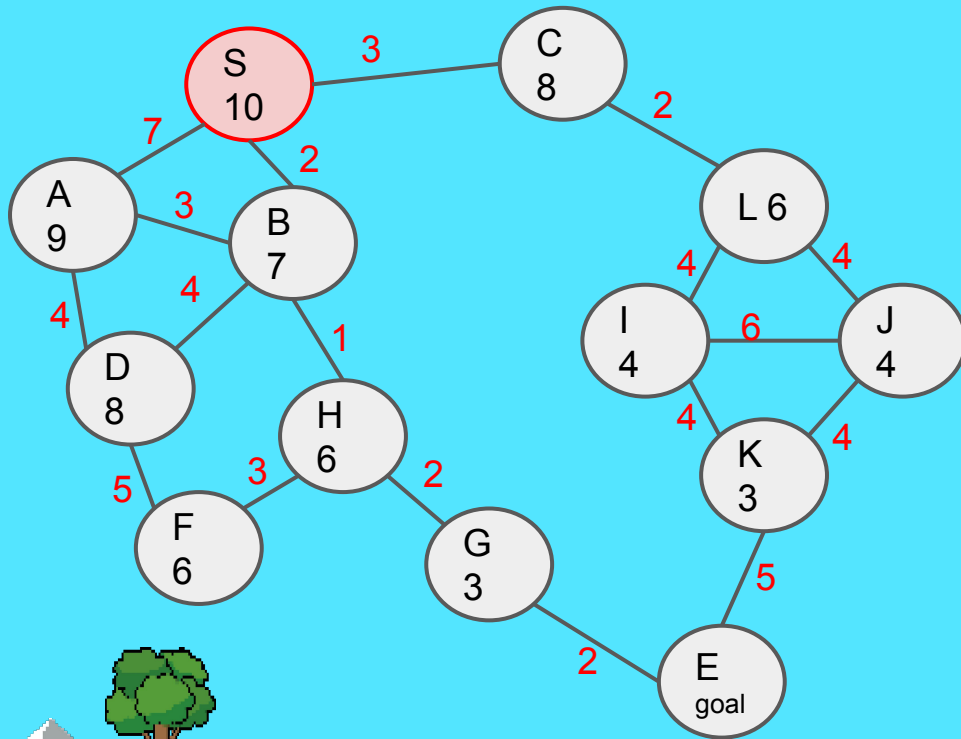
## Queue node

B	distance	Combined heuristic
	2	9

A	distance	Combined heuristic
	7	16

A is lowerd due to  
higher combined  
heuristic value.

# Example



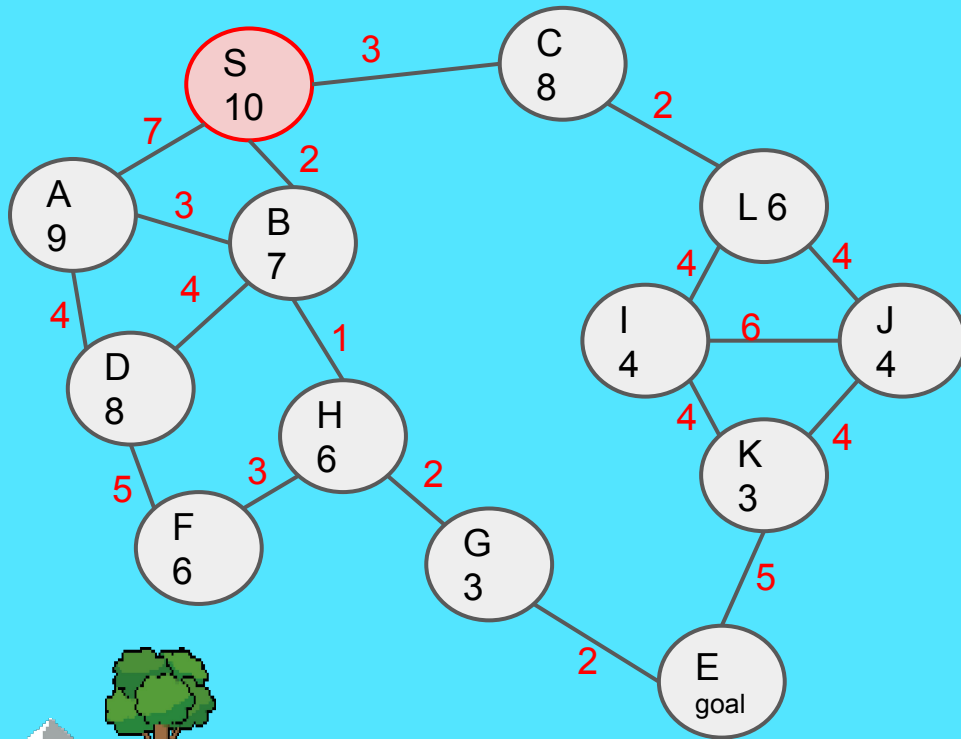
## Queue node

B	distance	Combined heuristic
	2	9

A	distance	Combined heuristic
	7	16

C	distance	Combined heuristic
	3	11

## Example



## Queue node

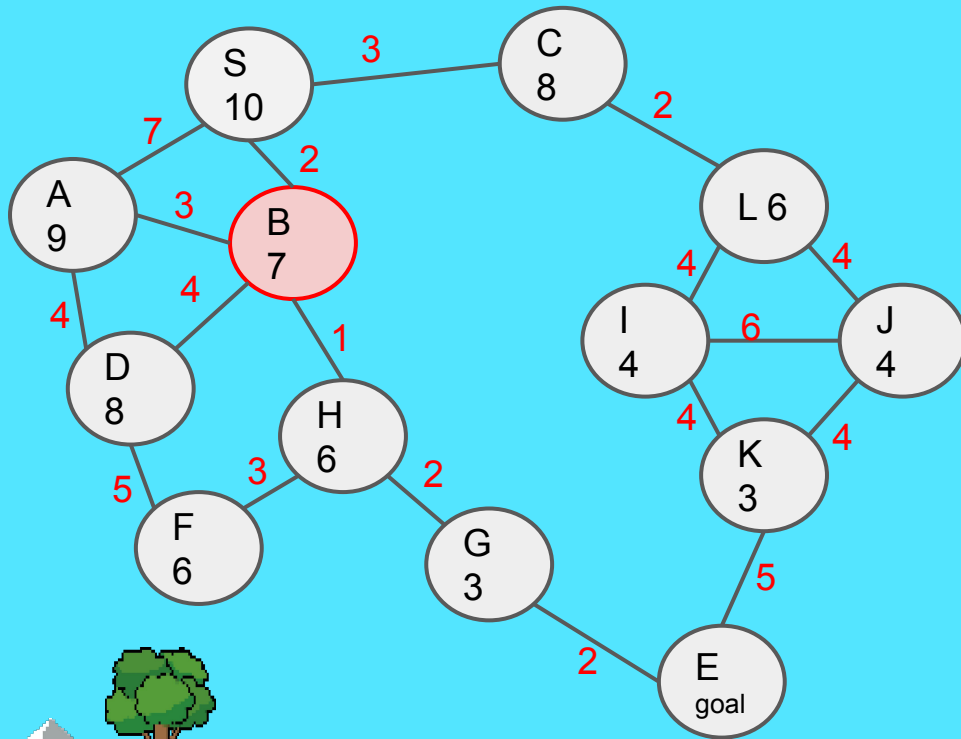
B	distance	Combined heuristic
	2	9

C	distance	Combined heuristic
	3	11

A	distance	Combined heuristic
	7	16

"C" goes above

## Example



## Queue node

B	distance	Combined heuristic
	2	9

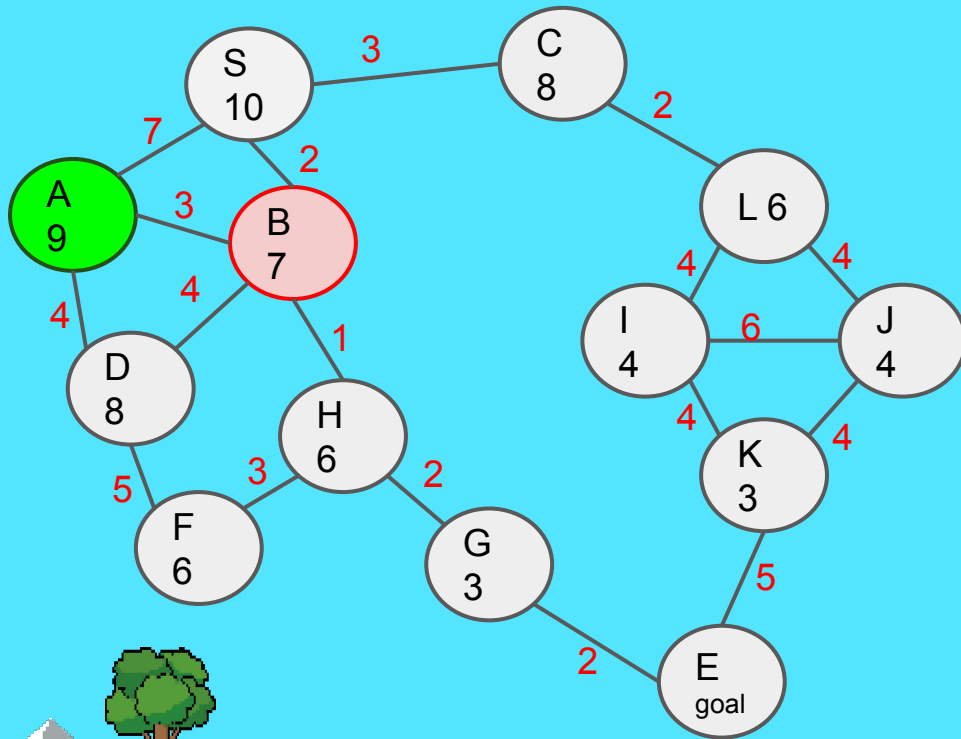
C	distance	Combined heuristic
	3	11

A	distance	Combined heuristic
	7	16

Expanding  
"B"



## Example



## Queue node

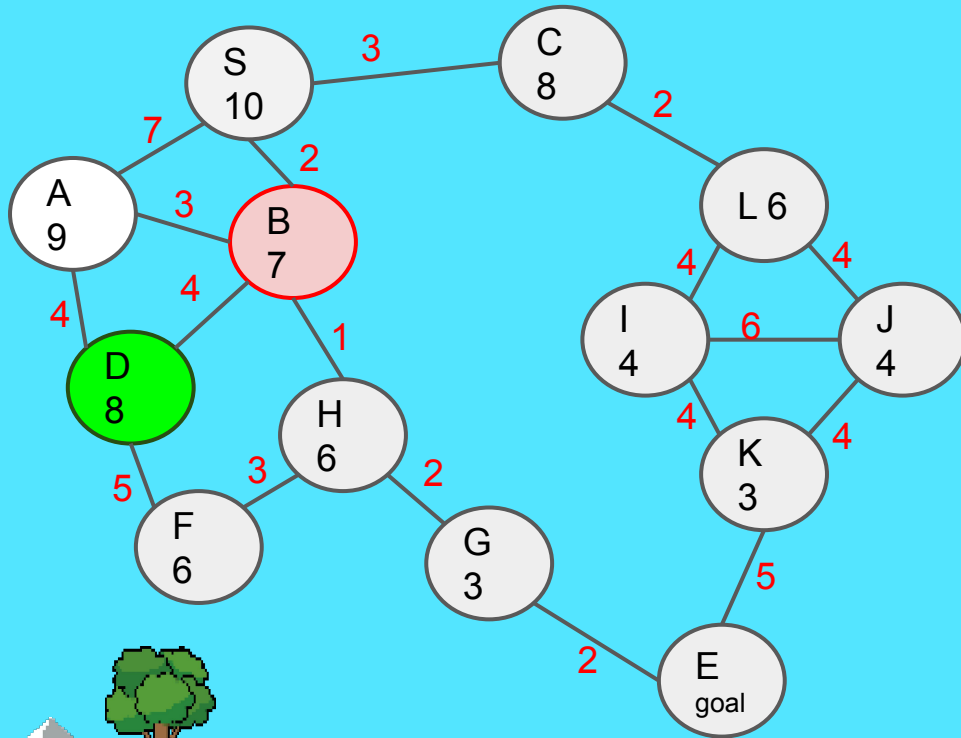
B	distance	Combined heuristic
	2	9

C	distance	Combined heuristic
	3	11

A	distance	Combined heuristic
	5	14

Change A value

## Example



## Queue node

B	distance	Combined heuristic
	2	9

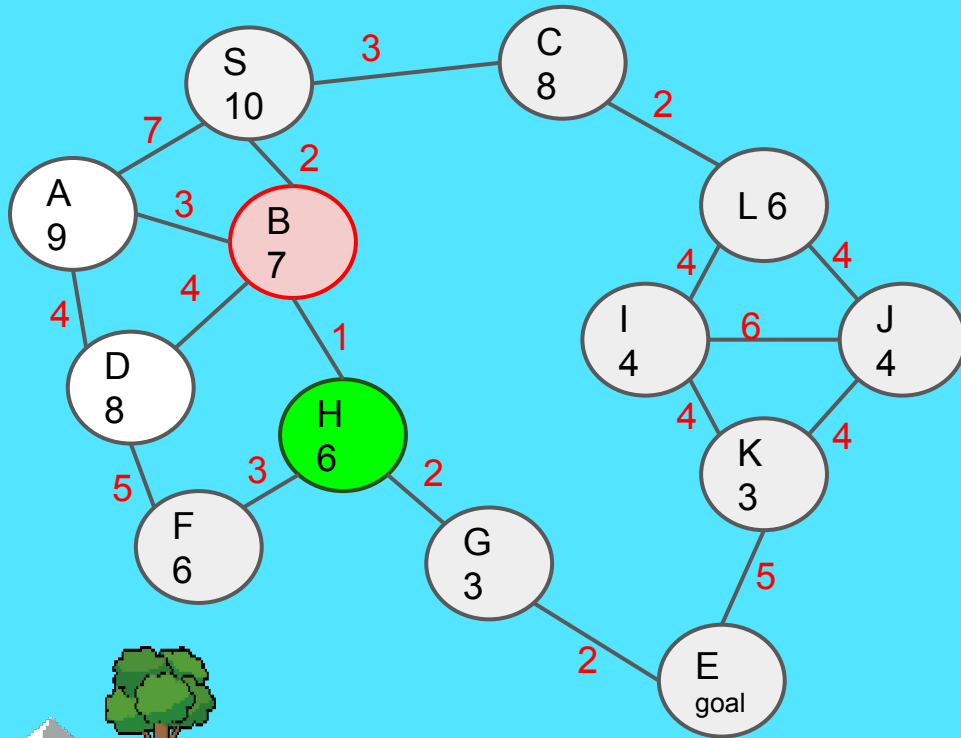
C	distance	Combined heuristic
	3	11

A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

Add "D"

## Example



## Queue node

B	distance	Combined heuristic
	2	9

C	distance	Combined heuristic
	3	11

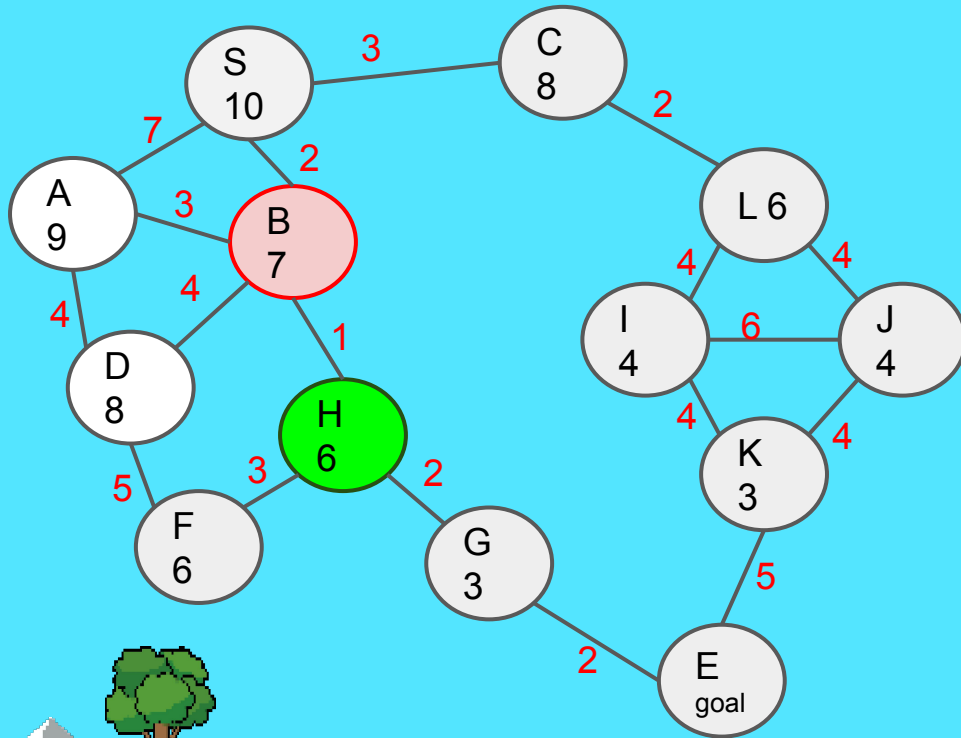
A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

H	distance	Combined heuristic
	3	9

Add "H"

# Example



## Queue node

B	distance	Combined heuristic
	2	9

H	distance	Combined heuristic
	3	9

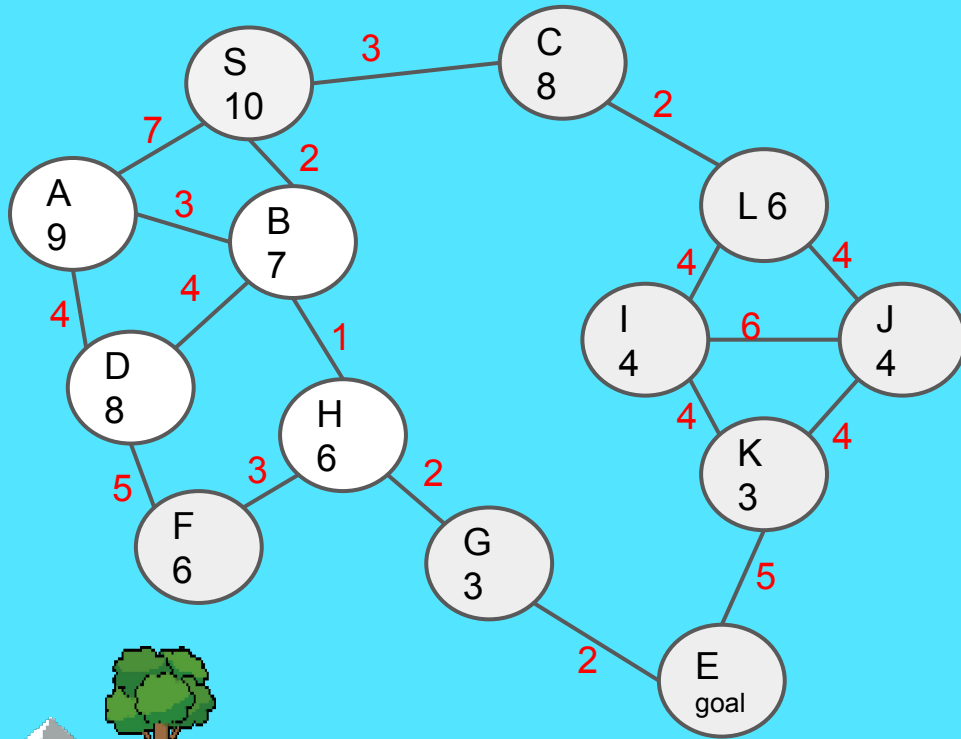
C	distance	Combined heuristic
	3	11

A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

"H" goes above

## Example



## Queue node

H	distance	Combined heuristic
	3	9

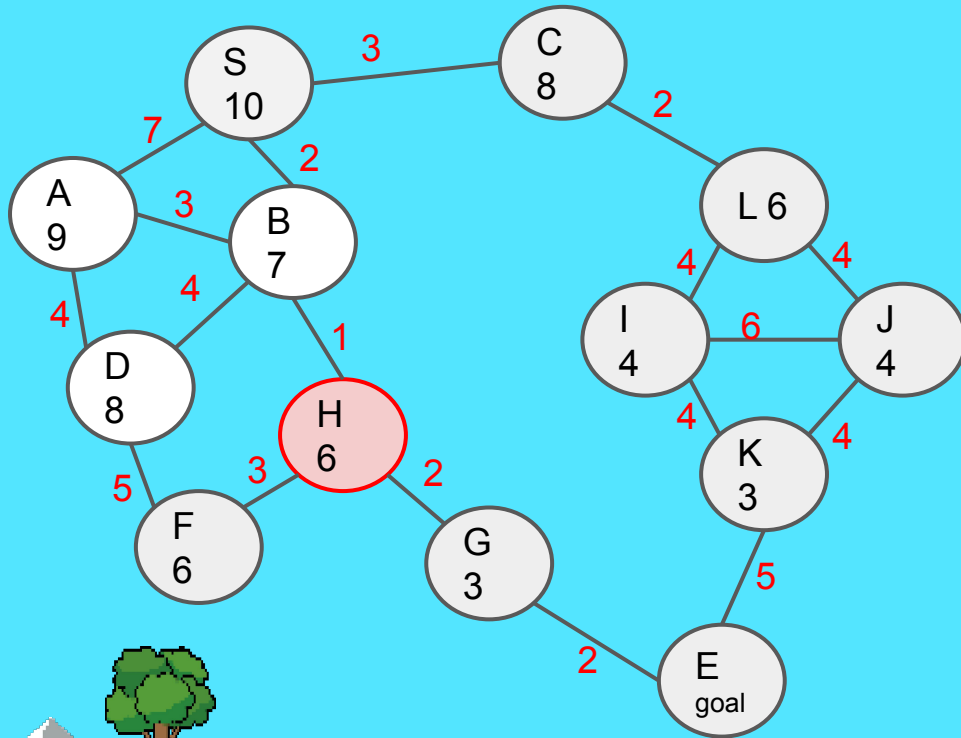
C	distance	Combined heuristic
	3	11

A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

"B" is popped

## Example



## Queue node

H	distance	Combined heuristic
	3	9

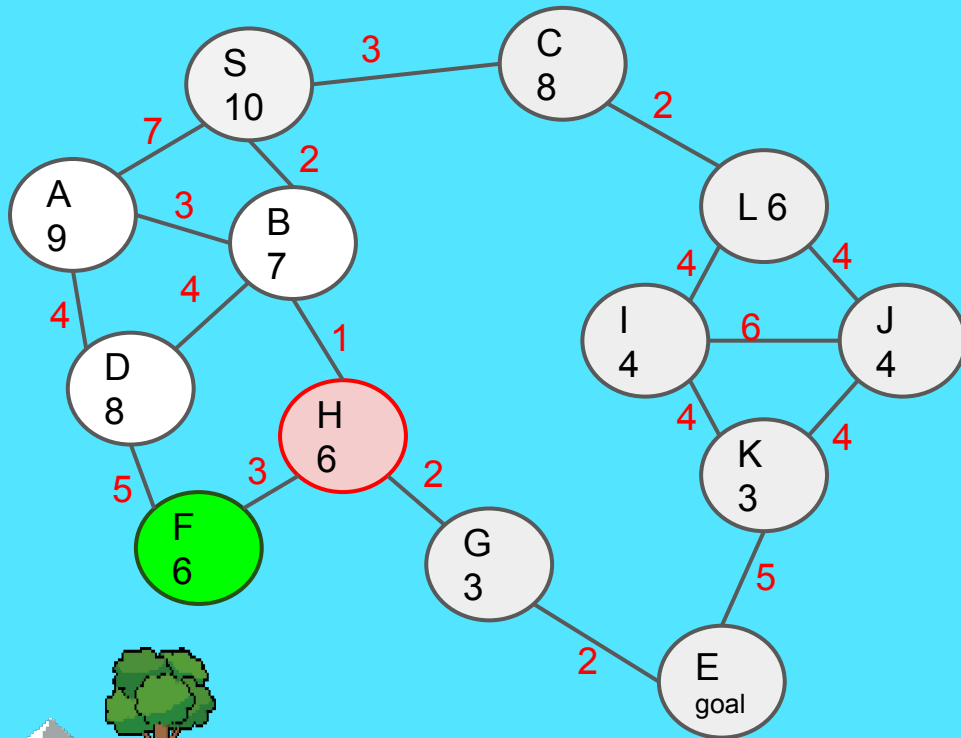
C	distance	Combined heuristic
	3	11

A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

Expanding "H"

## Example



## Queue node

H	distance	Combined heuristic
	3	9

C	distance	Combined heuristic
	3	11

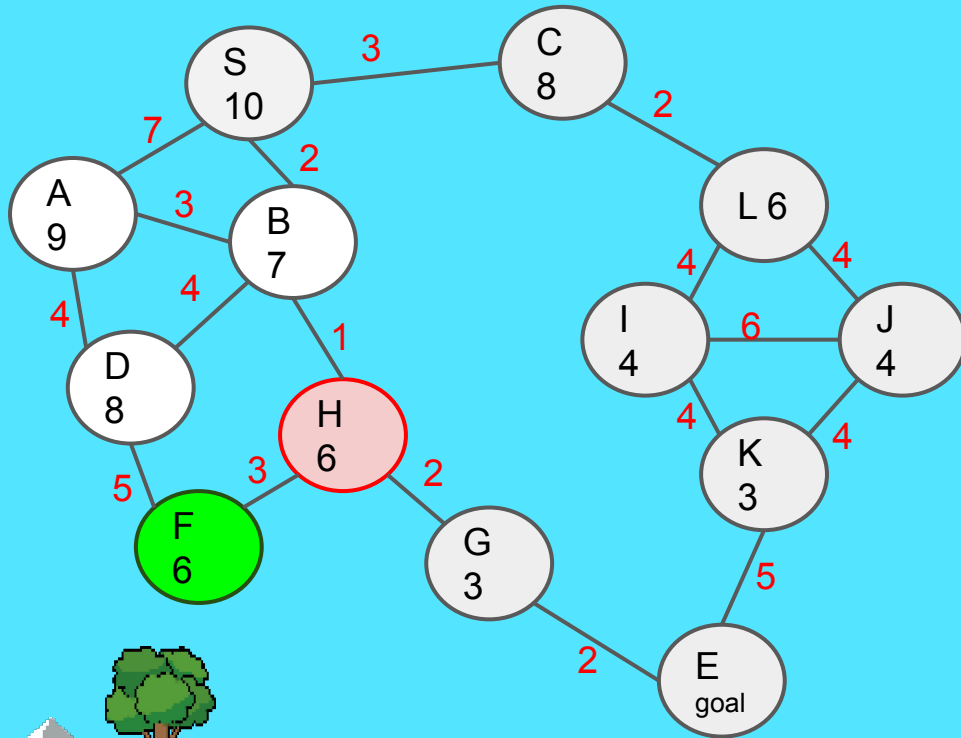
A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

F	distance	Combined heuristic
	6	12

Add "F"

# Example



## Queue node

H	distance	Combined heuristic
	3	9

C	distance	Combined heuristic
	3	11

F	distance	Combined heuristic
	6	12

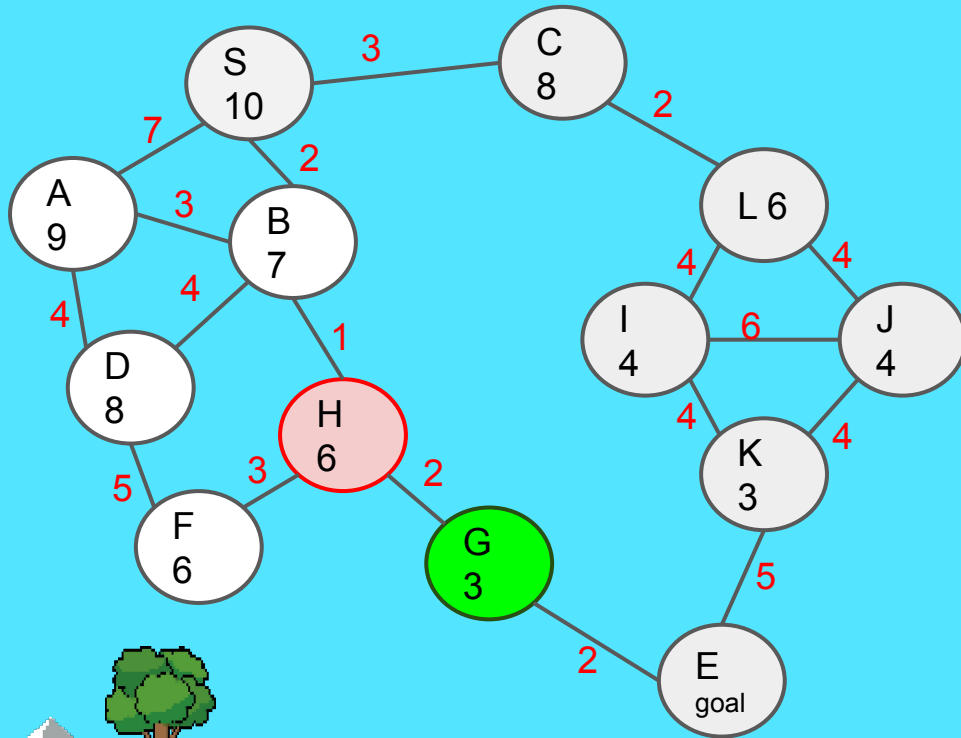
A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

"F" goes above



## Example



## Queue node

H	distance	Combined heuristic
	3	9

C	distance	Combined heuristic
	3	11

F	distance	Combined heuristic
	6	12

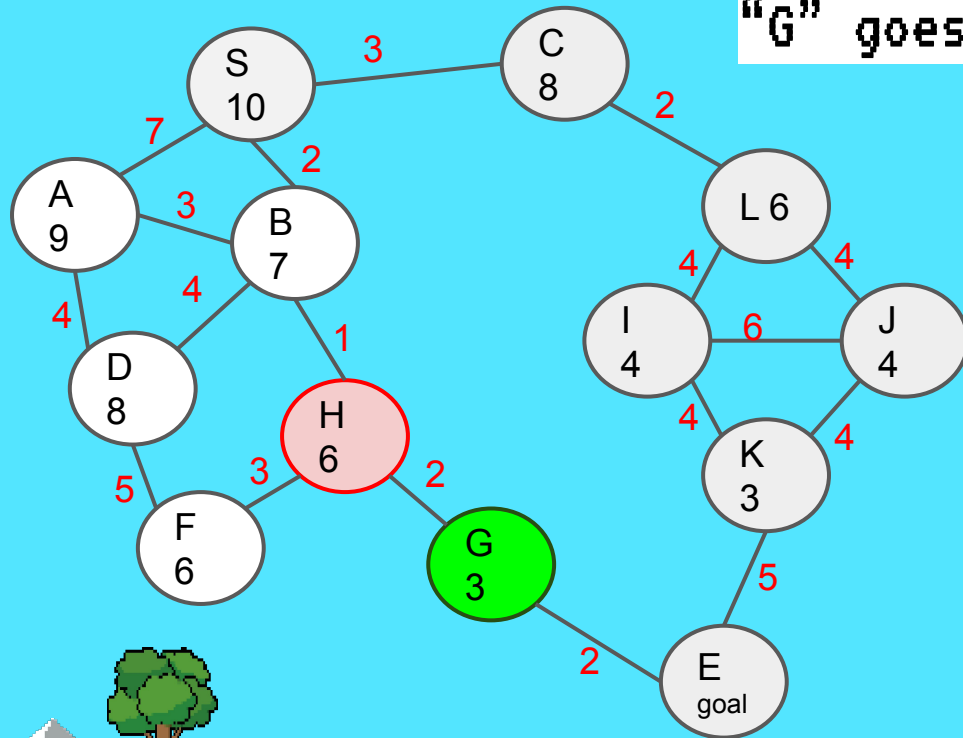
A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

G	distance	Combined heuristic
	5	8

Add "G"

## Example



"G" goes above

## Queue node

G	distance	Combined heuristic
	5	8

H	distance	Combined heuristic
	3	9

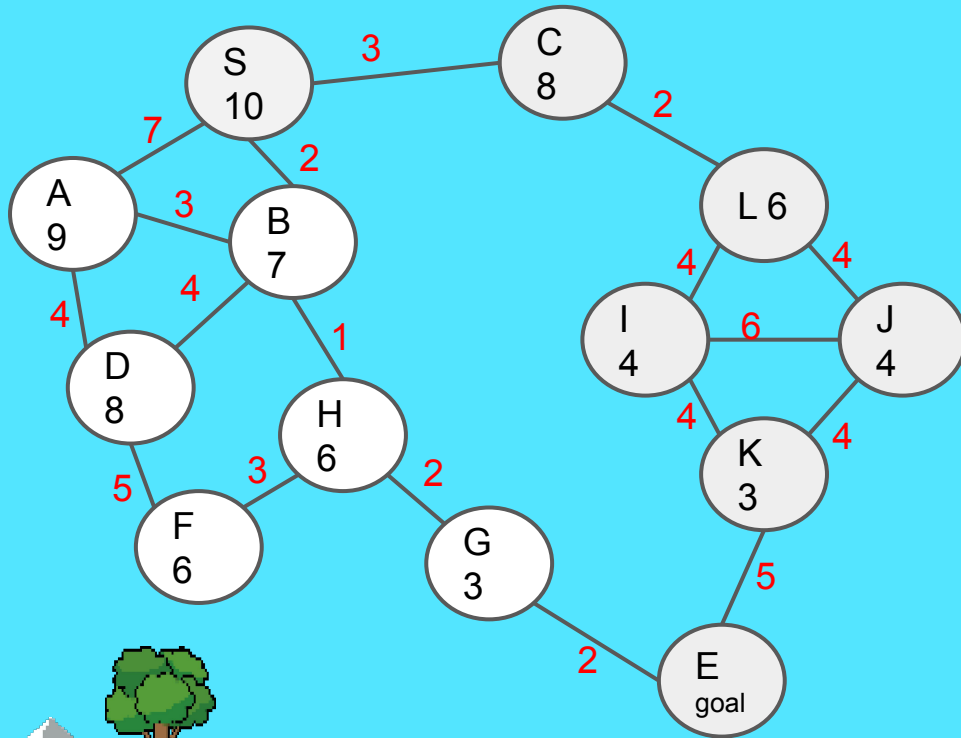
C	distance	Combined heuristic
	3	11

F	distance	Combined heuristic
	6	12

A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

## Example



## Queue node

G	distance	Combined heuristic
	5	8

C	distance	Combined heuristic
	3	11

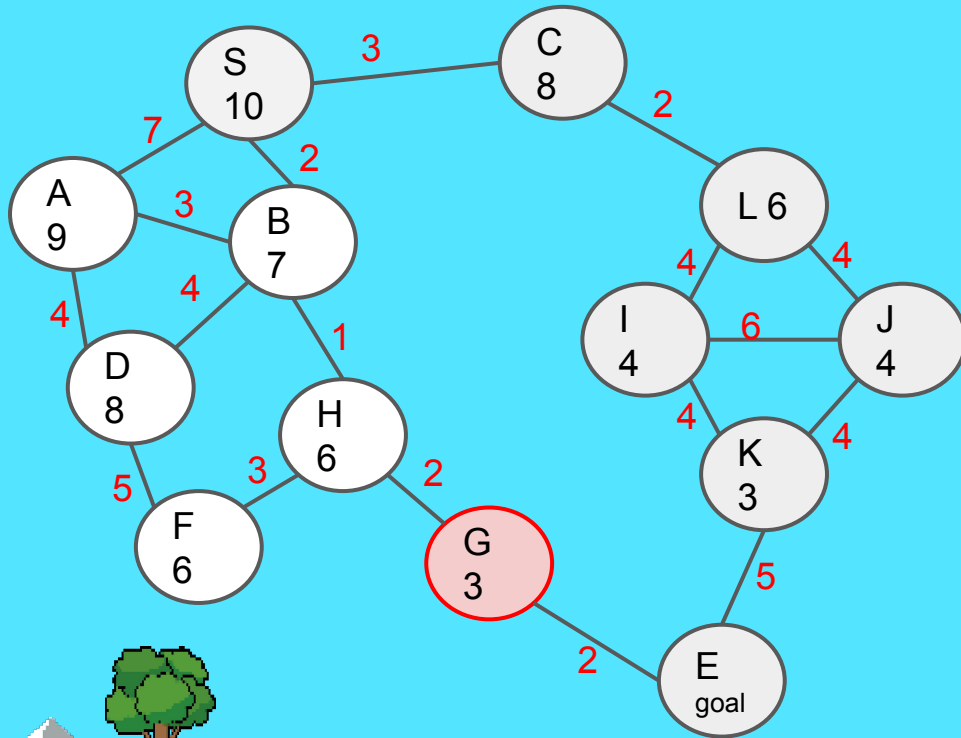
F	distance	Combined heuristic
	6	12

A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

"H" is popped

# Example



## Queue node

G	distance	Combined heuristic
	5	8

C	distance	Combined heuristic
	3	11

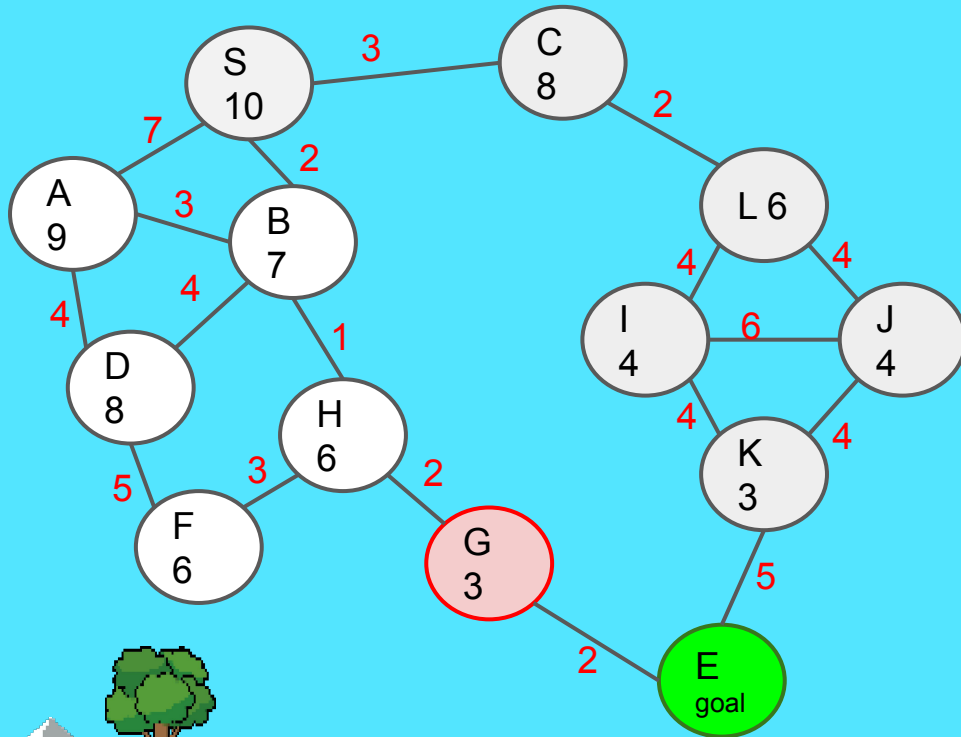
F	distance	Combined heuristic
	6	12

A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

Expanding "G"

# Example



## Queue node

G	distance	Combined heuristic
	5	8

C	distance	Combined heuristic
	3	11

F	distance	Combined heuristic
	6	12

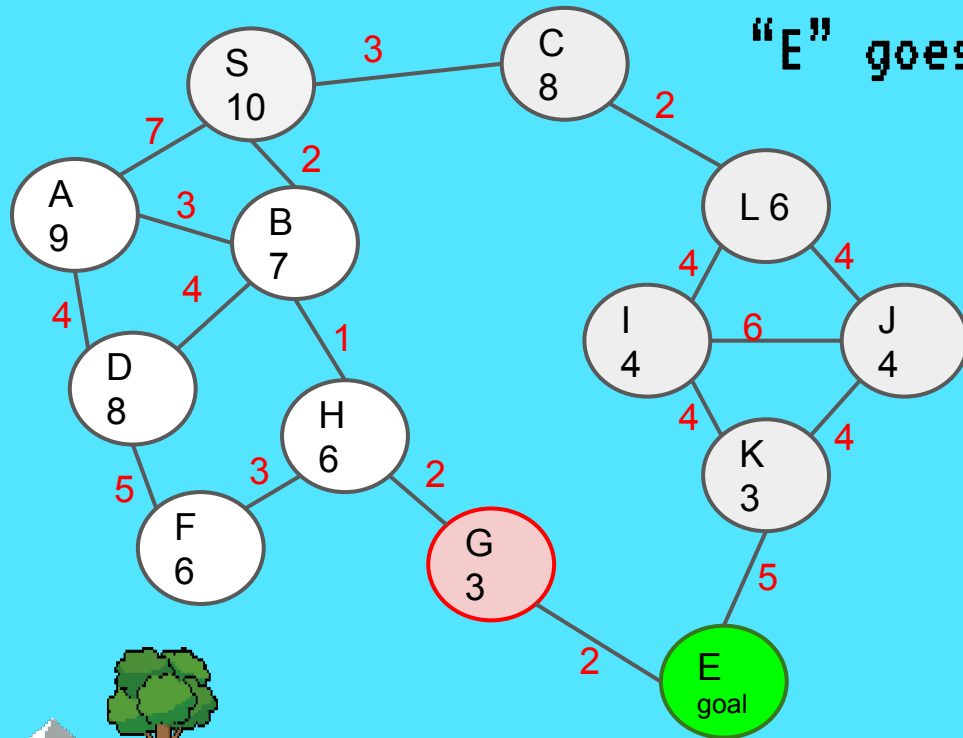
A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

E	distance	Combined heuristic
	7	0

Add "E"

# Example



“E” goes above

## Queue node

E	distance	Combined heuristic
	7	0

G	distance	Combined heuristic
	5	8

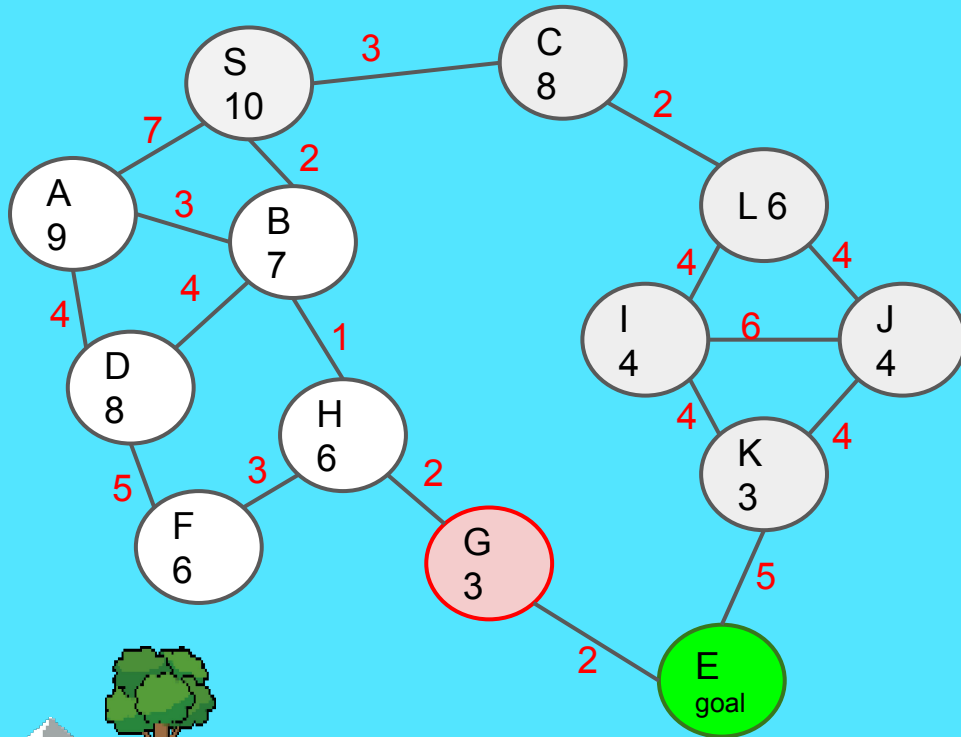
C	distance	Combined heuristic
	3	11

F	distance	Combined heuristic
	6	12

A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

## Example



## Queue node

E	distance	Combined heuristic
	7	0

C	distance	Combined heuristic
	3	11

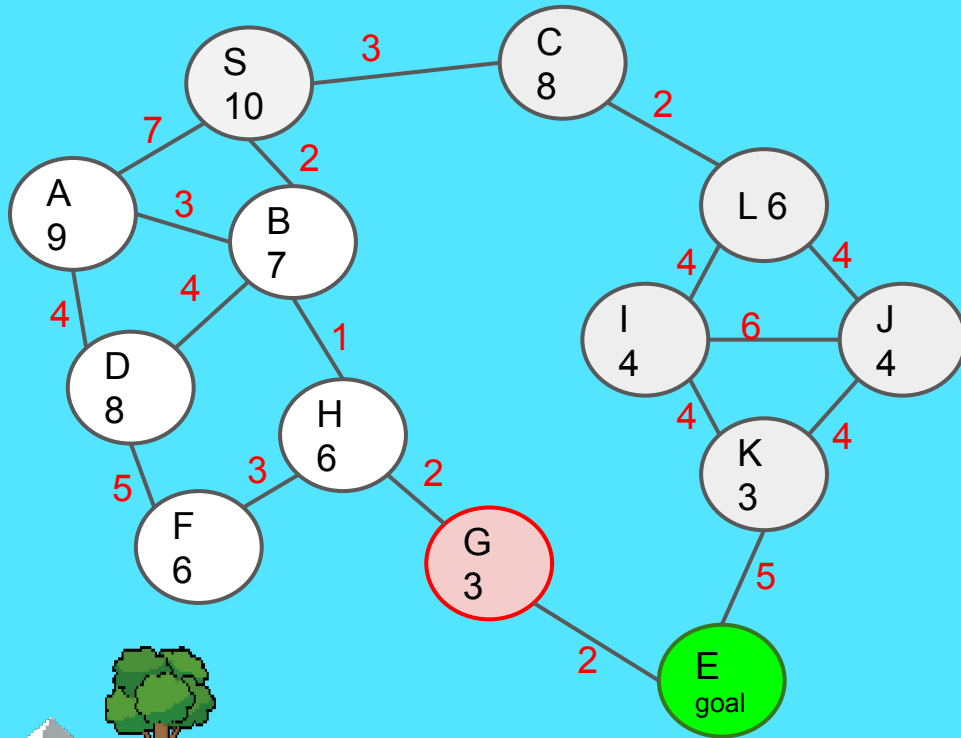
F	distance	Combined heuristic
	6	12

A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

"G" is popped

## Example



## Queue node

C	distance	Combined heuristic
	3	11

F	distance	Combined heuristic
	6	12

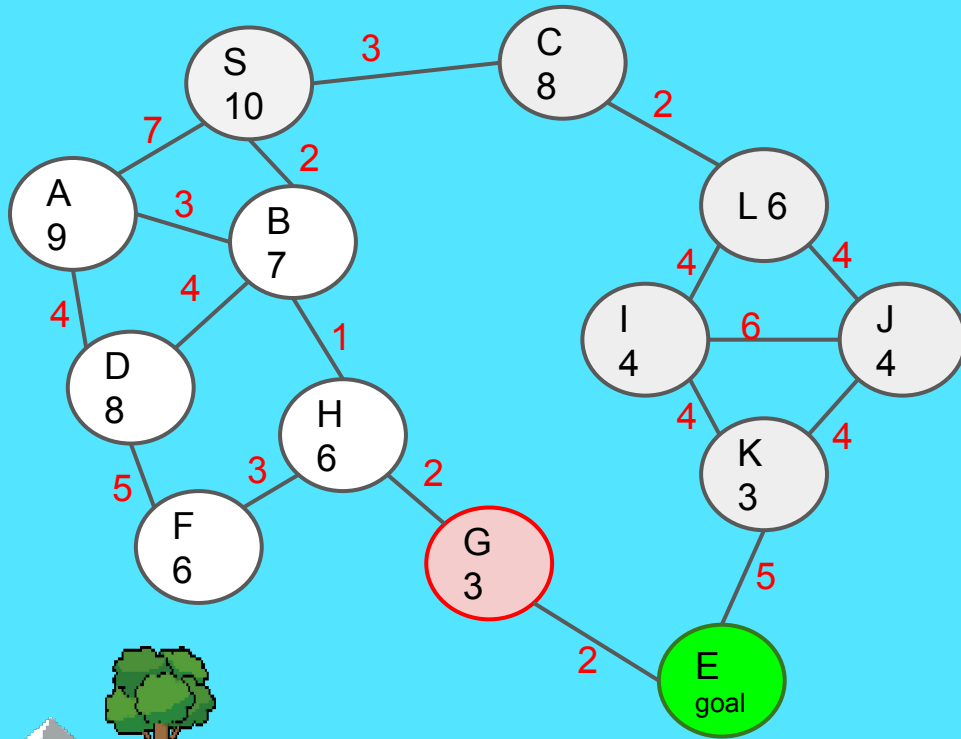
A	distance	Combined heuristic
	5	14

D	distance	Combined heuristic
	6	14

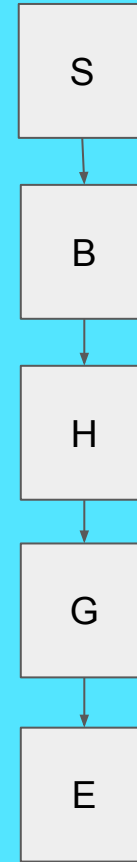
"E" is popped



## Example



## Path list



```
def a_star(cur_state, visited, itr):
    print('A* Initial ', end='')
    cur_state.print_state()

    hp = []
    heapq.heappush(hp, (0+cur_state.h(), 0, cur_state))
    parent = {}
    while(len(hp) != 0):
        itr[0]+=1
        top_ele = heapq.heappop(hp)
        cst = top_ele[2]
        print('cst here',cst)
        if cst.is_goal_state():
            itr[1] = top_ele[1]
            print_path(parent, cst, cur_state)
            print('okay')
            return cst
        cur_steps = top_ele[1]
        for state in cst.generate_states():
            y = repr(state.state)
            if y not in visited:
                visited.add(y)
                parent[y] = cst
                cost = cur_steps+1+state.h()
                heapq.heappush(hp, (cost, cur_steps+1, state))

    return None
```

Show initial state



```
def a_star(cur_state, visited, itr):
    print('A* Initial ', end='')
    cur_state.print_state()

    hp = []
    heapq.heappush(hp, (0+cur_state.h(), 0, cur_state))
    parent = {}

    while(len(hp) != 0):
        itr[0]+=1
        top_ele = heapq.heappop(hp)
        cst = top_ele[2]
        print('cst here',cst)
        if cst.is_goal_state():
            itr[1] = top_ele[1]
            print_path(parent, cst, cur_state)
            print('okay')
            return cst
        cur_steps = top_ele[1]
        for state in cst.generate_states():
            y = repr(state.state)
            if y not in visited:
                visited.add(y)
                parent[y] = cst
                cost = cur_steps+1+state.h()
                heapq.heappush(hp, (cost, cur_steps+1, state))

    return None
```

Push cost and state



```
def a_star(cur_state, visited, itr):
    print('A* Initial ', end='')
    cur_state.print_state()

    hp = []
    heapq.heappush(hp, (0+cur_state.h(), 0, cur_state))
    parent = {}
    while(len(hp) != 0):
        itr[0]+=1
        top_ele = heapq.heappop(hp)
        cst = top_ele[2]
        print('cst here',cst)
        if cst.is_goal_state():
            itr[1] = top_ele[1]
            print_path(parent, cst, cur_state)
            print('okay')
            return cst
        cur_steps = top_ele[1]
        for state in cst.generate_states():
            y = repr(state.state)
            if y not in visited:
                visited.add(y)
                parent[y] = cst
                cost = cur_steps+1+state.h()
                heapq.heappush(hp, (cost, cur_steps+1, state))

    return None
```

Check if the goal is reached



```
def a_star(cur_state, visited, itr):
    print('A* Initial ', end='')
    cur_state.print_state()

    hp = []
    heapq.heappush(hp, (0+cur_state.h(), 0, cur_state))
    parent = {}
    while(len(hp) != 0):
        itr[0]+=1
        top_ele = heapq.heappop(hp)
        cst = top_ele[2]
        print('cst here',cst)
        if cst.is_goal_state():
            itr[1] = top_ele[1]
            print_path(parent, cst, cur_state)
            print('okay')
            return cst
        cur_steps = top_ele[1]
        for state in cst.generate_states():
            y = repr(state.state)
            if y not in visited:
                visited.add(y)
                parent[y] = cst
                cost = cur_steps+1+state.h()
                heapq.heappush(hp, (cost, cur_steps+1, state))

    return None
```

Loop state


Check if state is visited

Get cost of the state

Then push the state



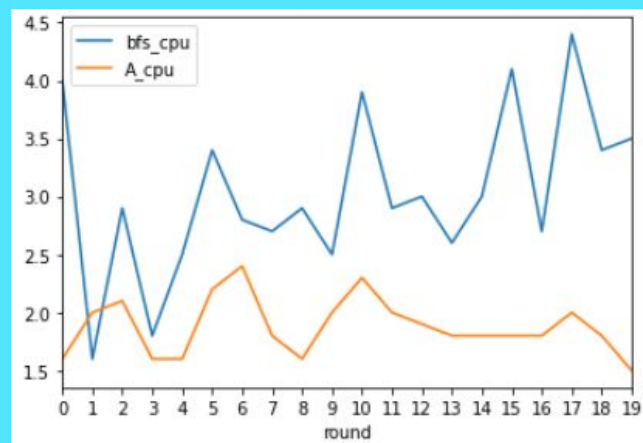
# Statistics

The image features a pixel art style background. A large, white rectangular sign with a black border and a slight drop shadow is centered in the upper half of the frame. The word "Statistics" is written on the sign in a black, monospaced, typewriter-style font. The background consists of a bright blue sky with two white, pixelated clouds. Below the sky, there are blue mountains with white snow-capped peaks. In the foreground, there is a green grassy field and a brown dirt ground. On the left side, there is a brown stone tower with a red flag on top. On the right side, there is a brown stone wall with a crenelated top. Two green trees are also visible, one on the left and one on the right.

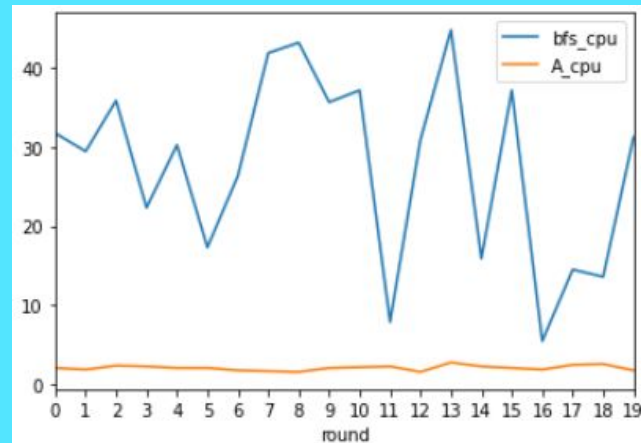
# Compare CPU

## Board Size

3x3



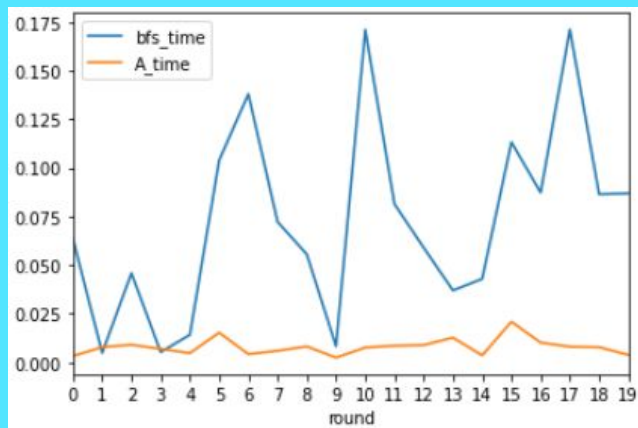
4x4



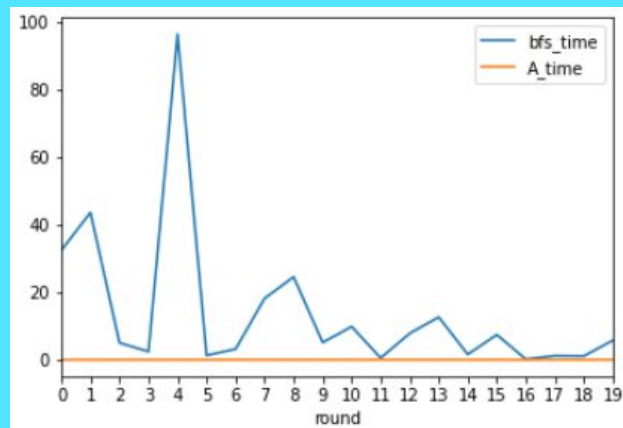
# Compare Time

## Board Size

3x3



4x4

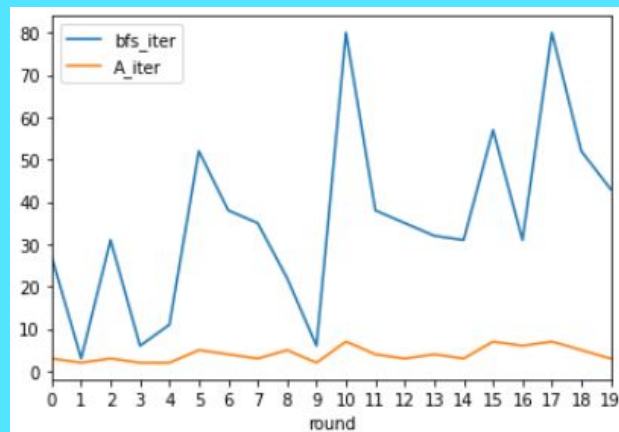




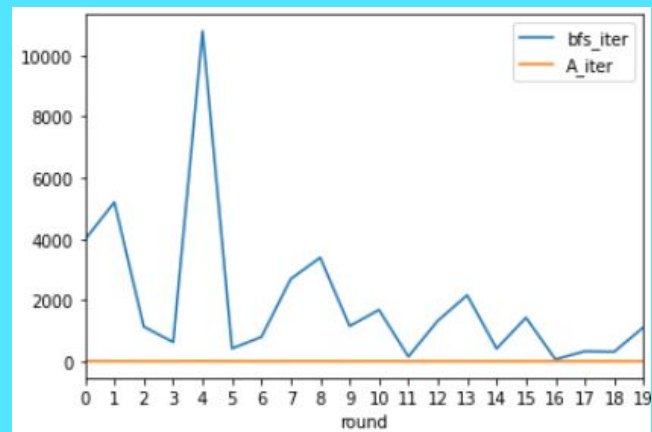
# Compare Iteration

## Board Size

3x3



4x4



A pixel art illustration of a landscape. In the background, there are blue mountains with white snow-capped peaks under a bright blue sky with two white, pixelated clouds. The foreground features a green grassy field above a brown dirt layer. On the left, a stone tower with a red conical roof and a small red flag stands next to a green tree. On the right, another stone tower with a crenelated top stands next to a smaller green tree. In the center, a large white rectangular sign with a black border and a slight 3D effect contains the text "Thanks!".

Thanks!