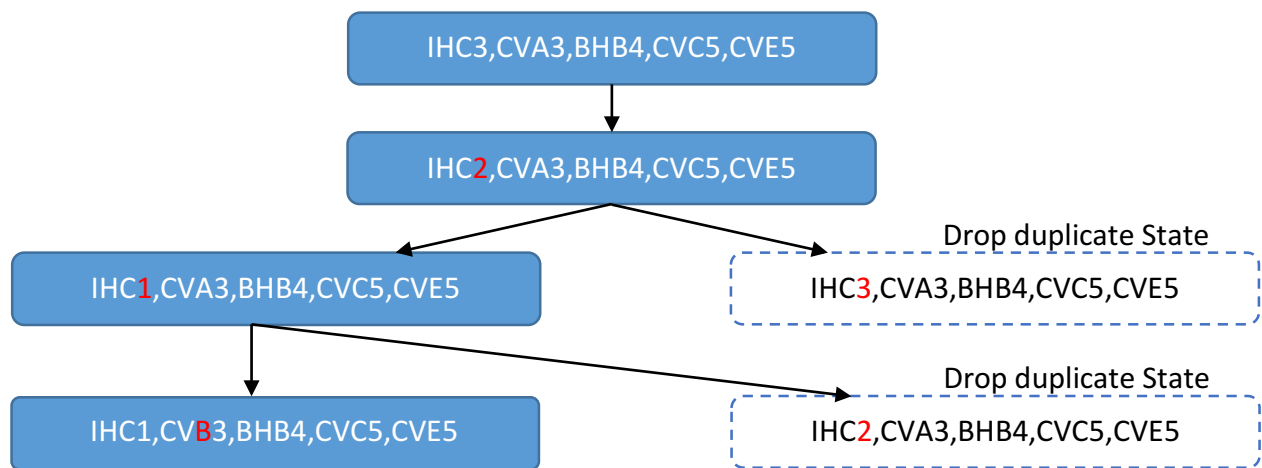


Exercise 1: Rush Hour

1. The tree tree for the limit 3 iteration for Easy 1.

On my code, I prune the branch which have the visited state and need more steps to reach that state. So I did the same thing on the image of my tree.



2. The depth of the tree for Easy 1 and 5 and the number of visited nodes.

DFS:

Image 1: IHC3,CVA3,BHB4,CVC5,CVE5



```
By Deep-First-Search:
---*---
---****
---***-
---*---
---*---
---*---
---*---
The ice chuck position is C3
[1 <- 1, 1 <- 1, 2 ↑ 1, 2 ↑ 1, 2 ↑ 1, 1 → 1, 1 → 1, 3 <- 1, 3 <- 1, 4 ↑ 1, 4 ↑ 1, 1 → 1, 1 → 1, 1 → 1, 1 → 1]
The total number of visited nodes is 580
The cost of Move ice truck is 15
```

The depth of the tree for Easy 1 is 15, and the number of visited nodes is 580.

Image 5: IHC4,CVA4,CVD4,BVA6,CHE5



```
By Deep-First-Search:
---*---
---*---
---***-
---*---
---***-
---*---
---*---
The ice chuck position is C4
[1 <- 1, 1 <- 1, 3 ↑ 1, 4 ↑ 1, 5 <- 1, 4 ↑ 1, 4 ↑ 1, 5 <- 1, 5 <- 1, 3 ↑ 1, 1 → 1, 1 → 1, 1 → 1, 1 → 1, 1 → 1]
The total number of visited nodes is 1726
The cost of Move ice truck is 15
```

The depth of the tree for Easy 5 is 15, and the number of visited nodes is 1726.

3. The number of visited nodes for SMA* algorithm for all images and the effective branching factor.

SMA*

Image 1: IHC3,CVA3,BHB4,CVC5,CVE5



```
By SMA*:
---*---
---****
---***-
---*-
---*-
---*-
---*-
---*-
The ice chuck position is C3
1 <- 1, 1 <- 1, 2 ↑ 1, 2 ↑ 1, 2 ↑ 1, 1 → 1, 1 → 1, 3 <- 1, 3 <- 1, 4 ↑ 1, 4 ↑ 1, 1 → 1, 1 → 1, 1 → 1, 1 → 1
The cost of Move ice truck is 15
The total number of visited nodes is 168
```

The number of visited nodes for SMA* algorithm for images 1 is 168. And the effective branching factor is 1.27.

Image 5: IHC4,CVA4,CVD4,BVA6,CHE5



```
By SMA*:
---*---
---*---
---*---
---*---
---*---
---*---
---*---
The ice chuck position is C4
1 <- 1, 1 <- 1, 3 ↑ 1, 5 <- 1, 5 <- 1, 5 <- 1, 3 ↑ 1, 1 → 1, 4 ↑ 1, 4 ↑ 1, 4 ↑ 1, 1 → 1, 1 → 1, 1 → 1, 1 → 1
The cost of Move ice truck is 15
The total number of visited nodes is 429
```

The number of visited nodes for SMA* algorithm for images 5 is 429. And the effective branching factor is 1.37.

Image 9: IHC4,CVD5,BVC6,CVE3,BHF4,BHD1



```
By SMA*:
-----
*****
-----**
*****
-----**
*****
-----
The ice chuck position is C4
1 <- 1, 2 + 1, 2 + 1, 2 + 1, 1 -> 1, 3 + 1, 3 + 1, 6 -> 1, 6 -> 1, 6 -> 1, 4 + 1, 4 + 1, 4 + 1, 6 <- 1, 3 + 1, 5 <- 1, 3 + 1, 3 + 1, 1 -> 1, 1 -> 1, 1 -> 1
The cost of Move ice truck is 21
The total number of visited nodes is 1123
```

The number of visited nodes for SMA* algorithm for images 9 is **1123**. And the effective branching factor is **1.30**.

Image 11: IHC1,CVC3,BHE1,BHA2,CHA5,CHB5,CVC6,CVE6



```
By SMA*:
-----
*****
-----**
*****
-----**
*****
-----
The ice chuck position is C1
6 <- 1, 7 + 1, 4 <- 1, 5 <- 1, 7 + 1, 8 + 1, 3 -> 1, 3 -> 1, 8 + 1, 3 -> 1, 2 + 1, 1 -> 1, 1 -> 1, 1 -> 1, 2 + 1, 3 <- 1, 8 + 1, 1 -> 1, 1 -> 1, 1 -> 1
The cost of Move ice truck is 20
The total number of visited nodes is 1668
```

The number of visited nodes for SMA* algorithm for images 11 is **1668**. And the effective branching factor is **1.36**.

4. The new heuristic and its discussion.

I found out that it's unnecessary to the ice-cream truck moves forward immediately. Look at the original heuristic:

$$h(n) = D(n) + M(n)$$

$D(n)$: the number of steps the ice-cream truck need to take to get to the output,

$M(n)$: the number of vehicles blocking row C that needs to move to make room for the ice-cream truck.

I found that the original heuristic would lead the ice-cream truck to immediately move forward when the truck has space in front of its.

Undoubtedly, the ice-cream truck should be led to output. But as we know, there should leave some spaces to allow moving the vehicle blocking the row C. We don't have to immediately move ice-cream truck when it can move forward.

Compare to move ice-cream truck forward one grid, using two or even more costs to remove one vehicle blocking row C should be a better choice. So we need to adjust the weight of two factors in the $h(n)$, the **new heuristic**:

$$h(n) = D(n) + 2M(n)$$

Now, we test the new heuristic's performance. Using the image 9:

When we use the original heuristic:

```
By SMA*:
-----
-----
-----*---
*---*---
*---*---
*---*---
-----*---
-----*---
The ice chuck position is C4
1 <- 1, 2 ↑ 1, 2 ↑ 1, 2 ↑ 1, 1 → 1, 3 ↑ 1, 3 ↑ 1, 6 → 1, 6 → 1, 6 → 1, 4 ↑ 1, 4 ↑ 1, 4 ↑ 1, 6 <- 1, 3 ↓ 1, 5 <- 1, 3 ↓ 1, 3 ↓ 1, 1 → 1, 1 → 1, 1 → 1
The cost of Move ice truck is 21
The total number of visited nodes is 1123
```

When we use the new heuristic:

```
By SMA*:
-----
-----
-----*---
*---*---
*---*---
*---*---
-----*---
-----*---
The ice chuck position is C4
1 <- 1, 2 ↑ 1, 2 ↑ 1, 2 ↑ 1, 1 → 1, 3 ↑ 1, 6 → 1, 6 → 1, 3 ↑ 1, 6 → 1, 4 ↑ 1, 4 ↑ 1, 4 ↑ 1, 6 <- 1, 3 ↓ 1, 3 ↓ 1, 5 <- 1, 3 ↓ 1, 1 → 1, 1 → 1, 1 → 1
The cost of Move ice truck is 21
The total number of visited nodes is 993
```

The new heuristic still didn't find the best result, but it only visited 993 nodes to find the same result! The original heuristic need to visit 1123 nodes.

5. Discuss what would happen (tree structure, depth, previous heuristic ...) if the action is changed to moving one vehicle any number of grid elements and the cost is changed to per vehicle movement independent of the number of advanced grid units.

a. Tree Structure and depth.

Because per vehicle can move more than one grid on every step, every node would have more child nodes than before. Every level of tree would have more nodes. And there would be less steps to move the ice-cream truck to output. The depth of tree would be much smaller.

b. Previous heuristic

I thought that previous heuristic and SMA* couldn't find the best result which is the same as DFS's result. Because the cost is changed to per vehicle movement independent of the number of advanced grid units. The cost of moving ice-cream truck forward have to pay the more cost when the moving split to multiple steps than just being one step. The previous heuristic, strongly leading the ice-cream truck move forward immediately when the truck has space in front of its, wouldn't be a good strategy to search the best result. But SMA* still can turn in a pretty good result (even it's not the best result).

c. Visited nodes

Because the numbers of nodes of each tree level was greatly increase, the heuristic can filter more nodes on each tree level. And there would be less depth to move the ice-cream truck to output. The ice-truck could get more close to the output on one iteration than before.

Therefore, the number of visited nodes would be much smaller.

Exercise 2: Sudoku

1. The solution for SuDoKu in the image when using my code.

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

```

There are 0 blanks
4 8 3 9 2 1 6 5 7
9 6 7 3 4 5 8 2 1
2 5 1 8 7 6 4 9 3
5 4 8 1 3 2 9 7 6
7 2 9 5 6 4 1 3 8
1 3 6 7 9 8 2 4 5
3 7 2 6 8 9 5 1 4
8 1 4 2 5 3 7 6 9
6 9 5 4 1 7 3 8 2
  
```

Figure 2: Image for Exercise 2.

2. The messages that variable A1 sends to its three factors and the message that the factors send to A1 in each iteration.

All the messages were normalized. Each message is a vector which indicates that the probabilities of this cell (A1) is filled by x , $x \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Step 1.

```

0 0 3 0 2 0 6 0 0
9 0 0 3 0 5 0 0 1
0 0 1 8 0 6 4 0 0
0 0 8 1 0 2 9 0 0
7 0 0 0 0 0 0 0 8
0 0 6 7 0 8 2 0 0
0 0 2 6 0 9 5 0 0
8 0 0 2 0 3 0 0 9
0 0 5 0 1 0 3 0 0
  
```

```

There are 49 blanks
The A1 sent to its factor messages are:
Row: 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111
Col: 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111
Box: 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111
The messages that the factors sent to A1 are:
Row: 0.16666666 0.0 0.0 0.16666666 0.16666666 0.0 0.16666666 0.16666666 0.16666666
Col: 0.16666666 0.16666666 0.16666666 0.16666666 0.16666666 0.16666666 0.0 0.0 0.0
Box: 0.0 0.16666666 0.0 0.16666666 0.16666666 0.16666666 0.16666666 0.16666666 0.0
  
```

Step 2.

```
0 0 3 0 2 0 6 0 0
9 0 0 3 0 5 0 0 1
0 0 1 8 0 6 4 0 0
0 0 8 1 0 2 9 0 0
7 0 0 0 0 4 1 0 8
0 0 6 7 0 8 2 0 0
0 0 2 6 0 9 5 0 0
8 0 0 2 0 3 0 0 9
0 0 5 4 1 0 3 0 0
```

There are 46 blanks

The A1 sent to its factor messages are:

Row: 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0

Col: 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0

Box: 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0

The messages that the factors sent to A1 are:

Row: 0.0 0.0 0.0 0.6666667 0.33333334 0.0 0.0 0.0 0.0

Col: 0.0 0.0 0.0 0.42857143 0.57142854 0.0 0.0 0.0 0.0

Box: 0.0 0.0 0.0 0.6666666 0.3333333 0.0 0.0 0.0 0.0

Step 3.

```
0 8 3 9 2 1 6 0 0
9 6 0 3 0 5 8 2 1
2 0 1 8 0 6 4 0 0
0 0 8 1 0 2 9 0 0
7 2 9 5 0 4 1 0 8
0 0 6 7 0 8 2 0 0
0 0 2 6 8 9 5 0 0
8 0 0 2 5 3 7 0 9
6 9 5 4 1 7 3 8 2
```

There are 28 blanks

The A1 sent to its factor messages are:

Row: 0.0 0.0 0.0 0.6 0.39999998 0.0 0.0 0.0 0.0

Col: 0.0 0.0 0.0 0.8 0.2 0.0 0.0 0.0 0.0

Box: 0.0 0.0 0.0 0.6 0.39999998 0.0 0.0 0.0 0.0

The messages that the factors sent to A1 are:

Row: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0

Col: 0.0 0.0 0.0 0.82654977 0.1734502 0.0 0.0 0.0 0.0

Box: 0.0 0.0 0.0 0.5538462 0.44615382 0.0 0.0 0.0 0.0

Step 4.

4	8	3	9	2	1	6	0	0
9	6	0	3	4	5	8	2	1
2	0	1	8	7	6	4	0	0
0	0	8	1	0	2	9	0	6
7	2	9	5	0	4	1	0	8
0	0	6	7	9	8	2	0	0
0	0	2	6	8	9	5	1	4
8	1	4	2	5	3	7	6	9
6	9	5	4	1	7	3	8	2

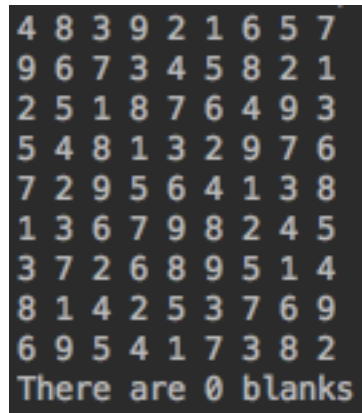
```
There are 18 blanks
The A1 sent to its factor messages are:
Row: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
Col: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
Box: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
The messages that the factors sent to A1 are:
Row: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
Col: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
Box: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
```

Step 5.

4	8	3	9	2	1	6	0	0
9	6	7	3	4	5	8	2	1
2	5	1	8	7	6	4	9	3
5	0	8	1	3	2	9	7	6
7	2	9	5	0	4	1	3	8
0	0	6	7	9	8	2	0	5
3	7	2	6	8	9	5	1	4
8	1	4	2	5	3	7	6	9
6	9	5	4	1	7	3	8	2

```
There are 7 blanks
The A1 sent to its factor messages are:
Row: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
Col: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
Box: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
The messages that the factors sent to A1 are:
Row: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
Col: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
Box: 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
```

Step 6.



4	8	3	9	2	1	6	5	7
9	6	7	3	4	5	8	2	1
2	5	1	8	7	6	4	9	3
5	4	8	1	3	2	9	7	6
7	2	9	5	6	4	1	3	8
1	3	6	7	9	8	2	4	5
3	7	2	6	8	9	5	1	4
8	1	4	2	5	3	7	6	9
6	9	5	4	1	7	3	8	2
There are 0 blanks								

Calculation has completed.

3. Discuss a way in which the code can be improved to solve SuDoKus with less known values.

As we know, the BP Algorithm can solve most of simple Sukodus. On my code, only the probability of cell which filled by certain value equal to 1, I would fill the cell with that value. If there are less known values at the beginning, we may meet one problem:

After iteration, there is not any cell's probability updated, and all the blank cells still have the multiple candidates can be filled.

On my code, the BP Algorithm will stop iteration when it meets that situation. So I figure out a way in which the code can be improved to solve that problem: **“BP Algorithm + Priority DFS”**.

- 1) The BP Algorithm has stopped iteration, and there still are some blank cells.
- 2) Travel all the blank cells, find the cell which have the minimum size of set of available value. The cell, denoted as c_m . The set of c_m 's available value, denoted as V_m .
- 3) Push the current board's state as a node into a stack, denoted as S. The node include the probabilities of all cells' available value, c_m , and V_m .
- 4) Read the board state from the stack S's peek node.
- 5) Travel all available value in V_m . Find the $X_0, X_0 \in V_m$, which is the most probable value in c_m . Fill the cell c_m with X_0 .
- 6) Run BP Algorithm under the new board state.
- 7) When the BP Algorithm stop iteration:

- a. If all cells are filled and meet the requirement, succeed, exit.
 - b. If some blank cells can't be filled by any value without violation of constraints. Go to 8).
 - c. If all blank cells have multiple available value, Go to 2).
- 8) Modify the peek node of stack S, remove X_0 from V_m , and update the board state, including setting $P_{(C_m = X_0)} = 0$ and renormalized the probabilities of C_m 's available value.
- a. If the V_m is empty, that means this branch is a dead end. Pop the peek node from stack S.
 - 1. If stack S is empty, fail, exit.
 - 2. If stack S wasn't empty. Read the peek node on stack S. Go to 8).
 - b. If the V_m isn't empty, need to find the new X_0 . Go to 5).