



Unit 33.

Backtracking

● Learning objectives

- ✓ ນັກສຶກສາຈະສາມາດເຂົ້າໃຈເຕັກນິກການ backtracking ແລະ ແກ້ໄຂບັນຫາ N-Queens ໂດຍໃຊ້ backtracking.
- ✓ ນັກຮຽນຈະສາມາດເຂົ້າໃຈໄດ້ວ່າເຕັກນິກການ backtracking ແມ່ນ depth-first search ໂດຍການເອີ້ນໃຊ້ຟັງຊັນ stack.
- ✓ ນັກຮຽນຈະສາມາດສ້າງຂັ້ນຕອນວິທີຢ່າງມີປະສິດທິພາບໂດຍຜ່ານການປັບແຕ່ງໃນຂະບວນການຄົ້ນຫາແບບ depth-first.

● Learning overview

- ✓ ຮຽນຮູ້ວິທີແກ້ໄຂບັນຫາ N-Queens.
- ✓ ຮຽນຮູ້ຫຼັກການຂອງການແກ້ໄຂ Sum-of-Subsets.
- ✓ ຮຽນຮູ້ວິທີການນຳໃຊ້ເຕັກນິກ backtracking.

● Concepts you will need to know from previous units

- ✓ ສາມາດເຂົ້າໃຈແລະສ້າງການກຳໜົດຂອງຟັງຊັນ recursive ແລະ ເງື່ອນໄຂການຢຸດເຮັດວຽກຊ້ຳຄືນ.
- ✓ ສາມາດສ້າງສາຂາທີ່ມີເງື່ອນໄຂໂດຍໃຊ້ປະໂຫຍກເງື່ອນໄຂ.
- ✓ ສາມາດໃຊ້ລາຍການຂອງ Python.

Keywords

Backtracking

**Depth-First
Search**

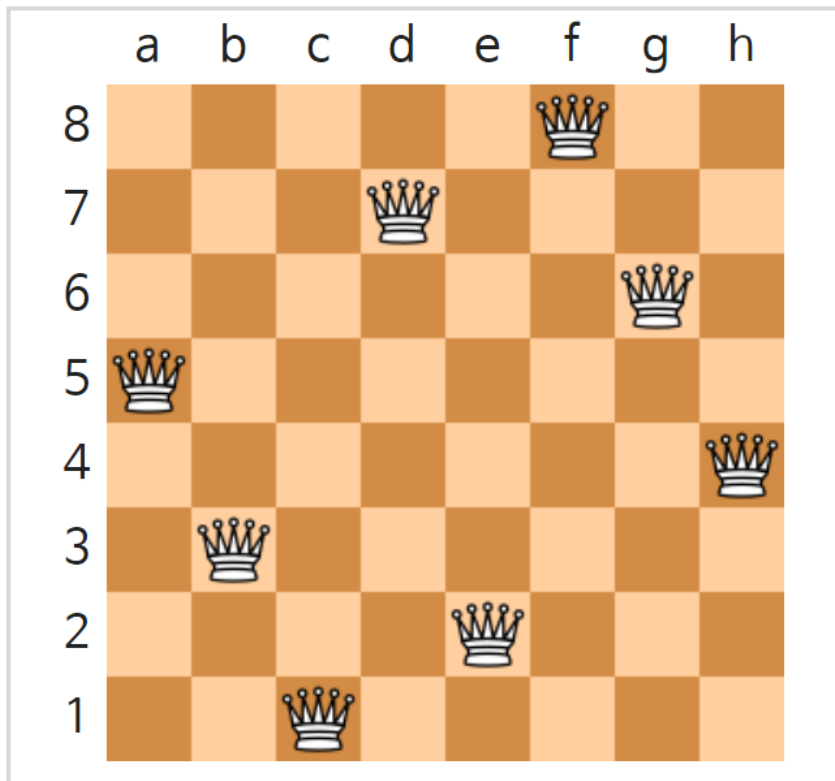
promising

pruning

| Mission

1. Real world problem

1.1. ແກ້ປິດສະໜາ 8 ນາງພະຍາ (The 8-Queens Puzzle)



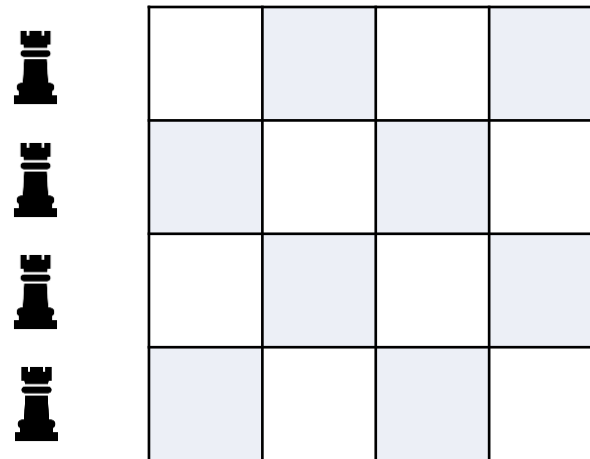
https://en.wikipedia.org/wiki/Eight_queens_puzzle

- ▶ ບັນຫາ 8-Queens ແມ່ນບັນຫາຂອງການວາງ 8 Queens ຢູ່ໃນກະດານ chessboard 8×8 .
- ▶ ໃນຫມາກລຸກ, Queen ສາມາດຍ້າຍຊ້າຍແລະຂວາ, ຂຶ້ນແລະລົງ ແລະ ທາງເສັ້ນເນັ້ງແຈ. ດັ່ງນັ້ນ, ບໍ່ມີ Queen ອື່ນສາມາດຖືກຈັດໃສ່ໃນແຖວ, ຖັນ ຫຼື ເສັ້ນຂວາງຄືກັບບ່ອນທີ່ Queen ອື່ນຖືກວາງໄວ້.
- ▶ ເພື່ອເຮັດໃຫ້ບັນຫານີ້ເປັນບັນຫາໂດຍທົ່ວໄປ, ມັນກາຍເປັນບັນຫາ N-Queens. ນັ້ນແມ່ນ, ມັນເປັນບັນຫາຂອງການຈັດ N-Queens ໃນກະດານ chessboard $N \times N$.
- ▶ ມາສ້າງຂັ້ນຕອນວິທີເພື່ອແກ້ໄຂບັນຫາ N-Queens.

2. Mission

2.1. ບັນຫາ N-Queens

- ບັນຫາ N-Queens ແມ່ນບັນຫາຂອງການວາງ N queens ເທິງກະດານ chessboard $N \times N$.
- ຕົວຢ່າງ, ບັນຫາ 4-Queens ໃນກໍລະນີຂອງ $N = 4$ ແມ່ນບັນຫາຂອງການວາງ 4 Queens ໃນກະດານ chessboard 4×4 .



3. ການແກ້ໄຂບັນຫາ

3.1. ວິທີການທີ່ການແກ້ໄຂບັນຫາ N-Queens ເຮັດວຽກ

- | ບັນຫາ N-Queens ແມ່ນບັນຫາຂອງການວາງ N-queens ຢູ່ໃນກະດານ chessboard $N \times N$.
- | ຕົວຢ່າງ, ບັນຫາ 4-Queens ໃນກໍລະນີຂອງ $N = 4$ ແມ່ນບັນຫາຂອງການວາງ 4 Queens ຢູ່ໃນກະດານ chessboard 4×4 .

```
1 N = int(input("Input the number of queens: "))
2 n_queens(-1, [-1] * N)
```

Input the number of queens: 4

[1, 3, 0, 2]

[2, 0, 3, 1]

3. ການແກ້ໄຂບັນຫາ

3.2. Code ສຸດທ້າຍຂອງການແກ້ບັນຫາ N-Queens

```
1 def n_queens(i, col):
2     if promising(i, col):
3         if i == len(col) - 1:
4             print(col)
5         else:
6             for j in range(len(col)):
7                 col[i + 1] = j
8                 n_queens(i + 1, col)
```

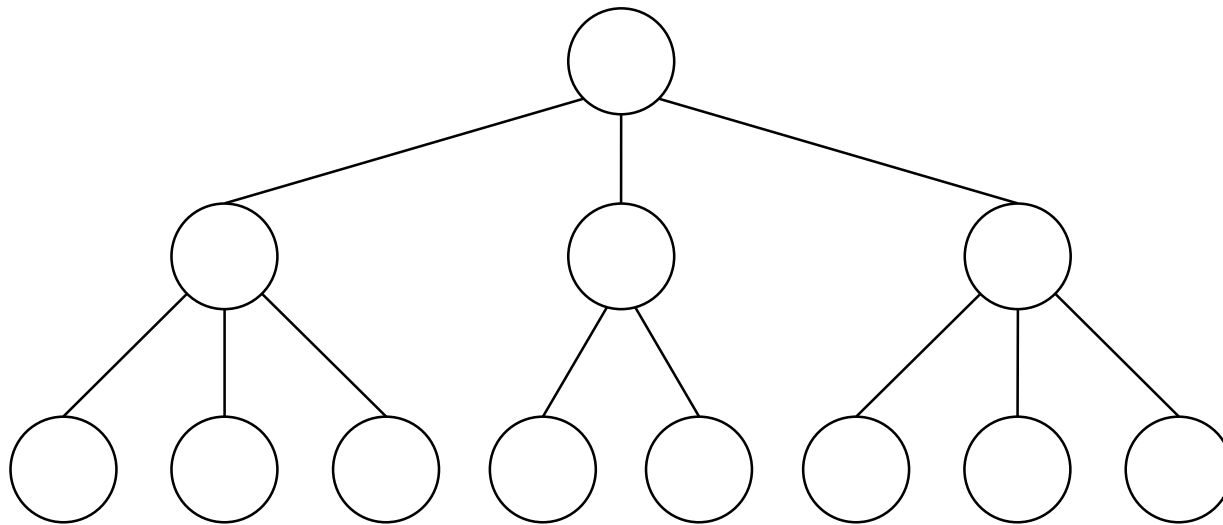
```
1 def promising(i, col):
2     for k in range(i):
3         if col[i] == col[k] or abs(col[i] - col[k]) == (i - k):
4             return False
5     return True
```

| Key concept

1. Backtracking

1.1. Depth First Search

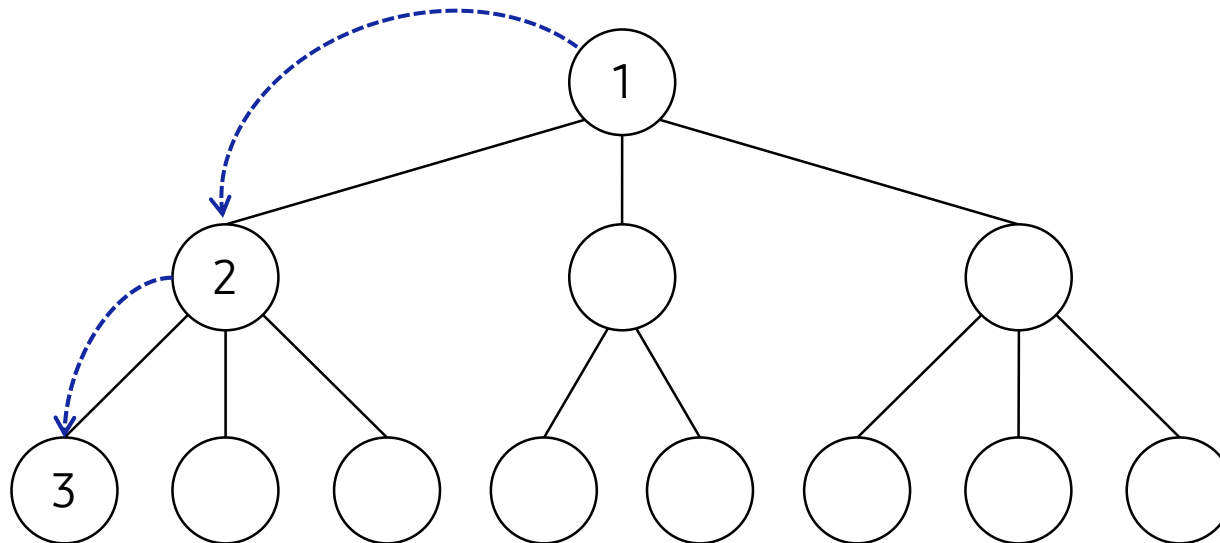
ວິທີການທີ່ສາມາດໄປຢ້ຽມຢາມ nodes ທັງຫມົດຫນຶ່ງຄັ້ງໃນໂຄງສ້າງຕົ້ນໄມ້ດັ່ງທີ່ສະແດງຂ້າງລຸ່ມນີ້ໄດ້ແນວໃດ?



1. Backtracking

1.1. Depth First Search

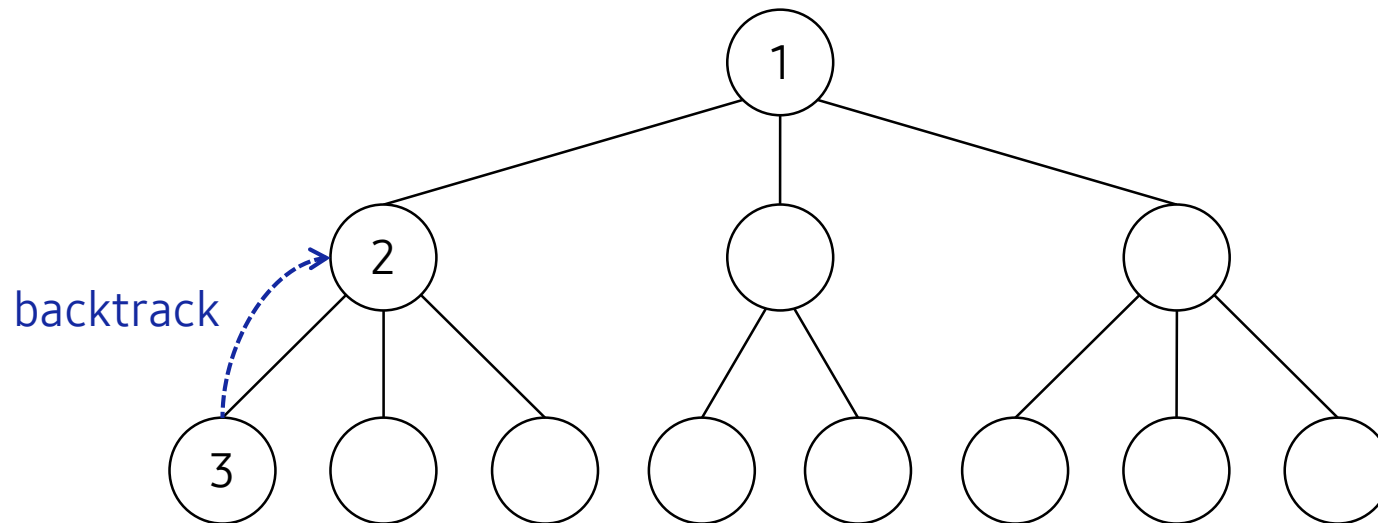
- ການຄົ້ນຫາ Depth-First ຫມາຍເຖິງວິທີການຄົ້ນຫາທີ່ໄປຄົ້ນຫາທາງເລິກກ່ອນ ດັ່ງຕໍ່ໄປນີ້.
- ທໍາອິດ, ໄປຢ້ຽມຢາມ root node ກ່ອນ, ຕິດຕາມດ້ວຍ node 2 ແລະ node 3.



1. Backtracking

1.1. Depth First Search

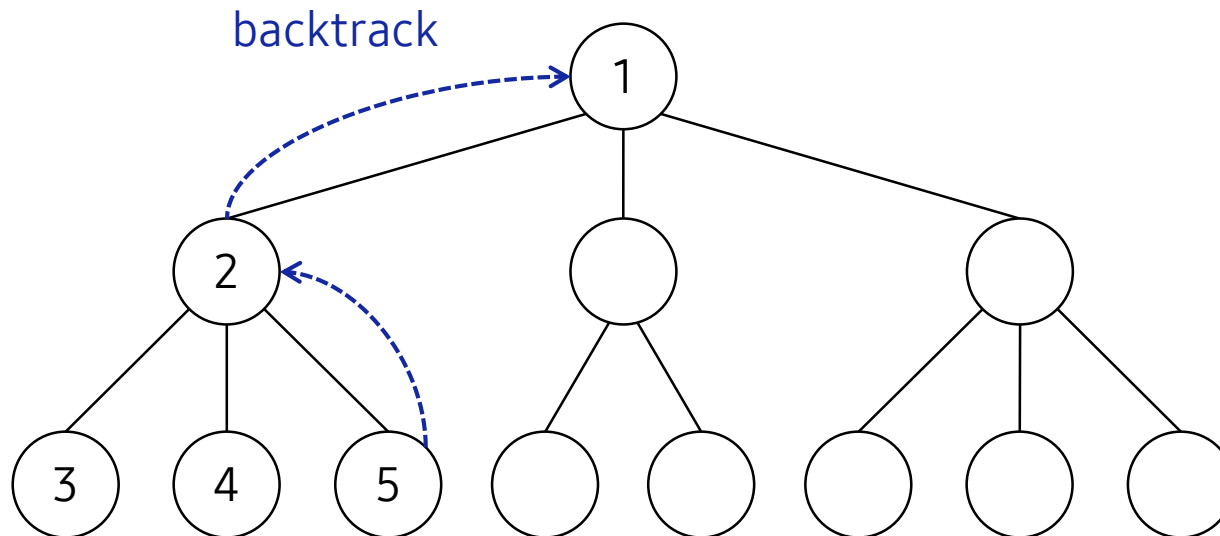
- | Node 3 ບໍ່ມີ node ທີ່ຈະໄປຢ້ຽມຢາມອີກຕໍ່ໄປ, ດັ່ງນັ້ນຕ້ອງກັບຄືນ node ທີ່ຜ່ານມາ.
- | ເນື່ອງຈາກມີ node ທີ່ຍັງເຫຼືອທີ່ຈະໄປຢ້ຽມຢາມຢູ່ node 2, ການຄົ້ນຫາ depth-first ອາດຈະຖືກສືບຕໍ່ເຮັດວຽກ.



1. Backtracking

1.1. Depth First Search

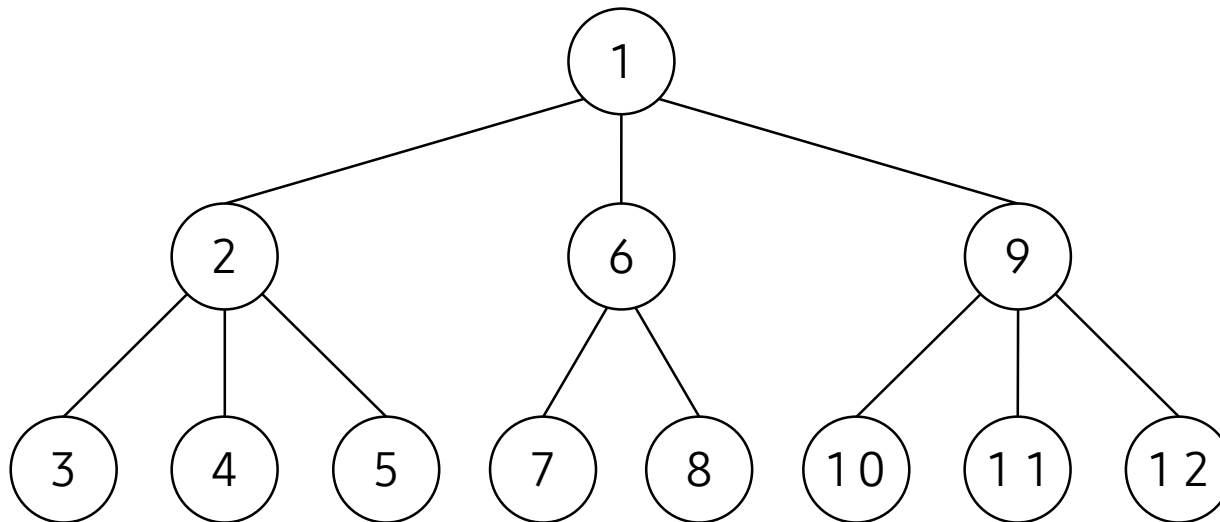
- | ຫຼັງຈາກຢ້ຽມຢາມ nodes 4 ແລະ 5, ມັນກັບຄືນສູ່ 2.
- | ເນື່ອງຈາກ node ຍ່ອຍທັງໝົດຂອງ node 2 ໄດ້ຖືກເຂົ້າຫາແລ້ວ, ພວກມັນຈະຕ້ອງກັບຄືນໄປຫາ node 1.



1. Backtracking

1.1. Depth First Search

- ດ້ວຍວິທີນີ້, ທຸກໆ nodes ສາມາດເຂົ້າໄປຫາຕາມລຳດັບຕໍ່ໄປນີ້.
- ຕົວເລກຂອງແຕ່ລະ node ແມ່ນລຳດັບຂອງ nodes ທີ່ໄປຢ້ຽມຢາມໂດຍ DFS.



1. Backtracking

1.1. Depth First Search


| pseudocode ຂອງການຄົ້ນຫາ depth-first ສາມາດສະແດງອອກດັ່ງຕໍ່ໄປນີ້.

```
def depth_first_search(node v):  
    visit v  
    for each child u of v:  
        depth_first_search(u)
```



1. Backtracking

1.2. ວິທີການ Backtracking

Backtracking ແມ່ນວິທີການອອກແບບຂັ້ນຕອນວິທີທີ່ມີປະໂຫຍດ ເຊິ່ງສ້າງຄໍາຕອບທີ່ອາດເປັນໄປໄດ້ ແບບເປັນຂັ້ນຕອນ ແລະ ຢ້ອນກັບທັນທີເມື່ອຮູ້ວ່າການແກ້ໄຂບັນຫາບໍ່ສາມາດນໍາໄປສູ່ການແກ້ໄຂທີ່ຖືກຕ້ອງ.

 Focus Backtracking ແກ້ໄຂບັນຫາກັບຂະບວນການຕໍ່ໄປນີ້

- ▶ ກໍານົດເນື້ອທີ່ຄົ້ນຫາໃນຮູບແບບຂອງໂຄງສ້າງຕົ້ນໄມ້.
- ▶ ສໍາຫຼວດເນື້ອທີ່ຄົ້ນຫາໂຄງສ້າງແບບຕົ້ນໄມ້ໂດຍການຄົ້ນຫາ depth-first.
- ▶ ຢ້ອນກັບຄືນມາເມື່ອມັນບໍ່ມີເປົ້າໝາຍຄົ້ນຫາ.

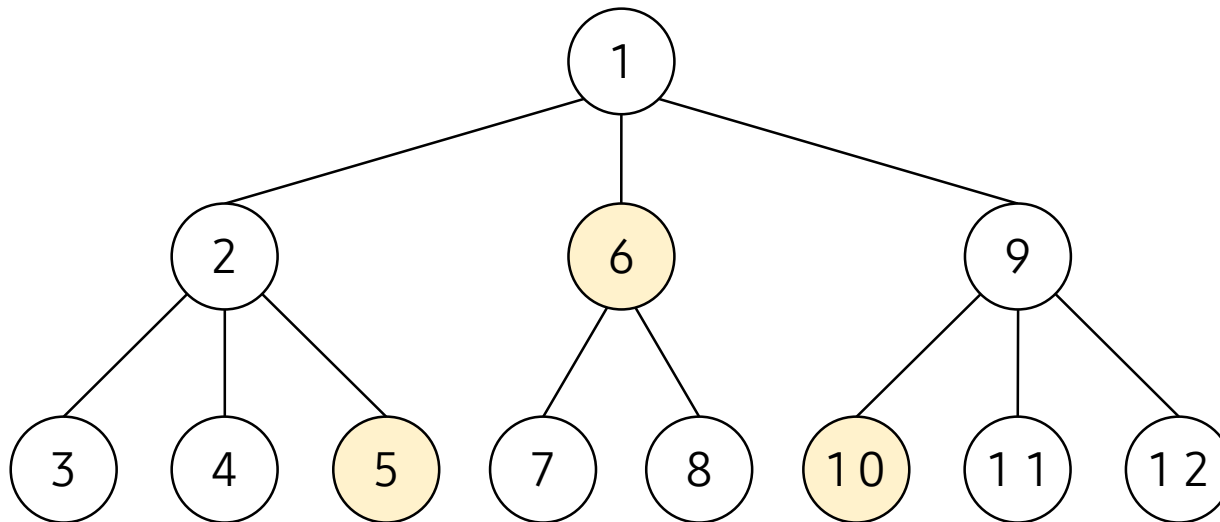
 Focus Backtracking ໃຊ້ເຕັກນິກການຕັດ(pruning) ດ້ວຍວິທີການດັ່ງຕໍ່ໄປນີ້.

- ▶ pruning ຫມາຍເຖິງການກັບຄືນໂດຍບໍ່ມີການຢ້ຽມຢາມ nodes ໃນ subtree ທີ່ບໍ່ມີເປົ້າໝາຍການຄົ້ນຫາ(not promising).
- ▶ Promising ຫມາຍເຖິງສ່ວນທີ່ການແກ້ໄຂອາດຈະມີຢູ່ໃນ subtree ຂອງ node ທີ່ກໍາລັງຢ້ຽມຢາມໃນປັດຈຸບັນ.

1. Backtracking

1.2. ວິທີການ Backtracking

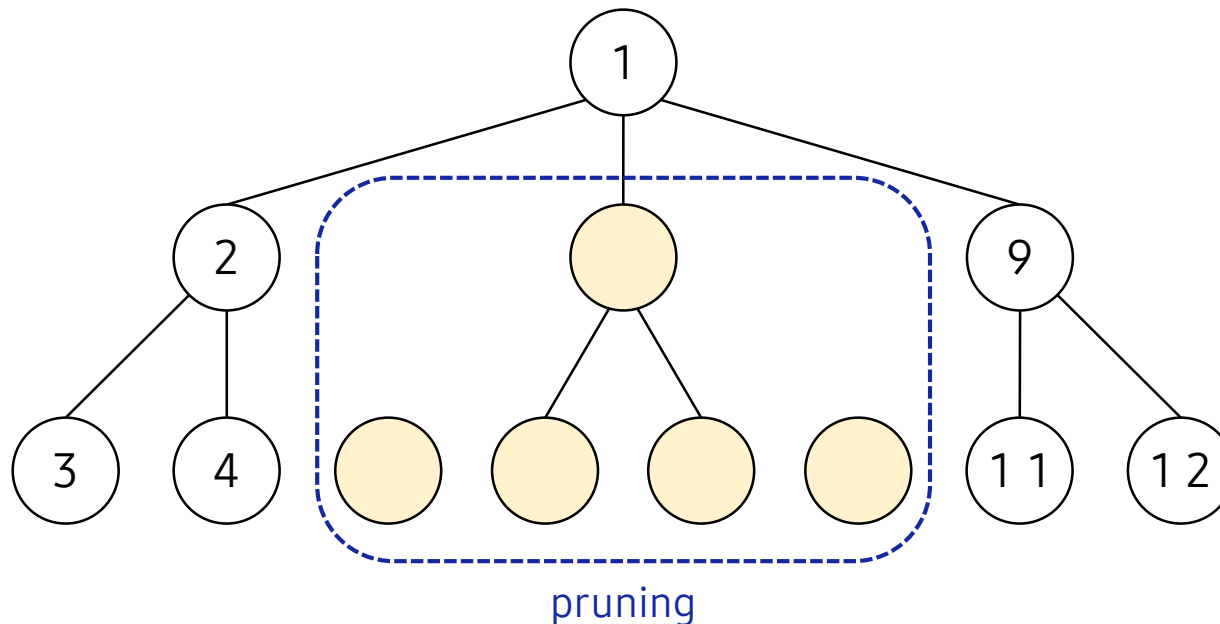
ຕົວຢ່າງ, ດັ່ງທີ່ສະແດງຢູ່ຂ້າງລຸ່ມນີ້, nodes 5, 6, ແລະ 10 ບໍ່ຈຳເປັນຕ້ອງໄປຢ້ຽມຢາມ node ເຫຼົ່ານີ້ ແລະ sub-nodes ຂອງມັນຖ້າພວກມັນບໍ່ມີເປົ້າໝາຍການຄົ້ນຫາ.



1. Backtracking

1.2. ວິທີການ Backtracking

ດັ່ງນັ້ນ, ປະສິດທິພາບຂອງ algorithm ອາດຈະໄດ້ຮັບການປັບປຸງໂດຍການຕັດ(pruning) nodes ທີ່ບໍ່ມີເປົ້າໝາຍການຄົ້ນຫາ(unpromising) ອອກ ດັ່ງຕໍ່ໄປນີ້ແລະດັ່ງນັ້ນຈຶ່ງບໍ່ຈຳເປັນຕ້ອງຢ້ຽມຢາມ.

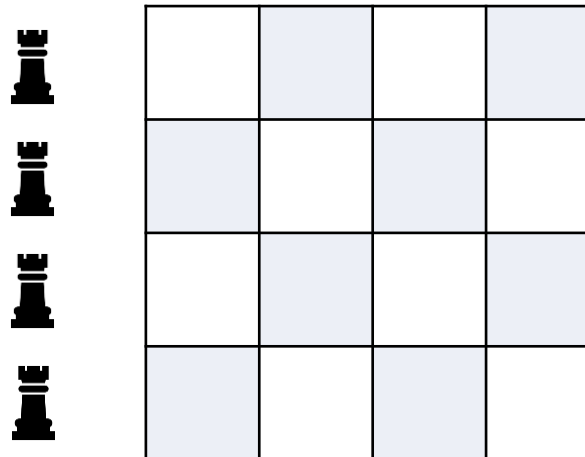


| Let's code

1. ບັນຫາ N-Queens ກັບ Backtracking

1.1. ແກ້ໄຂບັນຫາ N-Queens Problem ດ້ວຍ Backtracking

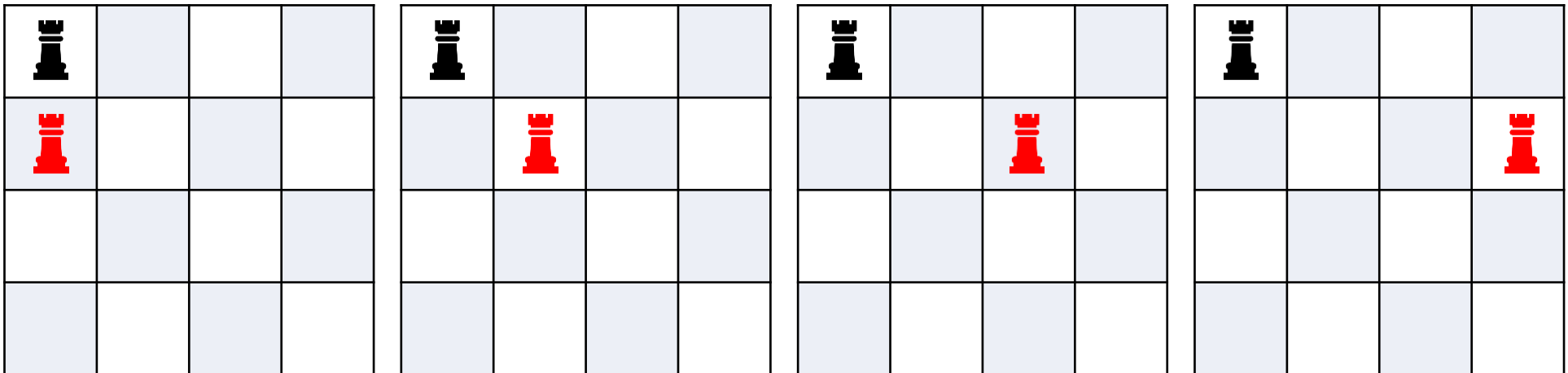
■ ເພື່ອແກ້ໄຂບັນຫາ N-Queens, ໃຫ້ພິຈາລະນາກໍລະນີຂອງ $N=4$. ບັນຫາ 4-Queens ແມ່ນບັນຫາຂອງການວາງສີ່ Queens ຢູ່ໃນກະດານ chessboard 4×4 .



1. ບັນຫາ N-Queens ກັບ Backtracking

1.1. ແກ້ໄຂບັນຫາ N-Queens Problem ດ້ວຍ Backtracking

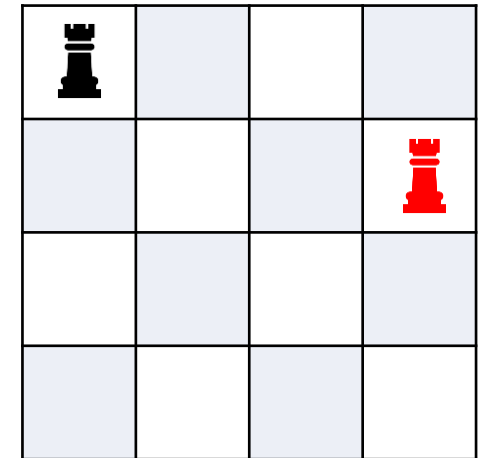
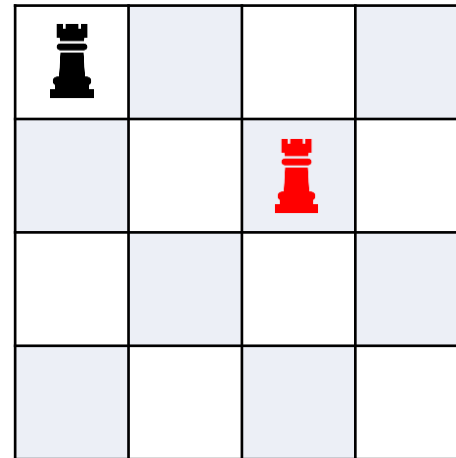
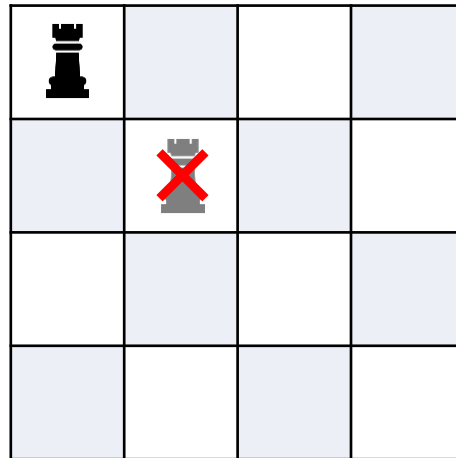
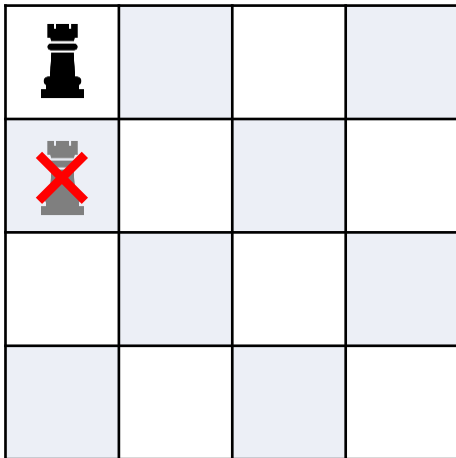
- ວິທີທີ່ງ່າຍທີ່ສຸດແມ່ນການວາງ Queen ໃນຕຳແໜ່ງທີ່ສາມາດກຳນົດໄດ້.
- ຖ້າ Queen ສືບຕໍ່ຖືກຈັດໃສ່ໃນຕຳແໜ່ງຂອງ (0, 0) ດັ່ງທີ່ສະແດງຂ້າງລຸ່ມນີ້, ສາມາດວາງ Queen ສືບຕໍ່ໃນທຸກໆຕຳແໜ່ງທີ່ເປັນໄປໄດ້ໃນແຖວທີສອງ.



1. ບັນຫາ N-Queens ກັບ Backtracking

1.1. ແກ້ໄຂບັນຫາ N-Queens Problem ດ້ວຍ Backtracking

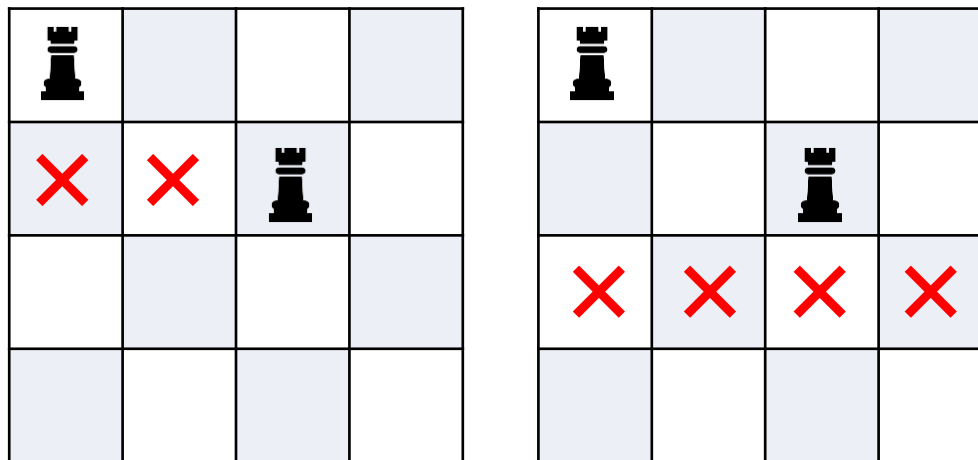
- ຢ່າງໃດກໍຕາມ, ຊ່ອງທຳອິດແລະທີສອງໃນແຖວທີສອງບໍ່ສາມາດວາງໄວ້.
- ເນື່ອງຈາກວ່າ Queen ສີດຳ, ທີ່ວາງໄວ້ແລ້ວ, ບໍ່ສາມາດຢູ່ຮ່ວມກັນກັບ Queen ອື່ນໃນແຖວດຽວກັນຫຼືເສັ້ນຂວາງໄດ້.



1. ບັນຫາ N-Queens ກັບ Backtracking

1.1. ແກ້ໄຂບັນຫາ N-Queens Problem ດ້ວຍ Backtracking

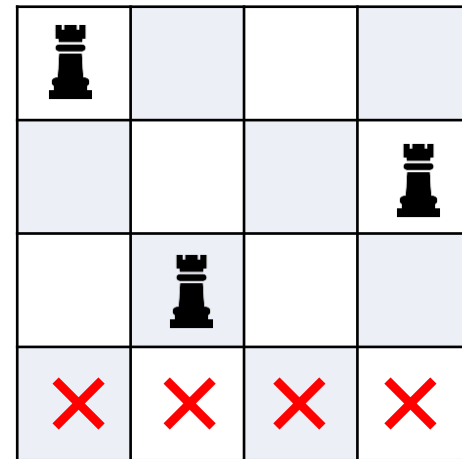
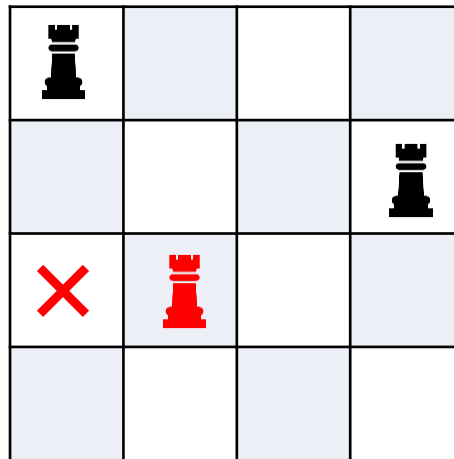
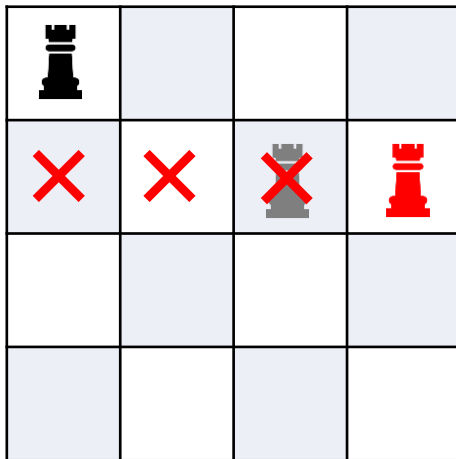
- ຖ້າຫາກວ່າ, ໃນແຖວທີສອງ, Queen ໄດ້ຖືກຈັດໃສ່ໃນຖັນທີສາມ.
- ໃນແຖວທີສາມ, Queen ບໍ່ສາມາດຖືກວາງໄວ້ທຸກບ່ອນໄດ້.
- ຍ້ອນວ່າມັນບໍ່ສາມາດຖືກຈັດໃສ່ໃນແຖວ, ຖັນ, ຫຼືເສັ້ນຂວາງດຽວກັນຍ້ອນວ່າ 2 Queen ຢູ່ໃນຕຳແໜ່ງນັ້ນແລ້ວ.



1. ບັນຫາ N-Queens ກັບ Backtracking

1.1. ແກ້ໄຂບັນຫາ N-Queens Problem ດ້ວຍ Backtracking

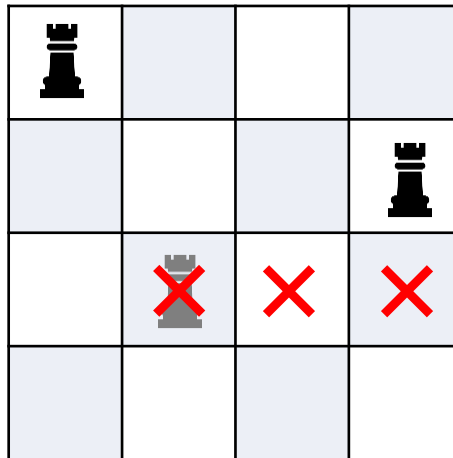
- | ໃຫ້ກັບຄືນໄປຫາສ່ວນທີ່ສອງແລະວາງ Queen ຢູ່ໃນຫ້ອງທີ່ສີ່.
- | ເວລານີ້, Queen ສາມາດຖືກຈັດໃສ່ໃນຖັນທີສອງໃນແຖວທີສາມ.
- | ຢ່າງໃດກໍຕາມ, queen ບໍ່ສາມາດຖືກຈັດໃສ່ໃນຊ່ອງສີ່ໄດ້.



1. ບັນຫາ N-Queens ກັບ Backtracking

1.1. ແກ້ໄຂບັນຫາ N-Queens Problem ດ້ວຍ Backtracking

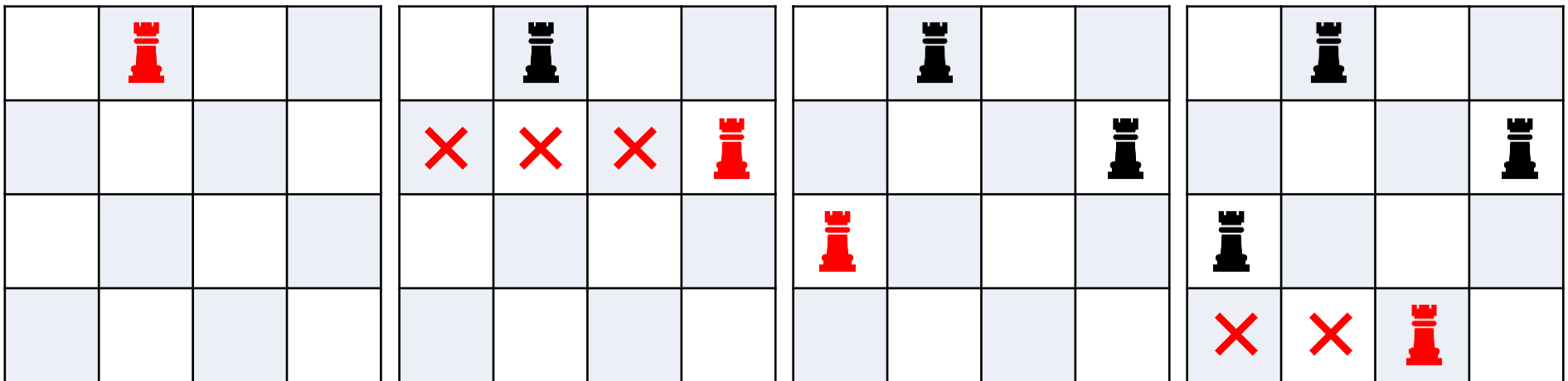
- | ໃຫ້ກັບຄືນໄປບ່ອນແຖວທີສາມແລະວາງ Queen.
- | Queen ບໍ່ສາມາດຖືກຈັດໃສ່ໃນແຖວທີສາມແລະສີ່, ດັ່ງນັ້ນພວກເຮົາຕ້ອງກັບຄືນໄປບ່ອນແຖວທີສອງອີກເທື່ອຫນຶ່ງ.
- | ເນື່ອງຈາກພວກເຮົາໄດ້ພະຍາຍາມວາງໃນແຖວທີສອງທັງໝົດແລ້ວ, ໃຫ້ກັບຄືນໄປຫາແຖວທຳອິດ ແລະ ພະຍາຍາມໃນແຖວຕໍ່ໄປ.



1. ບັນຫາ N-Queens ກັບ Backtracking

1.1. ແກ້ໄຂບັນຫາ N-Queens Problem ດ້ວຍ Backtracking

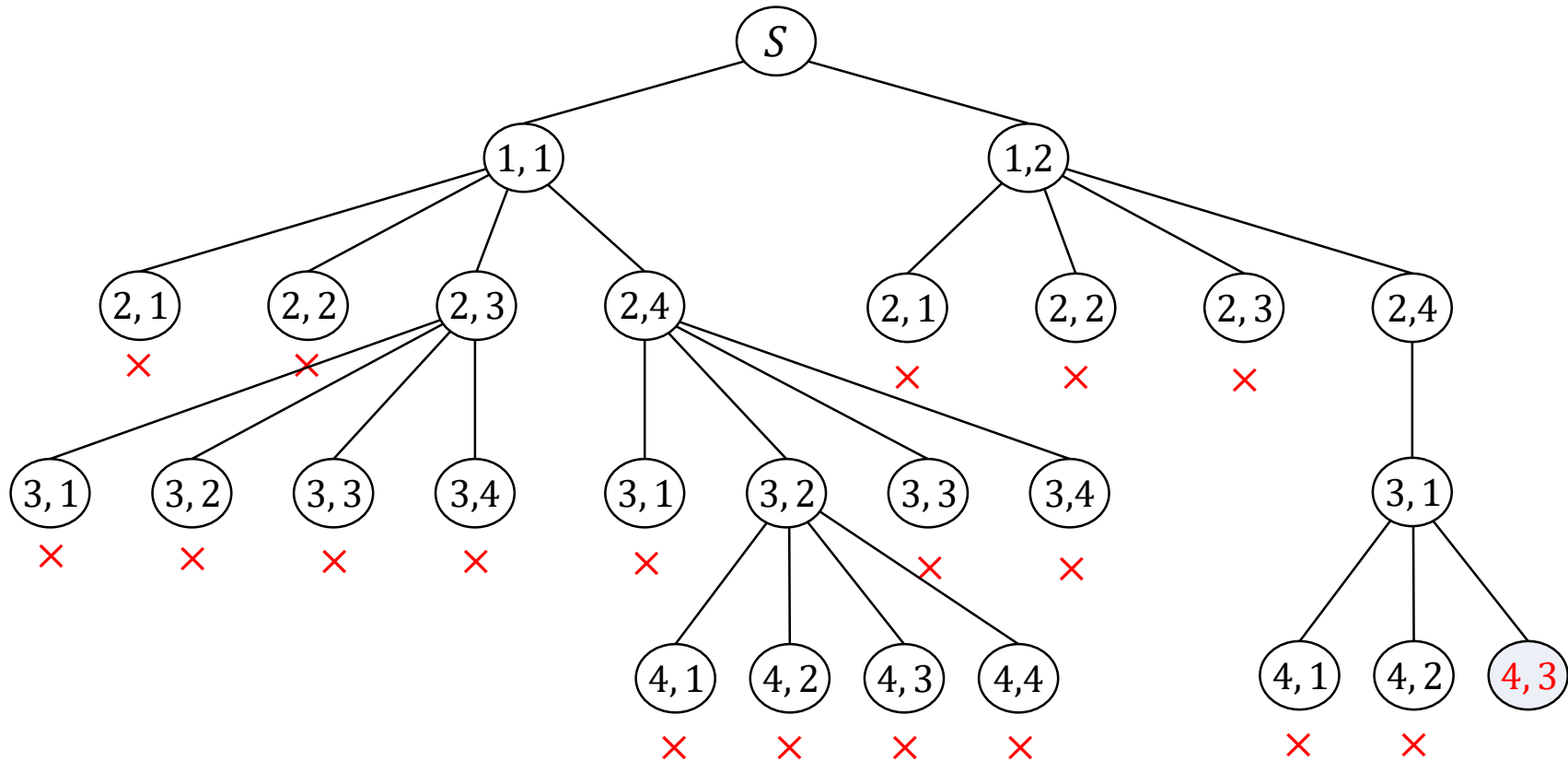
- ຖ້າ Queen ຖືກວາງຢູ່ເທິງຖັນທີສອງໃນແຖວທຳອິດ, Queen ສາມາດຖືກວາງຢູ່ເທິງຖັນທີສີ່ຂອງແຖວທີສອງ.
- ຖ້າເອົາ Queen ໃນຖັນທຳອິດຂອງແຖວທີສາມ, ສາມາດໃສ່ Queen ໃນຖັນທີສາມຂອງແຖວທີສີ່.
- ດຽວນີ້ພວກເຮົາໄດ້ແກ້ໄຂບັນຫານີ້ແລ້ວ ເພາະວ່າພວກເຮົາມີທັງໝົດສີ່ Queen.



1. ບັນຫາ N-Queens ກັບ Backtracking

1.2. ເນື້ອທີ່ຄົ້ນຫາ (Search Space) ກັບການຕັດອອກ (Pruning)

Search space tree ຂອງບັນຫາ N-Queens ສາມາດຖືກແຕ້ມດັ່ງຕໍ່ໄປນີ້.



1. ບັນຫາ N-Queens ກັບ Backtracking

1.3. Pseudo-code ສໍາຫຼັບການແກ້ໄຂບັນຫາ N-Queens

| pseudocode ສໍາລັບການແກ້ໄຂບັນຫານີ້ສາມາດສະແດງອອກດັ່ງຕໍ່ໄປນີ້.

```
def n_queens(node v):  
    if (promising(v)):  
        if (there is a solution at v):  
            write the solution;  
    else  
        for (each child u of v):  
            n_queens(u);
```

1. ບັນຫາ N-Queens ກັບ Backtracking

1.4. ການສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ຟັງຊັນທີ່ຕິດວນກວດເບິ່ງວ່າສອງ Queen ມີຢູ່ໃນແຖວ, ຖັນ, ຫຼື ເສັ້ນຂວາງດຽວກັນບໍ່.

- ສົມມຸດວ່າ $col[i]$ ແມ່ນຕົວເລກຂອງຖັນໃນແຖວທີ i .
- ນັ້ນແມ່ນ, $col = [1, 0, 2, 3]$ ສະແດງໃຫ້ເຫັນສະຖານະຕໍ່ໄປນີ້.

$col = [1, 0, 2, 3]$

	0	1	2	3
0		♔		
1	♔			
2			♔	
3				♔

$col[0]=1: (0, 1)$

$col[1]=0: (1, 0)$

$col[2]=2: (2, 2)$

$col[3]=3: (3, 3)$

1. ບັນຫາ N-Queens ກັບ Backtracking

1.4. ການສ້າງ ແລະ ຂຽນໂປຣແກຣມ

- ເນື່ອງຈາກພວກເຮົາຈະວາງ queens ພຽງແຕ່ຢູ່ໃນແຖວທີ່ແຕກຕ່າງກັນ, ພວກເຮົາສາມາດຫຼີກເວັ້ນການທັບຊ້ອນຂອງແຖວ.
- ເພື່ອກວດເບິ່ງວ່າສອງ Queen ມີຢູ່ໃນຖັນດຽວກັນບໍ່, ແມ່ນໃຫ້ກວດສອບຕາມສະຖານະການຕໍ່ໄປນີ້.

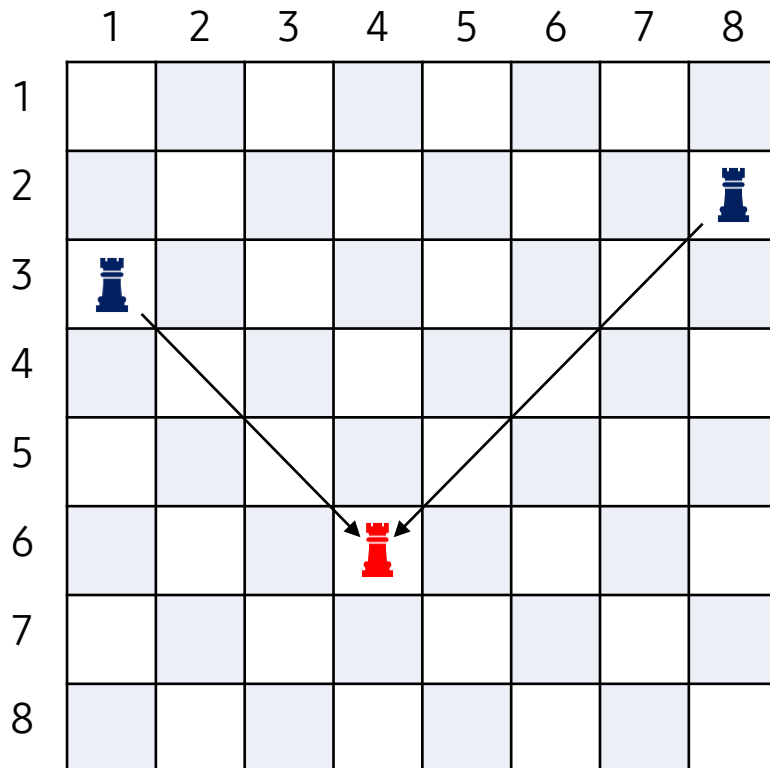
	1	2	3	4	5	6	7	8
1								
2								
3				♔				
4								
5								
6				♚				
7								
8								

$col[i] == col[j]$

1. ບັນຫາ N-Queens ກັບ Backtracking

1.4. ການສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ເພື່ອກຳນົດວ່າສອງ queens ສາມາດມີຢູ່ໃນເສັ້ນຂວາງໄດ້ຫຼືບໍ່, ຄຳສຳບຸນຂອງຄວາມແຕກຕ່າງລະຫວ່າງແຖວແລະຖັນຄວນໄດ້ຮັບການກວດສອບດັ່ງຕໍ່ໄປນີ້.



$$|col[i] - col[j]| == i - j$$

1. ບັນຫາ N-Queens ກັບ Backtracking

1.4. ການສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ການໃຊ້ງານຟັງ promise ຄວນກວດສອບວ່າ ຖັນທີ i ມີແນວໂນ້ມຢູ່ໃນລາຍການຖັນທີ່ກຳໜົດໄວ້ຫຼືບໍ່

```
1 def promising(i, col):
2     for k in range(i):
3         if col[i] == col[k] or abs(col[i] - col[k]) == (i - k):
4             return False
5     return True
```

 Line 1-2

- ໃຫ້ລາຍການຖັນ ແລະ ດັດສະນີ i ເປັນພາຣາມິເຕີທີ່ປ້ອນເຂົ້າ, ກວດສອບວ່າຖັນທີ i ມີແນວໂນ້ມດີຫຼືບໍ່.
- ແຖວທີ່ queen ໄດ້ຖືກວາງໄວ້ແລ້ວ ແມ່ນແຖວທີ 0 ເຖິງ ແຖວທີ $i - 1$, ດັ່ງນັ້ນ ມັນຈະເຄື່ອນຍ້າຍໂດຍໃຊ້ for loop ຈາກ 0 ເຖິງ $i - 1$.

1. ບັນຫາ N-Queens ກັບ Backtracking

1.4. ການສ້າງ ແລະ ຂຽນໂປຣແກຣມ

```
1 def promising(i, col):
2     for k in range(i):
3         if col[i] == col[k] or abs(col[i] - col[k]) == (i - k):
4             return False
5     return True
```

 Line 3-5

- ສໍາຫຼັບດັດສະນີ k ແຕ່ລະອັນ, ກວດສອບວ່າມີ queens ທີ່ແຕກຕ່າງກັນຢູ່ໃນຖັນດຽວກັນຫຼືບໍ່ ຫຼືວ່າ ມີ queens ທີ່ແຕກຕ່າງກັນຢູ່ໃນເສັ້ນເນັ້ງແຈດຽວກັນຫຼືບໍ່.
- ຖ້າມີ queen ຢູ່ໃນຖັນດຽວກັນ ຫຼືວ່າ ຢູ່ໃນເສັ້ນເນັ້ງແຈດຽວກັນ ໃຫ້ສິ່ງຄ່າ False ກັບຄືນ. ຖ້າບໍ່ດັ່ງນັ້ນ ໃຫ້ສິ່ງຄ່າ True ກັບຄືນ.

1. ບັນຫາ N-Queens ກັບ Backtracking

1.4. ການສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ຟັງຊັນ `n_queens()` ໃຊ້ວິທີການ backtracking.

```
1 def n_queens(i, col):
2     if promising(i, col):
3         if i == len(col) - 1:
4             print(col)
5         else:
6             for j in range(len(col)):
7                 col[i + 1] = j
8                 n_queens(i + 1, col)
```

 Line 2-4

- ໃຫ້ `i` ແລະ `col` ເປັນພາຣາມິເຕີປ້ອນເຂົ້າ, ຖ້າວ່າຖັນທີ `i` ບໍ່ມີແນວໂນ້ມວ່າຈະໄດ້ຜົນດີ ລະບົບຈະສິ້ງຄືນທັນທີ.
- ຖ້າວ່າຖັນທີ `i` ມີແນວໂນ້ມໄດ້ຜົນດີ ແລະ `i` ເປັນແຖວສຸດທ້າຍ, ສະແດງວ່າພົບວິທີແກ້ໄຂບັນຫາແລ້ວ, ດັ່ງນັ້ນ ລາຍການ `col` ທີ່ກ່ຽວຂ້ອງຈຶ່ງເປັນຂໍ້ມູນປ້ອນເຂົ້າ.

1. ບັນຫາ N-Queens ກັບ Backtracking

1.4. ການສ້າງ ແລະ ຂຽນໂປຣແກຣມ

```
1 def n_queens(i, col):
2     if promising(i, col):
3         if i == len(col) - 1:
4             print(col)
5         else:
6             for j in range(len(col)):
7                 col[i + 1] = j
8                 n_queens(i + 1, col)
```

 Line 5-8

- ຖ້າວ່າຖັນທີ i ມີແນວໂນ້ມໄດ້ຜົນດີ ແລະ i ບໍ່ແມ່ນແຖວສຸດທ້າຍ, ໃຫ້ສືບຕໍ່ຄົ້ນຫາແບບ depth-first.
- Line 7: ກ່ອນສືບຕໍ່ຄົ້ນຫາແບບ depth-first, ໃຫ້ວາງ queen ຢູ່ໃນຖັນທີ $i+1$ ໃນອັນດັບທີ j ໃນລາຍການຖັນ,
- Line 8: ແລ້ວສືບຕໍ່ຄົ້ນຫາໃນແຖວທີ $i + 1$.

1. ບັນຫາ N-Queens ກັບ Backtracking

1.5. ຮັບໂປຣແກຣມ N-Queens Solver

■ ຫຼັງຈາກໄດ້ຮັບຄ່າຂອງ N ທີ່ເປັນຂໍ້ມູນປ້ອນເຂົ້າຈາກຜູ້ໃຊ້, ລາຍການ col ເລີ່ມຕົ້ນໃນແຖວທີ i ຫາ -1 ແລະ ລາຍການອົງປະກອບທີ່ມີຄ່າເປັນ -1 ໄດ້ຖືກກຳໜົດໃຫ້ເປັນຄ່າພາລາມິຕີ ແລ້ວຟັງຊັນ n_queens() ຖືກເອີ້ນໃຊ້.

```
1 N = int(input("Input the number of queens: "))
2 n_queens(-1, [-1] * N)
```

```
Input the number of queens: 4
[1, 3, 0, 2]
[2, 0, 3, 1]
```

| Pop quiz

Q1. ຖ້າທ່ານແທນທີ່ແຕ່ລະຕົວອັກສອນດ້ວຍຕົວເລກໃນຕົວອັກສອນຕໍ່ໄປນີ້, ຕົວເລກທີ່ສອດຄ້ອງກັນກັບທັງຫ້າຄໍາຈະເປັນຮູບຈະຕຸ້ລດ. ແລະສໍາລັບແຕ່ລະຄໍາ, ຜົນບວກຂອງແຕ່ລະຕົວເລກຍັງກາຍເປັນຕົວເລກຮູບຈະຕຸ້ລດ. ຊອກຫາຕົວເລກທີ່ແຕ່ລະຕົວອັກສອນເປັນຕົວແທນ.

A MERRY XMAS TO ALL

A M E R R Y X M A S T O A L L

8 1

Q1. ຖ້າທ່ານແທນທີ່ແຕ່ລະຕົວອັກສອນດ້ວຍຕົວເລກໃນຕົວອັກສອນຕໍ່ໄປນີ້, ຕົວເລກທີ່ສອດຄ້ອງກັນກັບທັງຫ້າຄໍາຈະເປັນຮູບຈະຕຸ້ລດ. ແລະສໍາລັບແຕ່ລະຄໍາ, ຜົນບວກຂອງແຕ່ລະຕົວເລກຍັງກາຍເປັນຕົວເລກຮູບຈະຕຸ້ລດ. ຊອກຫາຕົວເລກທີ່ແຕ່ລະຕົວອັກສອນເປັນຕົວແທນ.

A MERRY XMAS TO ALL

A M E R R Y X M A S T O A L L

Hint

Note that T and O can be 8 and 1, respectively.
 Since 81 is a perfect square number, $81 = 9^2$.
 Also, $8 + 1$ is a perfect square number, since $9 = 3^2$.

8 1

| Pair programming



Pair Programming Practice

| ແນວທາງ, ກົນໄກ ແລະ ແຜນສຸກເສີນ

ການກະກຽມການສ້າງໂປຣແກຣມຮ່ວມກັນເປັນຄູ່ກ່ຽວຂ້ອງກັບການໃຫ້ຄໍາແນະນໍາແລະກົນໄກເພື່ອຊ່ວຍໃຫ້ນັກຮຽນຈັບຄູ່ຢ່າງຖືກຕ້ອງແລະ ໃຫ້ເຂົາເຈົ້າເຮັດວຽກເປັນຄູ່. ຕົວຢ່າງ, ນັກຮຽນຄວນປຸງຮຽນ "ເຮັດ." ການກະກຽມທີ່ມີປະສິດຕິຜົນຕ້ອງໃຫ້ມີແຜນການສຸກເສີນໃນກໍລະນີທີ່ຄູ່ຮ່ວມງານຫນຶ່ງບໍ່ຢູ່ຫຼືຕັດສິນໃຈທີ່ຈະບໍ່ເຂົ້າຮ່ວມດ້ວຍເຫດຜົນໃດຫນຶ່ງ ຫຼືດ້ວຍເຫດຜົນອື່ນ. ໃນກໍລະນີເຫຼົ່ານີ້, ມັນເປັນສິ່ງສໍາຄັນທີ່ຈະເຮັດໃຫ້ມັນຊັດເຈນວ່ານັກຮຽນທີ່ມີປະຕິບັດໜ້າທີ່ຢ່າງຫ້າວຫັນຈະບໍ່ຖືກລົງໂທດຍ້ອນວ່າການຈັບຄູ່ບໍ່ໄດ້ຜິດ.

| ການຈັບຄູ່ທີ່ຄ້າຍຄືກັນ, ບໍ່ຈໍາເປັນຕ້ອງເທົ່າທຽມກັນ, ຄວາມສາມາດເປັນຄູ່ຮ່ວມງານ

ການຂຽນໂປຣແກຣມຄູ່ຈະມີປະສິດທິພາບເມື່ອນັກຮຽນຕັ້ງໃຈຮ່ວມກັນເຮັດວຽກ, ຊຶ່ງວ່າບໍ່ຈໍາເປັນຕ້ອງມີຄວາມຮູ້ເທົ່າທຽມກັນ, ແຕ່ຕ້ອງມີຄວາມສາມາດເຮັດວຽກເປັນຄູ່ຮ່ວມງານ. ການຈັບຄູ່ນັກຮຽນທີ່ບໍ່ສາມາດເຂົ້າກັນໄດ້ມັກຈະເຮັດໃຫ້ການມີສ່ວນຮ່ວມທີ່ບໍ່ສົມດຸນກັນ. ຄຸສອນຕ້ອງເນັ້ນຫນັກວ່າການຂຽນໂປຣແກຣມຄູ່ບໍ່ແມ່ນຍຸດທະສາດ “divide-and-conquer”, ແຕ່ຈະເປັນຄວາມພະຍາຍາມຮ່ວມມືເຮັດວຽກທີ່ແທ້ຈິງໃນທຸກໆດ້ານສໍາລັບໂຄງການທັງຫມົດ. ຄູຄວນຫຼີກເວັ້ນການຈັບຄູ່ນັກຮຽນທີ່ອ່ອນຫຼາຍກັບນັກຮຽນທີ່ເກັ່ງຫຼາຍ.

| ກະຕຸ້ນນັກຮຽນໂດຍການໃຫ້ສິ່ງຈູງໃຈພິເສດ

ການສະເໜີແຮງຈູງໃຈພິເສດສາມາດຊ່ວຍກະຕຸ້ນນັກຮຽນໃຫ້ຈັບຄູ່, ໂດຍສະເພາະກັບນັກຮຽນທີ່ມີຄວາມສາມາດຫຼາຍ. ຈະເຫັນວ່າມັນເປັນປະໂຫຍດທີ່ຈະໃຫ້ນັກຮຽນຈັບຄູ່ເຮັດວຽກຮ່ວມກັນພຽງແຕ່ຫນຶ່ງຫຼືສອງວຽກເທົ່ານັ້ນ.



Pair Programming Practice

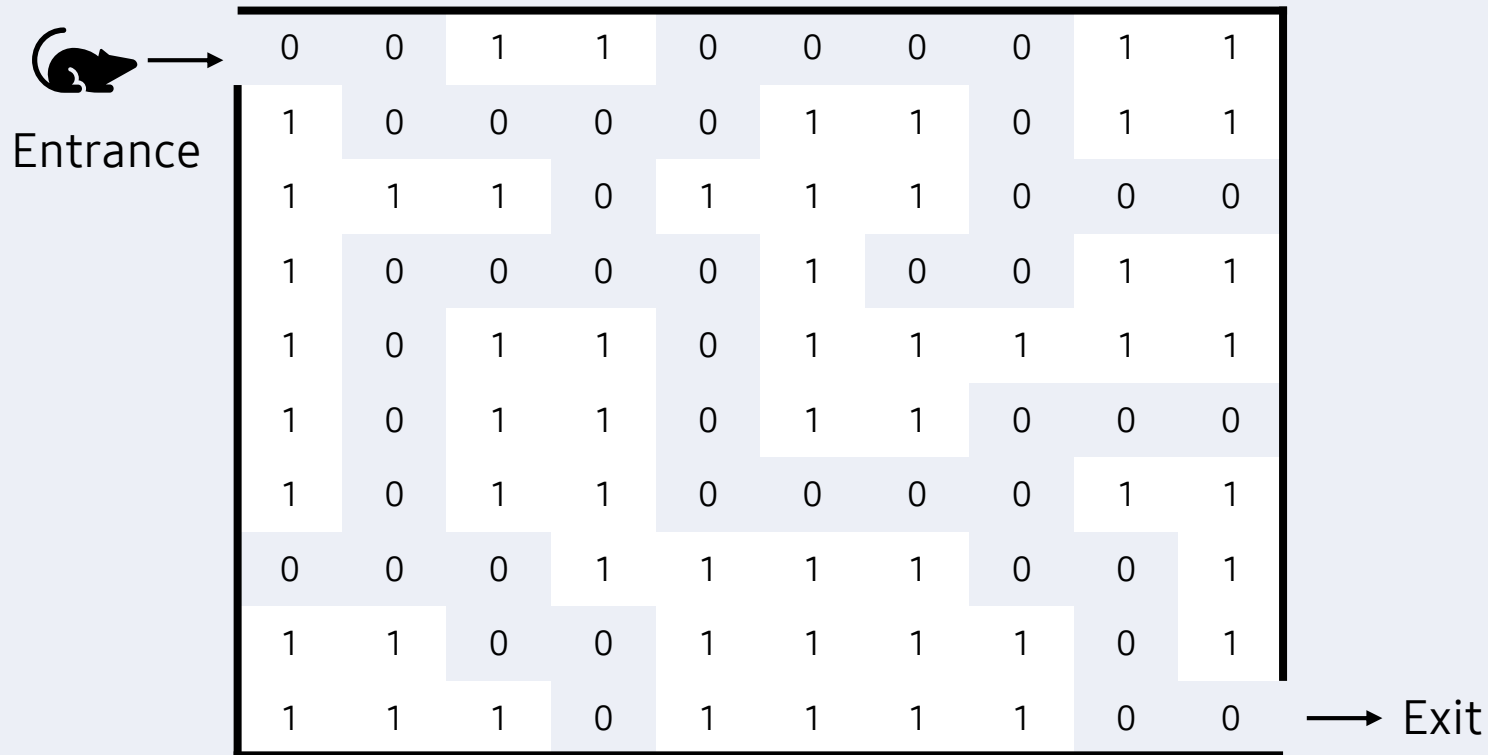
| ປ້ອງກັນການໂກງໃນການເຮັດວຽກຮ່ວມກັນ

ສິ່ງທ້າທາຍສໍາລັບຄູແມ່ນເພື່ອຊອກຫາວິທີທີ່ຈະປະເມີນຜົນໄດ້ຮັບຂອງບຸກຄົນ, ໃນຂະນະທີ່ນໍາໃຊ້ຜົນປະໂຫຍດຂອງການຮ່ວມມື. ຈະຮູ້ໄດ້ແນວໃດວ່ານັກຮຽນຕັ້ງໃຈເຮັດວຽກ ຫຼື ກົງແຮງງານຜູ້ຮ່ວມງານ? ຜູ້ຊ່ຽວຊານແນະນໍາໃຫ້ທົບທວນຄືນການອອກແບບບັນຫາສູດ ແລະ ການປະເມີນ ພ້ອມທັງປຶກສາຫາລືຢ່າງຈະແຈ້ງ ແລະ ຊັດເຈນກ່ຽວກັບພຶດຕິກຳຂອງນັກຮຽນທີ່ຈະຖືກຕິດຕາມວ່າຂີ້ຕົວະ. ຜູ້ຊ່ຽວຊານເນັ້ນໜັກໃຫ້ຄູເຮັດການມອບໝາຍໃຫ້ມີຄວາມໝາຍຕໍ່ນັກຮຽນ ແລະ ອະທິບາຍຄຸນຄ່າຂອງສິ່ງທີ່ນັກຮຽນຈະຮຽນຮູ້ໂດຍການເຮັດສໍາເລັດ.

| ສະພາບແວດລ້ອມການຮຽນຮູ້ໃນການຮ່ວມມື

ສະພາບແວດລ້ອມການຮຽນຮູ້ໃນຮ່ວມກັນເກີດຂຶ້ນໄດ້ທຸກເວລາທີ່ຜູ້ສອນຮຽກຮ້ອງໃຫ້ນັກຮຽນເຮັດວຽກຮ່ວມກັນໃນກິດຈະກຳການຮຽນຮູ້. ສະພາບແວດລ້ອມການຮຽນຮູ້ຮ່ວມກັນສາມາດມີສ່ວນຮ່ວມທັງກິດຈະກຳທີ່ເປັນທາງການ ແລະ ບໍ່ເປັນທາງການ ແລະ ອາດຈະບໍ່ລວມເຖິງການປະເມີນໂດຍກົງ. ຕົວຢ່າງ, ນັກສຶກສາຄູ່ເຮັດວຽກມອບໝາຍຮ່ວມກັນໃນການຂຽນໂປຣແກຣມ; ນັກສຶກສາກຸ່ມນ້ອຍໆສິນທະນາຄໍາຕອບທີ່ເປັນໄປໄດ້ຕໍ່ກັບຄໍາຖາມຂອງອາຈານໃນລະຫວ່າງການບັນຍາຍ; ແລະ ນັກຮຽນເຮັດວຽກຮ່ວມກັນນອກຫ້ອງຮຽນເພື່ອຮຽນຮູ້ແນວຄວາມຄິດໃໝ່. ການຮຽນຮູ້ການຮ່ວມມືແມ່ນແຕກຕ່າງຈາກໂຄງການທີ່ນັກຮຽນ “divide and conquer.” ເມື່ອນັກຮຽນແບ່ງວຽກກັນ, ແຕ່ລະຄົນຮັບຜິດຊອບພຽງແຕ່ສ່ວນໜຶ່ງຂອງການແກ້ໄຂບັນຫາ ແລະ ຈະບໍ່ຄ່ອຍມີບັນຫາຫຍັງໃນການເຮັດວຽກຮ່ວມກັບຄົນອື່ນໃນທີມ. ໃນສະພາບແວດລ້ອມການເຮັດວຽກຮ່ວມກັນ, ນັກຮຽນມີສ່ວນຮ່ວມໃນການສິນທະນາປຶກສາຫາລືເຊິ່ງກັນແລະກັນ.

Q1. $N \times N$ binary matrix ແມ່ນໄດ້ໃຫ້ເປັນປິດສະໜາດັ່ງຕໍ່ໄປນີ້ ແລະ ໝູ່ຈະຕ້ອງເຂົ້າໄປຈາກຈຸດ (0, 0) ໄປຫາຈຸດ (N-1, N-1) ໃຫ້ສໍາເລັດ



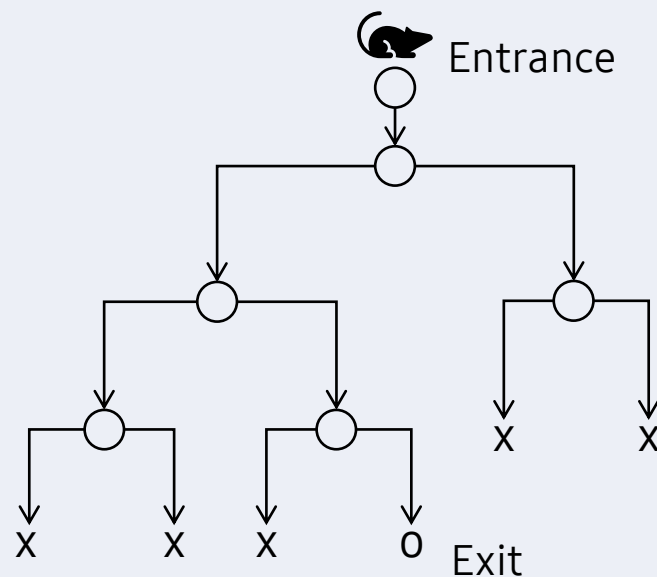
Q1. N*N binary matrix ແມ່ນໄດ້ໃຫ້ເປັນປິດສະໜາດັ່ງຕໍ່ໄປນີ້ ແລະທາງຈະຕ້ອງຈາກຈຸດ (0, 0) ໄປຫາຈຸດ (N-1, N-1) ໃຫ້ສໍາເລັດ

Hint

ເພື່ອແກ້ໄຂບົດສະໜານີ້, ໜັງສືເດີນທາງໄປຫາເສັ້ນທາງທີ່ມີເລກ 0.

| ສືບຕໍ່ການຄົ້ນຫາໂດຍໃຊ້ DFS, ກັບຄືນເມື່ອໄປເຖິງຈຸດສິ້ນສຸດ ແລະ ເດີນທາງຈົນກວ່າຈະມາຮອດຈຸດຫມາຍປາຍທາງ.

0	0	1	1	0	0	0	0	1	1
1	0	0	0	0	1	1	0	1	1
1	1	1	0	1	1	1	0	0	0
1	0	0	0	0	1	0	0	1	1
1	0	1	1	0	1	1	1	1	1
1	0	1	1	0	1	1	0	0	0
1	0	1	1	0	0	0	0	1	1
0	0	0	1	1	1	1	0	0	1
1	1	0	0	1	1	1	1	0	1
1	1	1	0	1	1	1	1	0	0



Q1. $N \times N$ binary matrix ແມ່ນໄດ້ໃຫ້ເປັນປິດສະໜາດັ່ງຕໍ່ໄປນີ້ ແລະ ຫນູຈະຕ້ອງຈາກຈຸດ (0, 0) ໄປຫາຈຸດ (N-1, N-1) ໃຫ້ສໍາເລັດ

Hint

| ໃຫ້ $N \times N$ ປິດສະໜາ, ຂຽນໂປຣແກຣມທີ່ສະແດງເສັ້ນທາງທີ່ ດັ່ງທີ່ສະແດງຂ້າງລຸ່ມນີ້.

```
1 maze = [ [0, 0, 0, 0],
2           [1, 0, 1, 0],
3           [1, 0, 1, 1],
4           [0, 0, 0, 0] ]
5 solve(maze)
```

```
0 0 1 1
1 0 1 1
1 0 1 1
1 0 0 0
```



| End of Document



SAMSUNG

Together for Tomorrow!
Enabling People

Education for Future Generations

©2021 SAMSUNG. All rights reserved.

Samsung Electronics Corporate Citizenship Office holds the copyright of book.

This book is a literary property protected by copyright law so reprint and reproduction without permission are prohibited.

To use this book other than the curriculum of Samsung innovation Campus or to use the entire or part of this book, you must receive written consent from copyright holder.

