Unit 25.

Binary Search

Learning objectives

- ✓ ເຂົ້າໃຈບັນຫາການຄົ້ນຫາ ແລະ ສາມາດແກ້ໄຂບັນຫາໂດຍຜ່ານການໃຊ້ binary search.
- ✓ ສາມາດລົງມືປະຕິບັດຕາມ binary search algorithm ດ້ວຍພາສາ Python.
- ✓ ສາມາດເຂົ້າໃຈ ແລະ ອະທິບາຍເວລາທີ່ໃຊ້ໃນການປະມວນຜົນສູງສຸດຂອງ binary search algorithm.

Learning overview

- ✓ ເຂົ້າໃຈບັນຫາການຄົ້ນຫາທີ່ຊອກຫາອົງປະກອບທີ່ກຳນົດຈາກຂໍ້ມູນທີ່ໄດ້ຈັດລຽງຕາມລຳດັບໄວ້ແລ້ວ.
- ✓ ຮຽນຮູ້ວິທີການແກ້ໄຂບັນຫາການຄົ້ນຫາຂໍ້ມູນທີ່ໄດ້ລຽງຕາມລຳດັບໄວ້ແລ້ວໂດຍໃຊ້ binary search.
- ✓ ຮຽນຮູ້ວິທີການຂຽນ binary search algorithms ໂດຍໃຊ້ພາສາ Python ແລະ ວິເຄາະ ເວລາທີ່ໃຊ້ໃນການປະມວນຜົນຕາມຂັ້ນຕອນວິທີດັ່ງກ່າວໃນກໍລະນີທີ່ດີທີ່ສຸດ, ບໍ່ດີທີ່ສຸດ ແລະ ສະເລ່ຍ.

Concepts you will need to know from previous units

- ✓ ວິທີການປຽບທຽບຂະໜາດຂອງອົງປະກອບພາຍໃນ list
- ✓ ວິທີການສ້າງຄວາມສຳພັນທີ່ເກີດຂຶ້ນຄືນໃຫມ່ ແລະ ແກ້ໄຂບັນຫາໂດຍໃຊ້ຟັງຊັນ recursive
- ✓ ວິທີການສະແດງເວລາທີ່ໃຊ້ໃນການປະມວນຕາມຂັ້ນຕອນວິທີໃດໜຶ່ງໂດຍໃຊ້ຕຳລາ Big-O

Keywords

- Searching Problem
- Binary Search
- Logarithmic Time

Mission

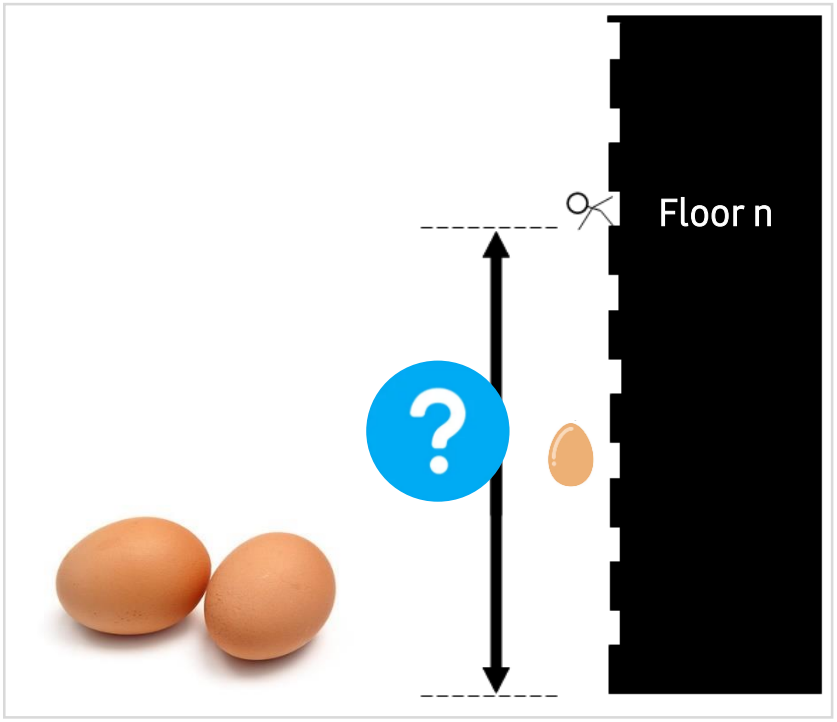
1. Real world problem

1.1. ການທົດລອງປ່ອຍໄຂ່ຈາກຕຶກ

- ມາເບິ່ງຄືນການປ່ອຍໄຂ່ລົງຈາກອາຄານອີກເທື່ອຫນຶ່ງ.
- ໃນຫົວຂໍ້ທີ່ຜ່ານມາ, ມີໄຂ່ພຽງໜ່ວຍດຽວ, ດັ່ງນັ້ນພວກເຮົາໄດ້ພະຍາຍາມຄິດໄລ່ຊັ້ນສູງທີ່ສຸດທີ່ໄຂ່ຍັງບໍ່ແຕກໃນຄັ້ງທໍາອິດ.
- ເວລານີ້, ສົມມຸດວ່າມີຈໍານວນໄຂ່ພຽງພໍ, ຊຶ່ງຫມາຍຄວາມວ່າທ່ານສາມາດສືບຕໍ່ເຮັດການທົດລອງໂດຍບໍ່ຄໍານຶງເຖິງໄຂ່ທີ່ແຕກ.
- ຖ້າມີໄຂ່ພຽງພໍ, ບັນຫາການຕົກຂອງໄຂ່ສາມາດແກ້ໄຂໄດ້ຢ່າງມີປະສິດທິພາບຫຼາຍເມື່ອທຽບໃສ່ກັບການຄົ້ນຫາແບບຕາມລໍາດັບ.
- ພະຍາຍາມແກ້ໄຂບັນຫາການຕົກຂອງໄຂ່ດ້ວຍການຄົ້ນຫາແບບ binary search.

2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່ທີ່ມີໄຂ່ບໍ່ຈຳກັດ

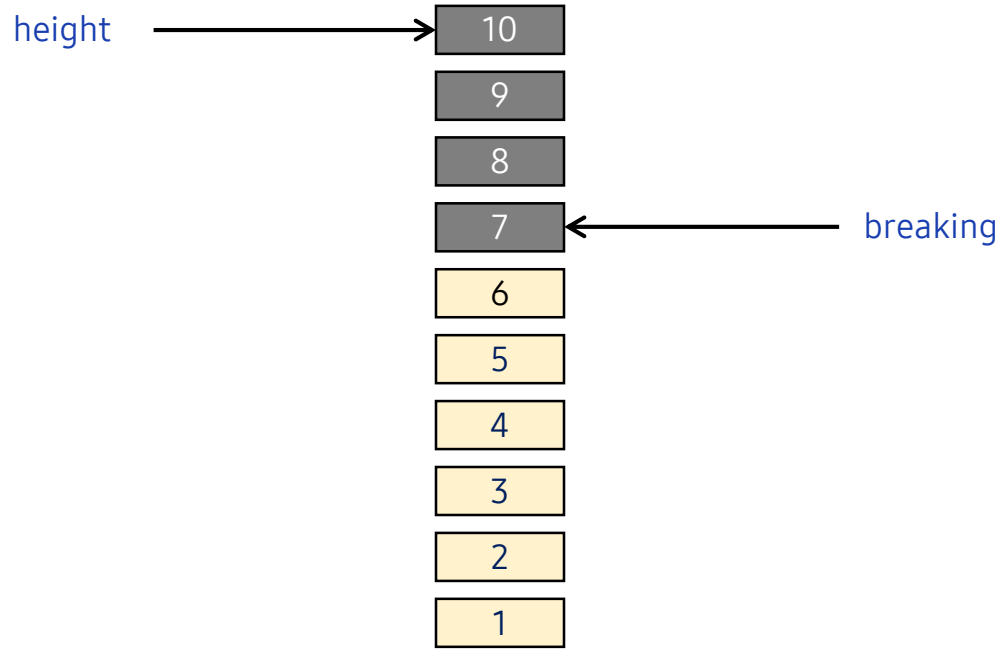


- ຖ້າກຳນົດໃຫ້ຊັ້ນສູງສຸດ ແລະ ຊັ້ນຕ່ຳສຸດຂອງອາຄານທີ່ໄຂ່ແຕກ, ຊອກຫາຊັ້ນສູງສຸດທີ່ໄຂ່ບໍ່ແຕກ.
- ເນື່ອງຈາກມີໄຂ່ທີ່ບໍ່ຈຳກັດ, ສາມາດສືບຕໍ່ທົດລອງເຖິງແມ່ນວ່າ ໄຂ່ແຕກ.
- ຖ້າສາມາດສືບຕໍ່ທົດລອງເຖິງແມ່ນວ່າໄຂ່ແຕກ, ຖ້າໄຂ່ແຕກຢູ່ ຊັ້ນກາງຂອງອາຄານ, ສະແດງວ່າຊັ້ນທີ່ໄຂ່ບໍ່ແຕກຈະຢູ່ຊັ້ນລຸ່ມ ລົງມາ. ທຳນອງດຽວກັນ, ຖ້າໄຂ່ບໍ່ແຕກຢູ່ຊັ້ນກາງຂອງອາຄານ, ສະແດງວ່າຊັ້ນທີ່ໄຂ່ແຕກຈະຢູ່ຊັ້ນເທິງ.
- ນຳໃຊ້ວິທີການດຽວກັນກັບເຄິ່ງຫນຶ່ງສຳລັບການຄົ້ນຫາຊັ້ນທີ່ ຕ້ອງການ.
- ປະເພດຂອງວິທີການນີ້ເອີ້ນວ່າ binary search.

2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່ທີ່ມີໄຂ່ບໍ່ຈຳກັດ

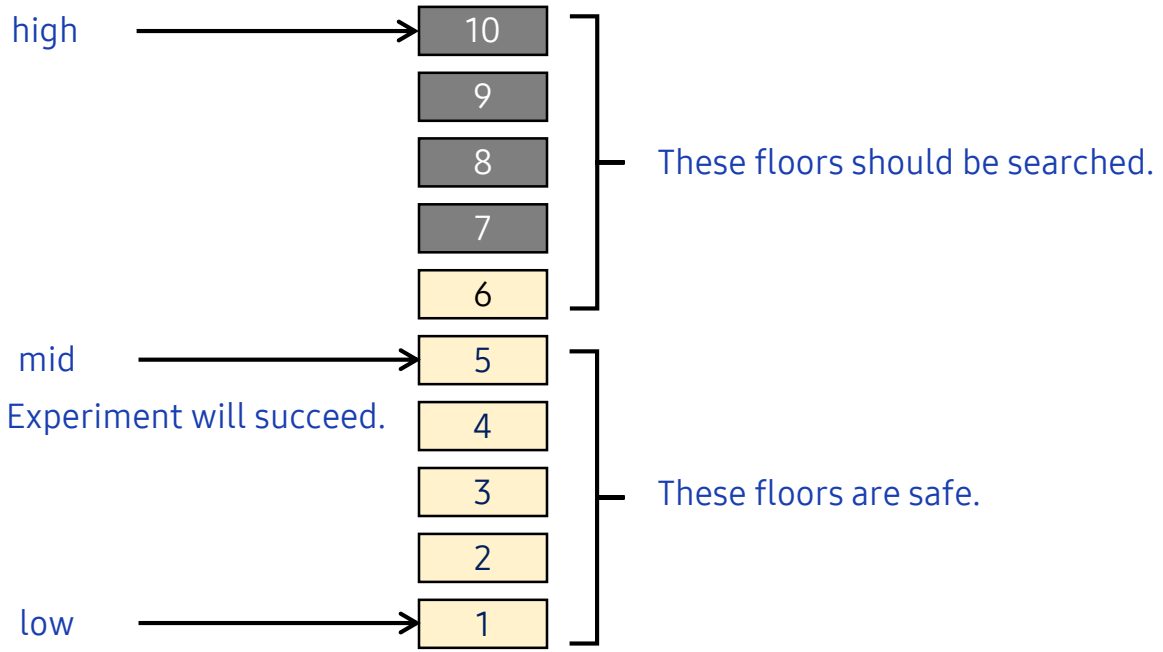
- ທົດລອງຢູ່ອາຄານຊັ້ນທີ N=10.
- ສົມມຸດວ່າການປ່ອຍໄຂ່ໃສ່ຊັ້ນຕ່ຳກວ່າຊັ້ນທີ 6 ໄຂ່ຍັງບໍ່ແຕກ ແລະ ປ່ອຍໄຂ່ຈາກຊັ້ນທີ່ສູງກວ່າຊັ້ນທີ 7 ໄຂ່ແຕກ.



2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່ທີ່ມີໄຂ່ບໍ່ຈຳກັດ

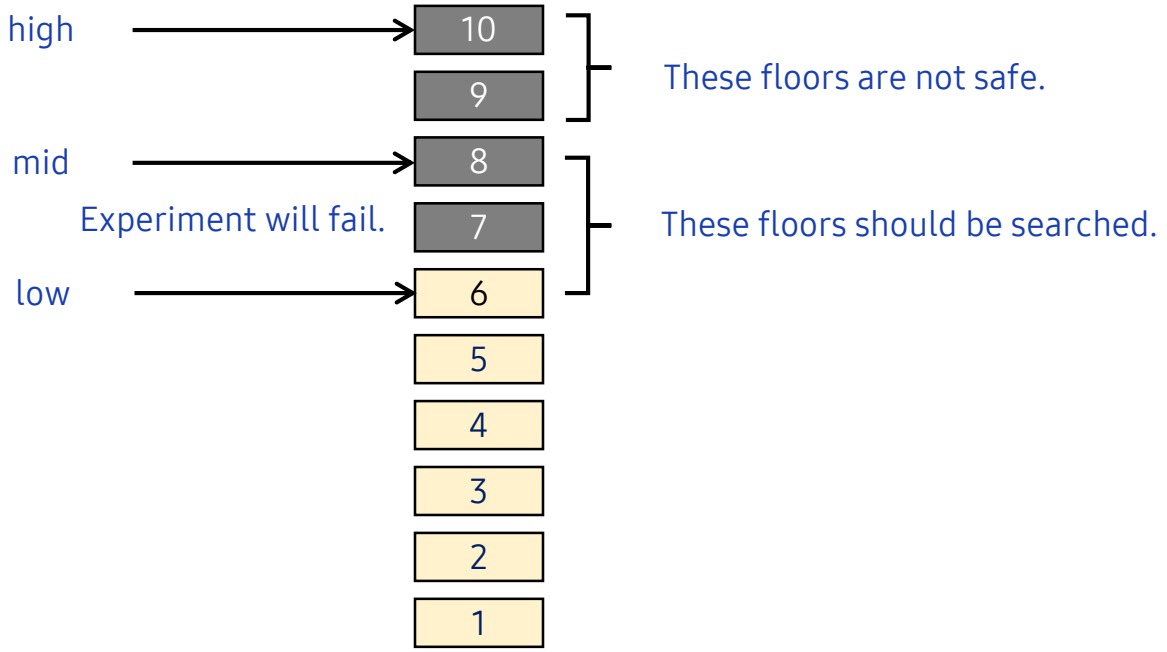
- ກ່ອນອື່ນ, ໃຫ້ເຮັດການທົດລອງຢູ່ຊັ້ນທີ 5 ເຊິ່ງເປັນຈຸດກາງຂອງຊັ້ນທີ 1 ແລະຊັ້ນທີ 10.
- ຖ້າໄຂ່ບໍ່ແຕກ, ຊັ້ນທີ່ໄຂ່ແຕກຈະສູງກວ່າຊັ້ນທີ 5.



2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່ທີ່ມີໄຂ່ບໍ່ຈຳກັດ

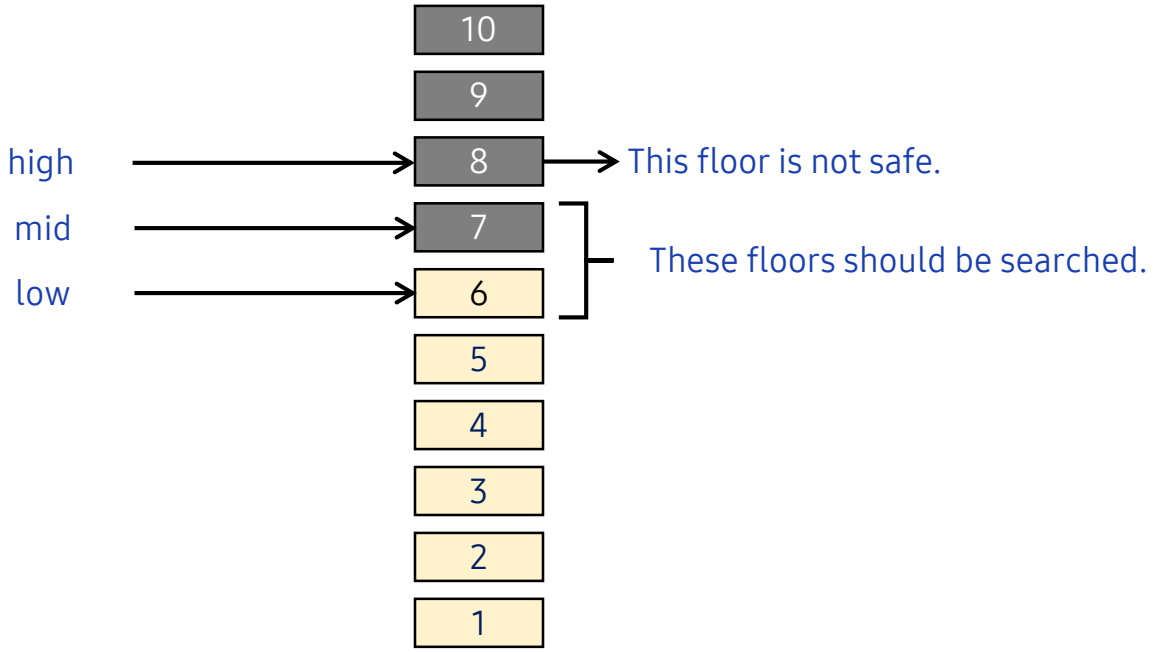
- ໃນປັດຈຸບັນ, ສ່ວນທີ່ຈະສຶບຄົ້ນແມ່ນເລີ່ມຕັ້ງແຕ່ຊັ້ນທີ 6 ຫາຊັ້ນທີ 10.
- ເຊັ່ນດຽວກັນ, ຖ້າໄຂ່ແຕກຢູ່ຊັ້ນ 8 ເຊິ່ງເປັນຈຸດກາງຂອງຊ່ວງນີ້, ໝາຍຄວາມວ່າຊັ້ນທຳອິດທີ່ໄຂ່ແຕກຈະເປັນຊັ້ນ 8 ຫຼື ຊັ້ນລຸ່ມ.



2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່ທີ່ມີໄຂ່ບໍ່ຈຳກັດ

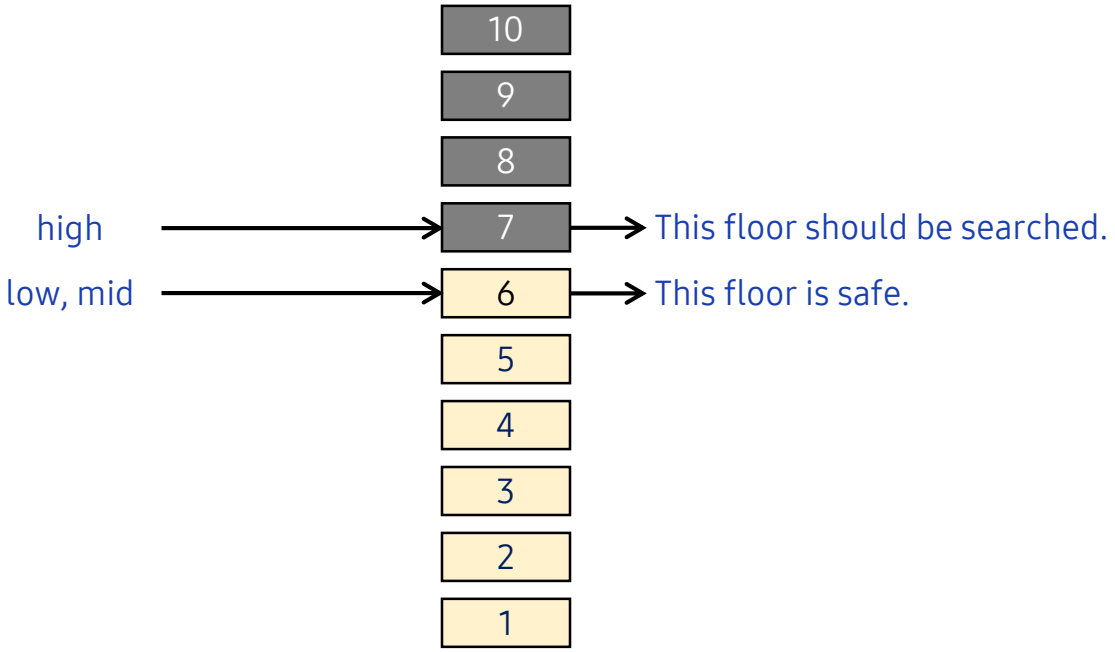
- ສ່ວນຕໍ່ໄປພວກເຮົາຈະສຶບຄົ້ນແມ່ນຕັ້ງແຕ່ຊັ້ນທີ 6 ຫາຊັ້ນ 8.
- ເຊັ່ນດຽວກັນ, ຖ້າໄຂ່ແຕກຢູ່ຊັ້ນ 7 ເຊິ່ງເປັນຈຸດກາງຂອງຊ່ວງນີ້, ສະແດງວ່າຊັ້ນທຳອິດທີ່ໄຂ່ແຕກຈະເປັນຊັ້ນ 7 ຫຼື ຊັ້ນລຸ່ມ.



2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່ທີ່ມີໄຂ່ບໍ່ຈຳກັດ

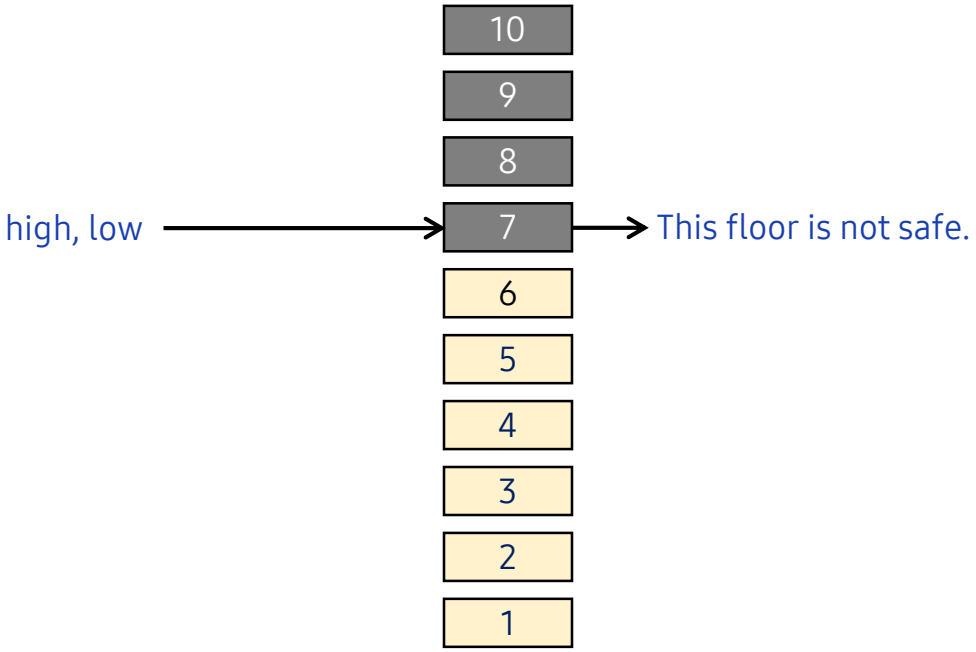
- ຂະນະນີ້ພວກເຮົາຄົ້ນຫາຢູ່ຊັ້ນ 6 ແລະ 7.
- ສົມມຸດວ່າຈຸດກາງແມ່ນຊັ້ນທີ 6 ແລະ ຖ້າໄຂ່ບໍ່ແຕກຈາກຊັ້ນທີ 6, ຊັ້ນທຳອິດທີ່ໄຂ່ແຕກຈະສູງກວ່າຊັ້ນທີ 6.



2. Mission

2.1. ບັນຫາການຕົກຂອງໄຂ່ທີ່ມີໄຂ່ບໍ່ຈຳກັດ

- ສ່ວນຕໍ່ໄປທີ່ຕ້ອງຄົ້ນຫາແມ່ນຊັ້ນທີ 7.
- ເຫຼືອພຽງຊັ້ນດຽວເທົ່ານັ້ນທີ່ຕ້ອງກວດສອບ ແລະ ເນື່ອງຈາກພວກເຮົາໄດ້ຊອກຫາຊັ້ນຕ່ຳສຸດທີ່ໄຂ່ແຕກ, ຊັ້ນທີ 6 ຖືວ່າ ເປັນຊັ້ນສູງສຸດທີ່ໄຂ່ບໍ່ແຕກ.



3. ການແກ້ໄຂບັນຫາ

3.1. ວິທີການແກ້ໄຂບັນຫາເຮັດວຽກ

ໄດ້ຮັບການປ້ອນຂໍ້ມູນຂອງຜູ້ໃຊ້ກ່ຽວກັບຄວາມສູງຂອງອາຄານແລະຊັ້ນທີ່ໄຂ່ແຕກເພື່ອຊອກຫາຊັ້ນສູງສຸດທີ່ໄຂ່ບໍ່ແຕກ.

```
1 height = int(input("Input the number of floors: "))
2 breaking = int(input("Input the first breaking floor: "))
3 floor = find_highest_safe_floor2(height, breaking)
4 print(f"Your egg will safe till the {floor}-th floor.")
```

```
Input the number of floors: 10
Input the first breaking floor: 7
Your egg will safe till the 6-th floor.
```

3. ການແກ້ໄຂບັນຫາ

3.2. code ສຸດທ້າຍຂອງບັນຫາການປ່ອຍໄຂ່

```python
def do_experiment(floor, breaking):
    return floor >= breaking

def find_highest_safe_floor2(height, breaking):
    low, high = 1, height
    while low < high:
        mid = (low + high) // 2
        if do_experiment(mid, breaking):
            high = mid
        else:
            low = mid + 1
    return low - 1
```

Key concept

1. Binary Search

1.1. ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບ binary search

ພິຈາລະນາວິທີການຄົ້ນຫາແບບ binary search ທີ່ຊອກຫາຈຳນວນຖ້ວນທີ່ກຳນົດຈາກ list ຂອງຈຳນວນຖ້ວນທີ່ໄດ້ລຽງລຳດັບແລ້ວ.

Focus ຕໍ່ໄປນີ້ແມ່ນບັນຫາ binary search algorithm, ແລະ ຂໍ້ມູນປ້ອນເຂົ້າ ແລະ ຜົນໄດ້ຮັບຂອງມັນ.

- ບັນຫາ: ຈຳນວນຖ້ວນ x ໃດໜຶ່ງຢູ່ໃນ S ຊຶ່ງເປັນຈຳນວນຖ້ວນທີ່ໄດ້ລຽງລຳດັບໄວ້ແລ້ວ?
- ຂໍ້ມູນປ້ອນເຂົ້າ: S ເປັນ list ຂອງຈຳນວນຖ້ວນທີ່ຈັດລຽງຕາມລຳດັບຈາກນ້ອຍຫາໃຫຍ່, x ເປັນຈຳນວນຖ້ວນແບບສຸ່ມເພື່ອຄົ້ນຫາ
- ຜົນໄດ້ຮັບ: ຕຳແໜ່ງທີ່ຢູ່ຂອງ x, ຖ້າ x ມີຢູ່ໃນ list S ແລະ -1 ຖ້າບໍ່ມີ.

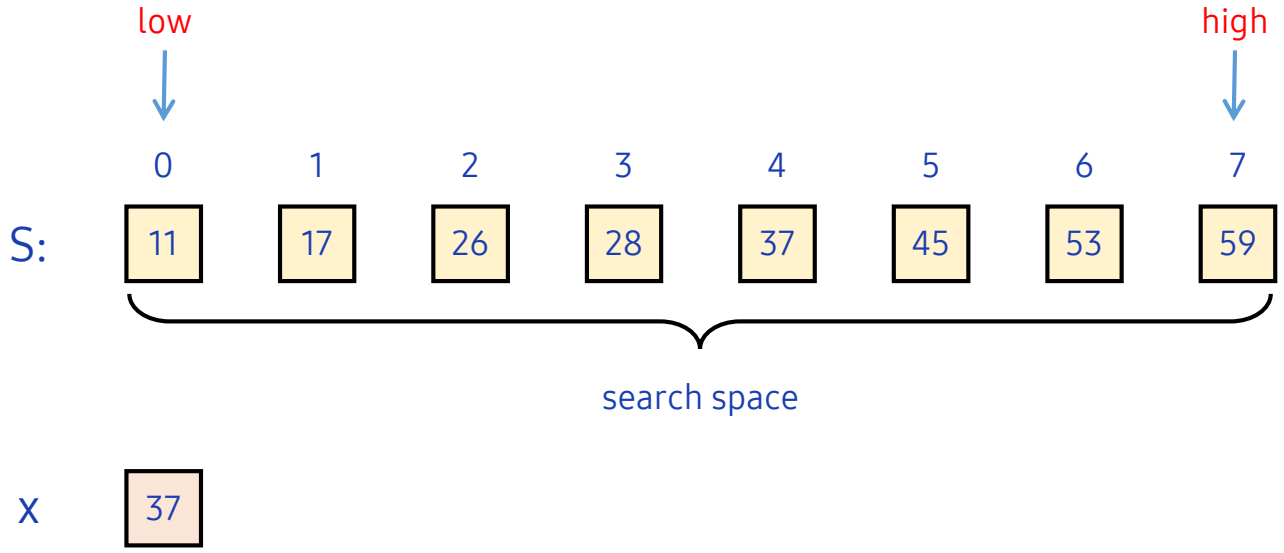
```
1 S = [11, 17, 26, 28, 37, 45, 53, 59]
2 x = int(input("Input the number to search: "))
3 pos = bin_search(S, x)
4 print(f"In S, {x} is at position {pos}.")
```

```
Input the number to search: 53
In S, 53 is at position 6.
```

1. Binary Search

1.1. ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບ binary search

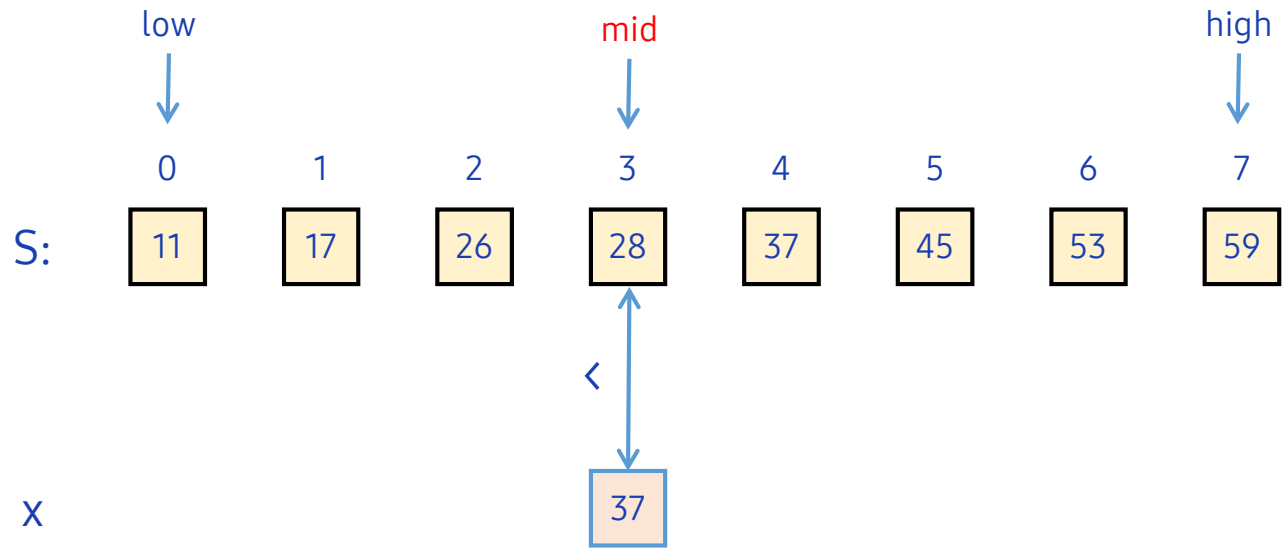
- Binary search algorithm ເປັນການຄົ້ນຫາແບບແບ່ງເຄິ່ງຈະຊອກຫາຄ່າ x ໃນ list S.
- ຊ່ວງທີ່ຄົ້ນຫາເບື້ອງຕົ້ນແມ່ນລະຫວ່າງຄ່າຕ່ຳສຸດແລະສູງສຸດ (ຕ່ຳສຸດແມ່ນດັດຊະນີອົງປະກອບທີ 1 ແລະ ຄ່າສູງສຸດແມ່ນດັດຊະນີອົງປະກອບສຸດທ້າຍ) ໃນ S.



1. Binary Search

1.1. ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບ binary search

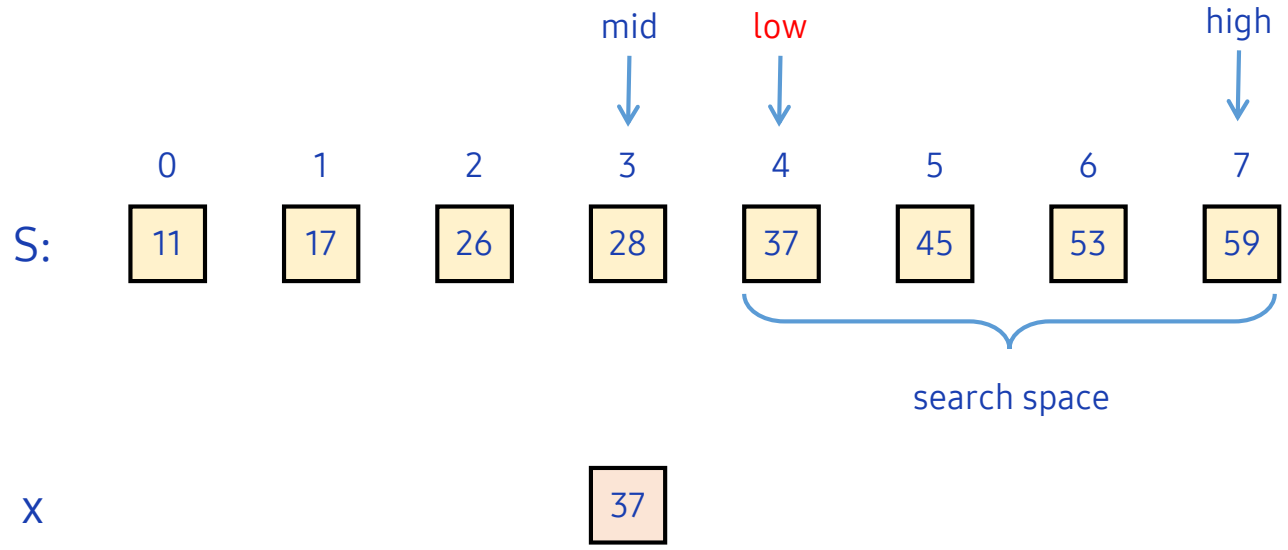
- ໃນ binary search algorithm, ລາຍການຂໍ້ມູນປ້ອນເຂົ້າ ຖືກຈັດລຽງຕາມລຳດັບຈາກນ້ອຍຫາໃຫຍ່.
- ປຽບທຽບຄ່າ x ກັບຄ່າກາງ ທີ່ຢູ່ລະຫວ່າງກາງຂອງຄ່າຕ່ຳສຸດ ແລະ ຄ່າສູງສຸດ.



1. Binary Search

1.1. ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບ binary search

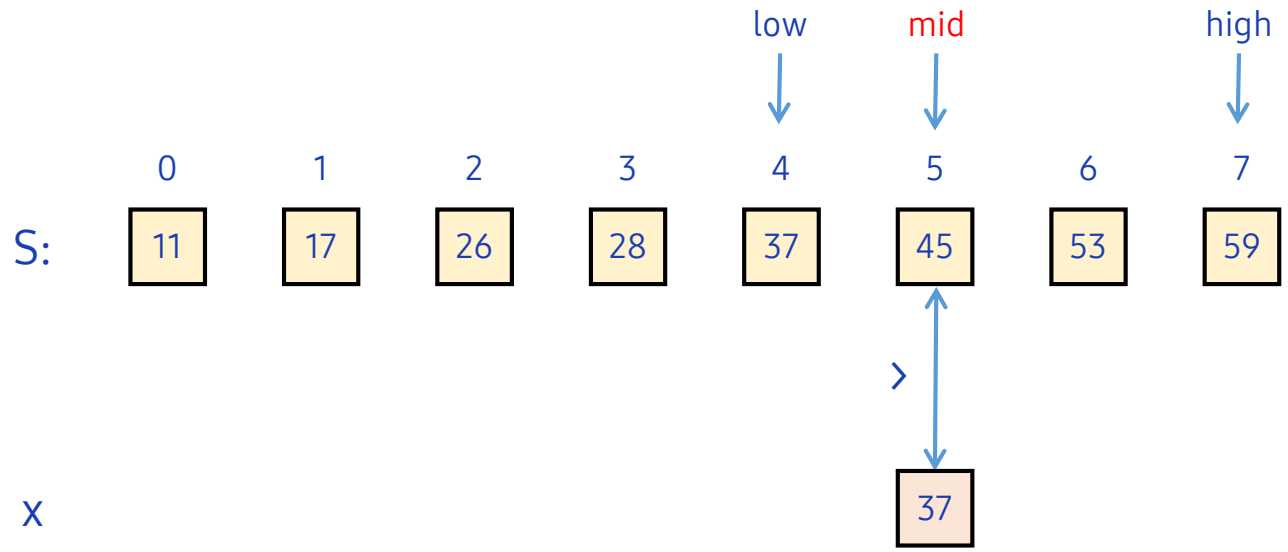
- ຖ້າ S[mid] ນ້ອຍກວ່າ x, x ທີ່ຕ້ອງການຄົ້ນຫາຈະຢູ່ລະຫວ່າງ mid+1 ແລະ high.
- ດັ່ງນັ້ນ, ຄ່າຕ່ຳສຸດຈະຖືກປ່ຽນເປັນ mid+1, ເຊິ່ງເປັນເຄິ່ງຫນຶ່ງຂອງຊ່ວງທີ່ຄົ້ນຫາຈາກ [0, 7] ຫຼຸດລົງເປັນ [4, 7].



1. Binary Search

1.1. ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບ binary search

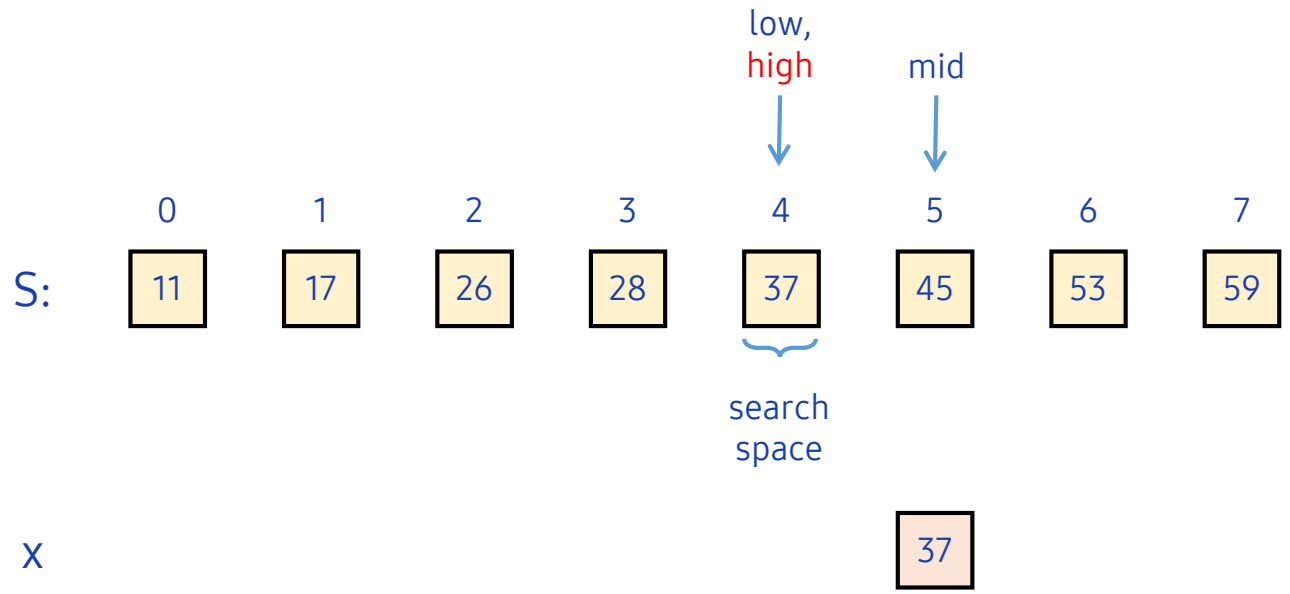
- ຄິດໄລ່ຄ່າ ' mid' ເຊິ່ງຢູ່ໃນລະຫວ່າງຄ່າຕ່ຳສຸດ ແລະ ຄ່າສູງສຸດ ເພື່ອປຽບທຽບ S[mid] ແລະ x.
- ກໍລະນີ S[mid] ໃຫຍ່ກວ່າ x, ດັ່ງນັ້ນ high ຈະຖືກປ່ຽນເປັນ mid-1.



1. Binary Search

1.1. ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບ binary search

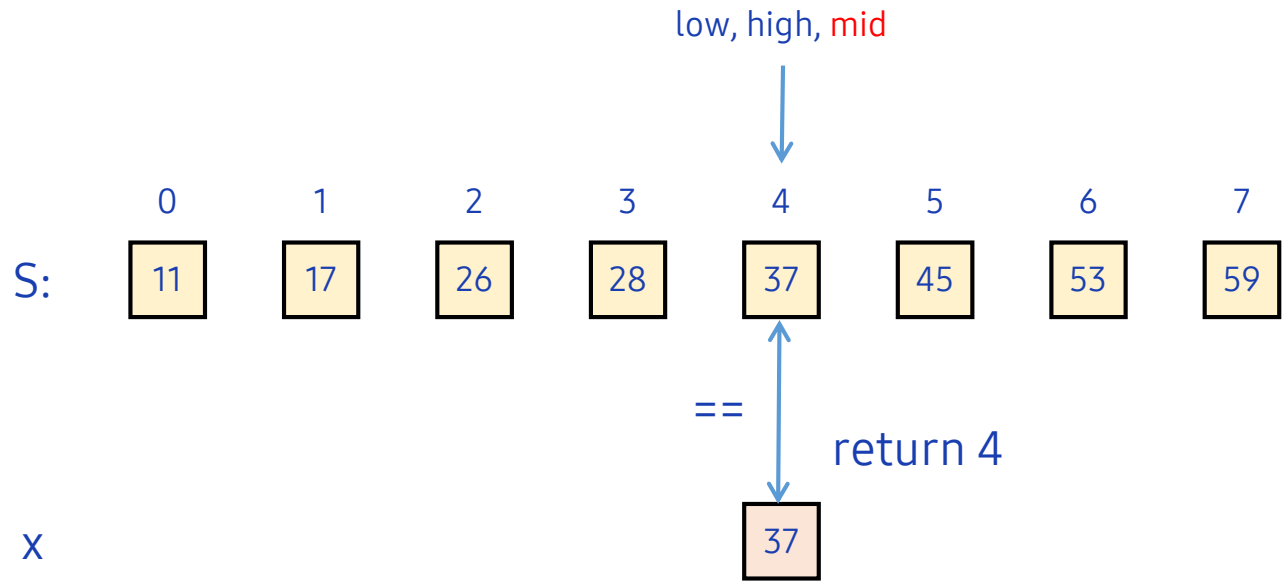
- ຊ່ວງການຄົ້ນຈະຖືກຫຼຸດລົງຈາກ [4, 7] ເປັນ [4, 4].



1. Binary Search

1.1. ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບ binary search

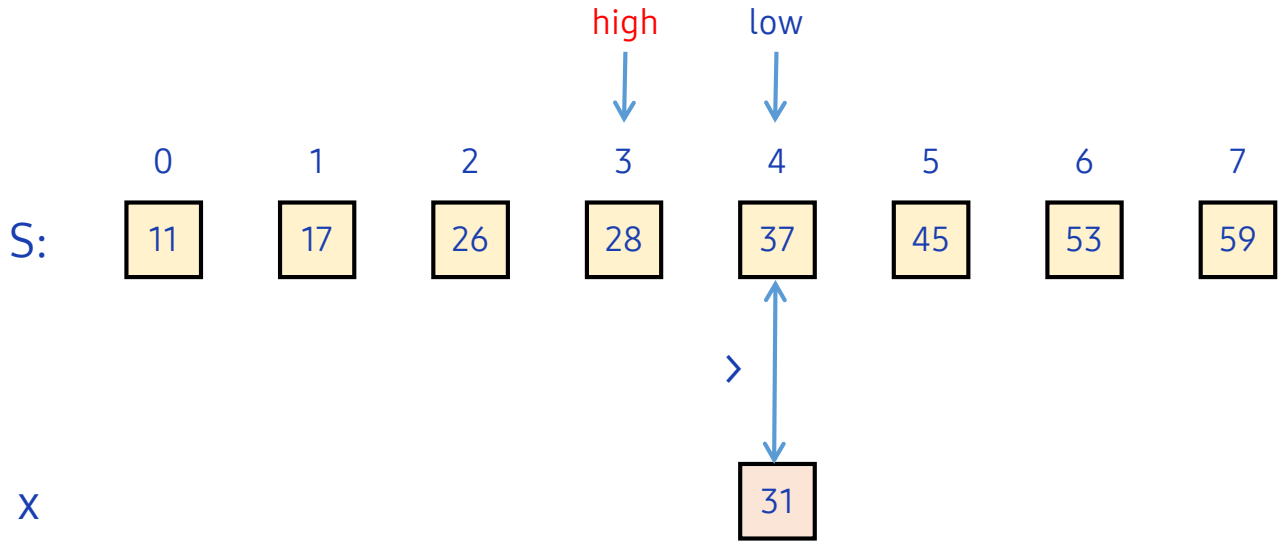
- ຄິດໄລ່ຄ່າ ' mid' ທີ່ຢູ່ລະຫວ່າງຄ່າຕ່ຳສຸດ ແລະ ຄ່າສູງສຸດ ເພື່ອປຽບທຽບ S[mid] ແລະ x.
- ເນື່ອງຈາກຄ່າ S[mid] ແລະ x ແມ່ນຄືກັນ, ໃຫ້ສົ່ງຄືນຄ່າກາງຢູ່ທີ່ຕຳແໜ່ງປັດຈຸບັນ.



1. Binary Search

1.1. ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບ binary search

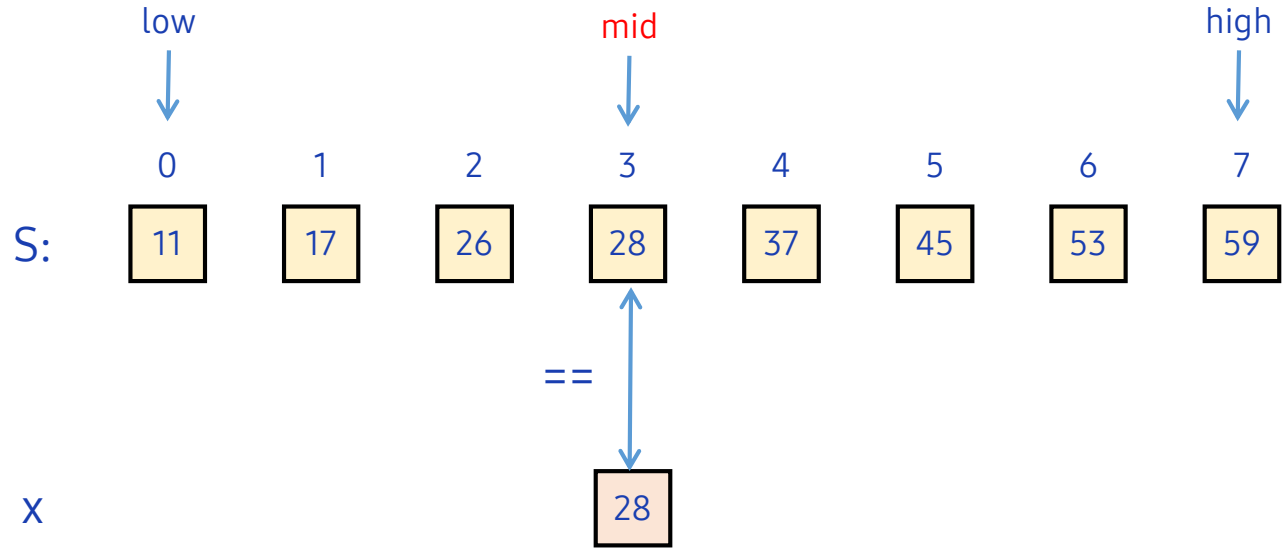
- ຖ້າຄ່າທີ່ຊອກຫາບໍ່ມີຢູ່ໃນລາຍການ, ຕົວຢ່າງ ຄ່າ 31, ເນື່ອງຈາກວ່າ x ນ້ອຍກວ່າ S[mid], high ຈະປ່ຽນເປັນ mid-1.
- ຕາມເງື່ອນໄຂ low < high ຊຶ່ງເຫັນວ່າບໍ່ຊອດຄ່ອງແລ້ວ, ສະນັ້ນໃຫ້ອອກຈາກການຄົ້ນຫາ ແລະ ສົ່ງຄືນ -1.



1. Binary Search

1.1. ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບ binary search

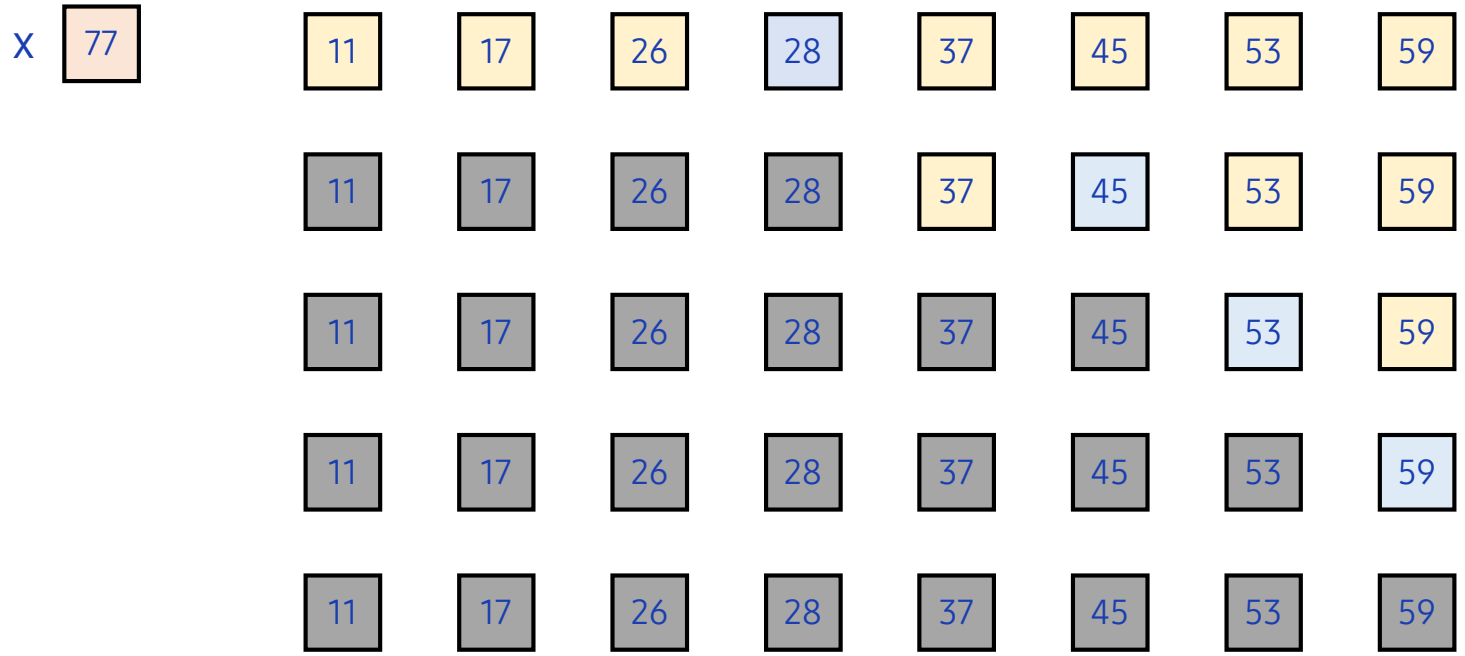
ກໍລະນີທີ່ດີທີ່ສຸດໃນການຄົ້ນແບບ binary search ແມ່ນຫຍັງ? ໃນກໍລະນີໂຊກດີທີ່ຄົ້ນຫາຄ່າກາງໃນ S, ພຽງແຕ່ການປຽບທຽບຄັ້ງດຽວກໍ່ພົບຄ່າທີ່ຕ້ອງການແລ້ວ.



1. Binary Search

1.1. ຂັ້ນຕອນວິທີການຄົ້ນຫາແບບ binary search

- ກໍລະນີທີ່ບໍ່ດີທີ່ສຸດໃນການຄົ້ນແບບ binary search ແມ່ນຫຍັງ? ຄ້າຍຄືກັນກັບການຄົ້ນຫາແບບຕາມລຳດັບ, ຈຳນວນການປຽບທຽບສູງສຸດແມ່ນການຄົ້ນຫາທັງໝົດແລ້ວບໍ່ພົບອົງປະກອບທີ່ຕ້ອງການໃນ S.
- ໃນຕົວຢ່າງຕໍ່ໄປນີ້, ຈຳນວນການປຽບທຽບທັງຫມົດ 4 ຄັ້ງຄວນຈະເຮັດເພື່ອຊອກຫາຄ່າ 77.



1. Binary Search

1.3. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

Binary search algorithm ຈະຖືກສ້າງດັ່ງຕໍ່ໄປນີ້. ໄດ້ຮັບ ' nums' ແລະ x ເປັນຂໍ້ມູນປ້ອນເຂົ້າເພື່ອກຳນົດ nums ທັງຫມົດເປັນ ເນື້ອທີ່ຄົ້ນຫາ.

```
1  def bin_search(nums, x):
2      low, high = 0, len(nums) - 1
3      while low <= high:
4          mid = (low + high) // 2
5          if nums[mid] == x:
6              return mid
7          elif nums[mid] > x:
8              high = mid - 1
9          else:
10             low = mid + 1
11     return -1
```

Line 2

- ' low' ແລະ ' high' ແມ່ນເລີ່ມຕົ້ນເປັນອົງປະກອບທຳອິດແລະສຸດທ້າຍຂອງ nums, ຕາມລຳດັບ.
- ໃນສ່ວນເລີ່ມຕົ້ນຂອງ binary search, numbs ທັງຫມົດແມ່ນເນື້ອທີ່ຄົ້ນຫາ. ດັ່ງນັ້ນ, ເນື້ອທີ່ຄົ້ນຫາແມ່ນ [0, len(num)-1].

1. Binary Search

1.3. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

Binary search ປຽບທຽບຄ່າກາງທີ່ຢູ່ໃນລະຫວ່າງ ' low' ແລະ ' high'. ເນື່ອງຈາກຂໍ້ມູນຖືກຈັດລຳດັບແລ້ວ, ການປຽບທຽບພຽງຄັ້ງດຽວສາມາດຫຼຸດລົງເນື່ອທີ່ການຄົ້ນຫາລົງເຄິ່ງຫນຶ່ງ.

```
1  def bin_search(nums, x):
2      low, high = 0, len(nums) - 1
3      while low <= high:
4          mid = (low + high) // 2
5          if nums[mid] == x:
6              return mid
7          elif nums[mid] > x:
8              high = mid - 1
9          else:
10             low = mid + 1
11     return -1
```

Line 3-10

- ຄ່າກາງແມ່ນຄິດໄລ່ເປັນ (low + high) // 2. ໃນທີ່ນີ້, ຜູ້ຕ໋ອດຳເນີນການ // ຈະສົ່ງຄືນຄ່າ ຄ່າຖ້ວນຂອງຜົນຫານ.
- ຖ້າຄ່າ nums[mid] ເທົ່າກັບ x, ການຄົ້ນຫາແມ່ນສຳເລັດແລ້ວ ດັ່ງນັ້ນໃຫ້ສົ່ງຄືນຄ່າກາງດັ່ງກ່າວ.
- ຖ້າຄ່າ nums[mid] ໃຫຍ່ກວ່າ x, ປ່ຽນ ‘high’ ເປັນ mid-1 ແລະ ຖ້າມັນນ້ອຍກວ່າ x, ປ່ຽນ ‘low’ ເປັນ mid + 1.
- ຖ້າຄ່າ low ຫຼື high ປ່ຽນເປັນ mid + 1 ຫຼື mid-1 ແລ້ວ ເນື້ອທີ່ຄົ້ນຫາຈະຫຼຸດລົງເຄິ່ງໜຶ່ງ.

1. Binary Search

1.3. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ສົ່ງຄືນຄ່າ -1 ຖ້າ x ບໍ່ພົບໃນຂະບວນການຄົ້ນຫາແບບ binary search.

```
1  def bin_search(nums, x):
2      low, high = 0, len(nums) - 1
3      while low <= high:
4          mid = (low + high) // 2
5          if nums[mid] == x:
6              return mid
7          elif nums[mid] > x:
8              high = mid - 1
9          else:
10             low = mid + 1
11     return -1
```

Line 11

- ຖ້າຄ່າ low ໃຫຍ່ກວ່າຄ່າ high, ໃຫ້ຢຸດການຄົ້ນຫາ ສະແດງວ່າ ຂໍ້ມູນທີ່ຕ້ອງການບໍ່ມີຢູ່ໃນໂຄງສ້າງຂໍ້ມູນນີ້.
- ຖ້າເປັນດັ່ງນັ້ນ, ໃຫ້ສົ່ງຄືນຄ່າ -1 ເພາະວ່າຄ່າ x ບໍ່ມີຢູ່ໃນ S.

One More Step

- ເວລາຂອງການຄົ້ນຫາແບບ binary search ແມ່ນເທົ່າໃດ? ຫນ້າທໍາອິດ, ກໍລະນີທີ່ດີທີ່ສຸດແມ່ນຄົ້ນຫາພຽງແຕ່ຄັ້ງດຽວແລ້ວພົບຂໍ້ມູນທີ່ຕ້ອງການເລີຍ, ດັ່ງນັ້ນມັນແມ່ນ O(1). ຫຼັງຈາກນັ້ນ, ຈະວິເຄາະເວລາທີ່ໃຊ້ໃນກໍລະນີທີ່ບໍ່ດີທີ່ສຸດຂອງ binary search ແມ່ນເທົ່າໃດ?
- ສົມມຸດວ່າ S ມີອົງປະກອບ N ອັນ. ຂະໜາດຂອງເນື້ອທີ່ຄົ້ນຫາທໍາອິດແມ່ນ N. ເນື້ອທີ່ການຄົ້ນຫາຮອບທີສອງໃນການຄົ້ນຫາແມ່ນຫຼຸດລົງເຄິ່ງໜຶ່ງຂອງເນື້ອທີ່ທໍາອິດ, ສະນັ້ນມັນແມ່ນ N/2. ເນື້ອທີ່ຄົ້ນຫາຕໍ່ໄປແມ່ນຫຼຸດລົງເຄິ່ງຫນຶ່ງອີກເທື່ອຫນຶ່ງ, ເຊິ່ງແມ່ນ N/4. ຄໍາອະທິບາຍນີ້ສາມາດຂຽນໂດຍທົ່ວໄປໄດ້ດັ່ງຕໍ່ໄປນີ້.

$$N, \frac{N}{2}, \frac{N}{2^2}, \cdots, \frac{N}{2^k}$$

- ການຄົ້ນຫາແບບ Binary search ຈະສໍາເລັດຖ້າມີບ່ອນຊອກຫາພຽງອັນດຽວ, ເຊິ່ງແມ່ນ $N/2^k=1$.

$$N = 2^k$$

$$\log_2 N = k$$

- ຄ່າສູງສຸດຂອງຈໍານວນການຄົ້ນຫາແບບ binary search ແມ່ນ $\log_2 N$, ດັ່ງນັ້ນ ກໍລະນີທີ່ບໍ່ດີທີ່ສຸດຂອງການຄົ້ນຫາແບບ binary search ແມ່ນ $O(\log_2 N)$.

1. Binary Search

1.4. ปะสิดทิพาบภามปะเมินผิน

เพิ่มฟังชัน print() ใส่ bin_search() เพื่อกวดสอบเวลาของภามค้นแบบ binary search.

```
1  def bin_search(nums, x):
2      low, high = 0, len(nums) - 1
3      while low <= high:
4          mid = (low + high) // 2
5          print(low, high, mid)
6          if nums[mid] == x:
7              return mid
8          elif nums[mid] > x:
9              high = mid - 1
10         else:
11             low = mid + 1
12     return -1
```

Line 5

- พิมถ่า low, high และ mid ใม loop เพื่อนับจำนอมภามเธ้ดอวูกซ้ำของ binary search.

1. Binary Search

1.4. ປະສິດທິພາບການປະເມີນຜົນ

ໃນ list ທີ່ມີ 8 ອົງປະກອບ, ການຄົ້ນຫາແມ່ນສຳເລັດຫຼັງຈາກການດຳເນີນງານຫນຶ່ງຄັ້ງໃນກໍລະນີທີ່ດີທີ່ສຸດ, ໃນຂະນະທີ່ການປະຕິບັດໃນກໍລະນີທີ່ບໍ່ດີທີ່ສຸດແມ່ນສີ່ຄັ້ງ.

```
1 S = [11, 17, 26, 28, 37, 45, 53, 59]
2 x = int(input("Input the number to search: "))
3 pos = bin_search(S, x)
4 print(f"In S, {x} is at position {pos}.")
```

```
Input the number to search: 28
0 7 3
In S, 28 is at position 3.
```

```
1 S = [11, 17, 26, 28, 37, 45, 53, 59]
2 x = int(input("Input the number to search: "))
3 pos = bin_search(S, x)
4 print(f"In S, {x} is at position {pos}.")
```

```
Input the number to search: 77
0 7 3
4 7 5
6 7 6
7 7 7
In S, 77 is at position -1.
```

Let’s code

1. ບັນຫາການຕົກຂອງໄຂ່

1.1. ນິຍາມບັນຫາ

- ຂຽນ algorithm ເພື່ອຊອກຫາຊັ້ນສູງທີ່ສຸດທີ່ໄຂ່ບໍ່ແຕກໄຂ່ເມື່ອມັນຕົກລົງມາ ໃນຄວາມສູງຂອງອາຄານທີ່ໃຫ້.
- ເງື່ອນໄຂການສ້າງ algorithm ມີດັ່ງນີ້.

1. ' breaking,' ເຊິ່ງເປັນຊັ້ນຕ່ໍາສຸດທີ່ໄຂ່ແຕກ, ຖືກເລືອກແບບສຸ່ມລະຫວ່າງ 1 ແລະຄວາມສູງ.
2. ການທົດລອງປ່ອຍໄຂ່ແມ່ນສະແດງດ້ວຍຟັງຊັນ do_experiment(floor).
   - ສົ່ງຄືນຄ່າ True ຖ້າໄຂ່ແຕກເມື່ອປ່ອຍລົງຈາກພື້ນ.
   - ຖ້າບໍ່ແມ່ນ, ສົ່ງຄືນຄ່າ False.
3. ຟັງຊັນ do_experiment() ສາມາດເອີ້ນໃຊ້ໂດຍບໍ່ຈໍາກັດ ເພາະມີໄຂ່ບໍ່ຈໍາກັດ.

1. ບັນຫາການຕົກຂອງໄຂ່

1.2. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ໄດ້ຮັບຂໍ້ມູນປ້ອນເຂົ້າຈາກຜູ້ໃຊ້ກ່ຽວກັບຄວາມສູງຂອງອາຄານ ' height' ແລະ ' 'breaking' ຊຶ່ງເປັນຊັ້ນຕ່ຳສຸດທີ່ເຮັດໃຫ້ໄຂ່ແຕກ.

```
1 height = int(input("Input the number of floors: "))
2 breaking = int(input("Input the first breaking floor: "))
3 floor = find_highest_safe_floor2(height, breaking)
4 print(f"Your egg will safe till the {floor}-th floor.")
```

```
Input the number of floors: 100
Input the first breaking floor: 93
Your egg will safe till the 92-th floor.
```

Line 1-2

- ປ່ຽນຄ່າຄວາມສູງ ແລະ ຄ່າ breaking ທີ່ຜູ້ໃຊ້ເປ້ົາເຂົ້າໂດຍຟັງຊັນ input() ເຂົ້າໄປໃນຟັງຊັນ int() ແລະ ເກັບໄວ້.

1. ບັນຫາການຕົກຂອງໄຂ່

1.2. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ເອີ້ນໃຊ້ຟັງຊັນທີ່ສົ່ງຄືນຄ່າຂອງຊັ້ນສູງສຸດທີ່ໄຂຍັງບໍ່ແຕກ.

```
1 height = int(input("Input the number of floors: "))
2 breaking = int(input("Input the first breaking floor: "))
3 floor = find_highest_safe_floor2(height, breaking)
4 print(f"Your egg will safe till the {floor}-th floor.")
```

```
Input the number of floors: 100
Input the first breaking floor: 93
Your egg will safe till the 92-th floor.
```

Line 3-4

- ຟັງຊັນ find_highest_safe_floor2() ຊອກຫາຊັ້ນທີ່ສູງທີ່ສຸດທີ່ໄຂ່ບໍ່ແຕກໂດຍຜ່ານ binary search.
- ໃຊ້ f-string ເພື່ອພິມຕົວປ່ຽນ floor, ເຊິ່ງຄວນຈະແມ່ນ -1 ຫຼາຍກ່ວາຄ່າ breaking.

1. ບັນຫາການຕົກຂອງໄຂ່

1.2. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ฟังชัน do_experiment() แม่นใช้สำลับการทิดลองป่อยไข่.

```
1  def do_experiment(floor, breaking):
2      return floor >= breaking
3
4  def find_highest_safe_floor2(height, breaking):
5      low, high = 1, height
6      while low < high:
7          mid = (low + high) // 2
8          if do_experiment(mid, breaking):
9              high = mid
10         else:
11             low = mid + 1
12     return low - 1
```

Line 1-2

- ' breaking' แม่นชั้นต่ำสุดที่ไข่แตก.
- ถ้าค่า floor เท่ากับ ຫຼື ใหย่กว่า breaking, ให้สิ่งคืนค่า True และ ถ้าบ่แม่น, ให้สิ่งคืนค่า False.

1. ບັນຫາການຕົກຂອງໄຂ່

1.2. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ฟังชัน find_highest_safe_floor2() ຊອກຫາຊັ້ນທີ່ສູງທີ່ສຸດທີ່ໄຂ່ບໍ່ແຕກໂດຍຜ່ານbinary search.

```
1  def do_experiment(floor, breaking):
2      return floor >= breaking
3
4  def find_highest_safe_floor2(height, breaking):
5      low, high = 1, height
6      while low < high:
7          mid = (low + high) // 2
8          if do_experiment(mid, breaking):
9              high = mid
10         else:
11             low = mid + 1
12     return low - 1
```

Line 4-5

- ເລີ່ມຕົ້ນເນື້ອທີ່ຄົ້ນຫາເບື້ອງຕົ້ນສຳລັບ binary search ໃຫ້ແກ່ຄ່າ low ແລະ high.
- ເລີ່ມຕົ້ນ ' low ' ກຳນົດເປັນຊັ້ນທີ 1 ແລະ ' high ' ກຳນົດເປັນຊັ້ນສູງສຸດ ' height ' ຂອງອາຄານ.

1. ບັນຫາການຕົກຂອງໄຂ່

1.2. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ຊອກຫາຊັ້ນທີ 1 ບ່ອນທີ່ໄຂ່ແຕກໂດຍການຫຼຸດເນື້ອທີ່ຄົ້ນຫາເປັນເຄິ່ງຫນຶ່ງໂດຍຜ່ານ binary search.

```
1  def do_experiment(floor, breaking):
2      return floor >= breaking
3
4  def find_highest_safe_floor2(height, breaking):
5      low, high = 1, height
6      while low < high:
7          mid = (low + high) // 2
8          if do_experiment(mid, breaking):
9              high = mid
10         else:
11             low = mid + 1
12     return low - 1
```

Line 6-11

- ກຳນົດ mid ເປັນຊັ້ນຢູ່ລະຫວ່າງຊັ້ນຕ່ຳສຸດ ແລະສູງສຸດ ແລະ ຫຼັງຈາກນັ້ນດຳເນີນການທົດລອງ.
- ຖ້າໄຂ່ບໍ່ແຕກ, ໃຫ້ປ່ຽນ low ເປັນ mid+1 ເພື່ອຫຼຸດເນື້ອທີ່ຄົ້ນຫາເຂົ້າໄປໃນຊັ້ນສູງກວ່າ.
- ຖ້າໄຂ່ແຕກ, ໃຫ້ປ່ຽນ high ເປັນ mid -1 ເພື່ອຫຼຸດເນື້ອທີ່ຄົ້ນຫາເຂົ້າໄປໃນຊັ້ນຕ່ຳ.
- ອອກຈາກການຄົ້ນຫາ ຖ້າຄ່າ low ມີຄ່າຫຼາຍກວ່າຫຼືເທົ່າກັບຄ່າ high.

1. ບັນຫາການຕົກຂອງໄຂ່

1.2. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ຫຼັງຈາກການຄົ້ນຫາສຳເລັດແລ້ວ, ໃຫ້ສົ່ງຄືນຄ່າຊັ້ນສູງສຸດທີ່ໄຂ່ບໍ່ແຕກ.

```
1  def do_experiment(floor, breaking):
2      return floor >= breaking
3
4  def find_highest_safe_floor2(height, breaking):
5      low, high = 1, height
6      while low < high:
7          mid = (low + high) // 2
8          if do_experiment(mid, breaking):
9              high = mid
10         else:
11             low = mid + 1
12     return low - 1
```

Line 12

- ຈາກການຄົ້ນຫາແບບ binary search, ຜົນໄດ້ຮັບແມ່ນຊັ້ນຕ່ຳສຸດທີ່ໄຂ່ແຕກ.
- ດັ່ງນັ້ນ, ຊັ້ນທີ່ສູງທີ່ສຸດທີ່ໄຂ່ບໍ່ແຕກຈະສົ່ງຄືນຄ່າ low-1 ຫຼັງຈາກການຄົ້ນຫາສຳເລັດ.

1. ບັນຫາການຕົກຂອງໄຂ່

1.3. ປະສິດທິພາບການປະເມີນ

ເພີ່ມຟັງຊັນ print() ເພື່ອກວດເບິ່ງວ່າ ການຄົ້ນຫາແບບ binary search ປະຕິບັດຈັກຄັ້ງໃຫ້ສຳເລັດ.

```
1  def do_experiment(floor, breaking):
2      return floor >= breaking
3
4  def find_highest_safe_floor2(height, breaking):
5      low, high = 1, height
6      while low < high:
7          mid = (low + high) // 2
8          print(low, high, mid)
9          if do_experiment(mid, breaking):
10             high = mid
11         else:
12             low = mid + 1
13     return low - 1
```

Line 8

- ພິມຄ່າ low, high ແລະ mid ໃນ loop ເພື່ອນັບຈຳນວນການເຮັດວຽກຊ້ຳຂອງຂອງ binary search.

1. ບັນຫາການຕົກຂອງໄຂ່

1.3. ປະສິດທິພາບການປະເມີນ

ຖ້າຊັ້ນທີ 50 ແມ່ນຊັ້ນທຳອິດທີ່ໄຂ່ແຕກ, ການຄົ້ນຫາແບບ binary search ຈະດຳເນີນການ 6 ຄັ້ງ.

```
1 height = int(input("Input the number of floors: "))
2 breaking = int(input("Input the first breaking floor: "))
3 floor = find_highest_safe_floor2(height, breaking)
4 print(f"Your egg will safe till the {floor}-th floor.")
```

```
Input the number of floors: 100
Input the first breaking floor: 50
1 100 50
1 50 25
26 50 38
39 50 44
45 50 47
48 50 49
Your egg will safe till the 49-th floor.
```

1. ปัญหาการตกของไข่

1.3. ประสิทธิภาพการประเมิน

ถ้าชั้นที่ 51 แม่นชั้นทำอิดที่ไข่แตก, การค้นหาแบบ binary search จะดำเนินการ 7 ครั้ง.

```
1 height = int(input("Input the number of floors: "))
2 breaking = int(input("Input the first breaking floor: "))
3 floor = find_highest_safe_floor2(height, breaking)
4 print(f"Your egg will safe till the {floor}-th floor.")
```

```
Input the number of floors: 100
Input the first breaking floor: 51
1 100 50
51 100 75
51 75 63
51 63 57
51 57 54
51 54 52
51 52 51
Your egg will safe till the 50-th floor.
```

Pop quiz

Quiz. #1

ต่ไปนี้แม่น code สำลับ สำหับเกม 'ต๊อเลกถาดกาม'. ถ้าถ่าสูงสุด = 100 และ ต๊อเลกถาดกาม = 51, จะพิมจำนอมเท๊าใด?

```
from random import randint

maximum = int(input("Enter the number of maximum: "))
number = int(input("Enter your guessing number: "))
count = 0
low, high = 1, maximum
while low < high:
    mid = (low + high) // 2
    count += 1
    if mid == number:
        print(f"Your number is {number}.")
        break
    elif mid > number:
        high = mid - 1
    else:
        low = mid + 1
print(f"Total {count} times are searched.")
```

Quiz. #2

ຖ້າຄ່າສູງສຸດ = 100 ແລະ ຕົວເລກຄາດການ = 25 ໃນເກມ 'ຕົວເລກຄາການ', ແລ້ວຈຳນວນຕົວເລກຈະຖືກພິມອອກເທົ່າໃດ?

Pair programming

Pair Programming Practice

ແນວທາງ, ກົນໄກ ແລະ ແຜນສຸກເສີນ

ການກະກຽມການສ້າງໂປຣແກຣມຮ່ວມກັນເປັນຄູ່ກ່ຽວຂ້ອງກັບການໃຫ້ຄຳແນະນຳແລະກົນໄກເພື່ອຊ່ວຍໃຫ້ນັກຮຽນຈັບຄູ່ຢ່າງຖືກຕ້ອງແລະ ໃຫ້ເຂົາເຈົ້າເຮັດວຽກເປັນຄູ່. ຕົວຢ່າງ, ນັກຮຽນຄວນປ່ຽນ "ເຮັດ." ການກະກຽມທີ່ມີປະສິດຕິຜົນຕ້ອງໃຫ້ມີແຜນການສຸກເສີນໃນກໍລະນີທີ່ຄູ່ຮ່ວມງານຫນຶ່ງບໍ່ຢູ່ຫຼືຕັດສິນໃຈທີ່ຈະບໍ່ເຂົ້າຮ່ວມດ້ວຍເຫດຜົນໃດຫນຶ່ງ ຫຼືດ້ວຍເຫດຜົນອື່ນ. ໃນກໍລະນີເຫຼົ່ານີ້, ມັນເປັນສິ່ງສໍາຄັນທີ່ຈະເຮັດໃຫ້ມັນຊັດເຈນວ່ານັກຮຽນທີ່ມີປະຕິບັດໜ້າທີ່ຢ່າງຫ້າວຫັນຈະບໍ່ຖືກລົງໂທດ ຍ້ອນວ່າການຈັບຄູ່ບໍ່ໄດ້ຜົນດີ.

ການຈັບຄູ່ທີ່ຄ້າຍຄືກັນ, ບໍ່ຈຳເປັນຕ້ອງເທົ່າທຽມກັນ, ຄວາມສາມາດເປັນຄູ່ຮ່ວມງານ

ການຂຽນໂປຣແກຣມຄູ່ຈະມີປະສິດທິພາບເມື່ອນັກຮຽນຕັ້ງໃຈຮ່ວມກັນເຮັດວຽກ, ຊຶ່ງວ່າບໍ່ຈຳເປັນຕ້ອງມີຄວາມຮູ້ເທົ່າທຽມກັນ, ແຕ່ຕ້ອງມີຄວາມສາມາດເຮັດວຽກເປັນຄູ່ຮ່ວມງານ. ການຈັບຄູ່ນັກຮຽນທີ່ບໍ່ສາມາດເຂົ້າກັນໄດ້ມັກຈະເຮັດໃຫ້ການມີສ່ວນຮ່ວມທີ່ບໍ່ສົມດຸນກັນ. ຄູສອນຕ້ອງເນັ້ນຫນັກວ່າການຂຽນໂປຣແກຣມຄູ່ບໍ່ແມ່ນຍຸດທະສາດ “divide-and-conque”, ແຕ່ຈະເປັນຄວາມພະຍາຍາມຮ່ວມມືເຮັດວຽກທີ່ແທ້ຈິງໃນທຸກໆດ້ານສໍາລັບໂຄງການທັງຫມົດ. ຄູຄວນຫຼີກເວັ້ນການຈັບຄູ່ນັກຮຽນທີ່ອ່ອນຫຼາຍກັບນັກຮຽນທີ່ເກັ່ງຫຼາຍ.

ກະຕຸ້ນນັກຮຽນໂດຍການໃຫ້ສິ່ງຈູງໃຈພິເສດ

ການສະເໜີແຮງຈູງໃຈພິເສດສາມາດຊ່ວຍກະຕຸ້ນນັກຮຽນໃຫ້ຈັບຄູ່, ໂດຍສະເພາະກັບນັກຮຽນທີ່ມີຄວາມສາມາດຫຼາຍ. ຈະເຫັນວ່າມັນເປັນປະໂຫຍດທີ່ຈະໃຫ້ນັກຮຽນຈັບຄູ່ເຮັດວຽກຮ່ວມກັນພຽງແຕ່ຫນຶ່ງຫຼືສອງວຽກເທົ່ານັ້ນ.

Pair Programming Practice

ປ້ອງກັນການໂກງໃນການເຮັດວຽກຮ່ວມກັນ

ສິ່ງທ້າທາຍສຳລັບຄູແມ່ນເພື່ອຊອກຫາວິທີທີ່ຈະປະເມີນຜົນໄດ້ຮັບຂອງບຸກຄົນ, ໃນຂະນະທີ່ນຳໃຊ້ຜົນປະໂຫຍດຂອງການຮ່ວມມື. ຈະຮູ້ໄດ້ແນວໃດວ່ານັກຮຽນຕັ້ງໃຈເຮັດວຽກ ຫຼື ກົ້ງແຮງງານຜູ້ຮ່ວມງານ? ຜູ້ຊ່ຽວຊານແນະນຳໃຫ້ທົບທວນຄືນການອອກແບບຫຼັກສູດ ແລະ ການປະເມີນ ພ້ອມທັງປຶກສາຫາລືຢ່າງຈະແຈ້ງ ແລະ ຊັດເຈນກ່ຽວກັບພຶດຕິກຳຂອງນັກຮຽນທີ່ຈະຖືກຕີຄວາມວ່າຂີ້ຕົວະ. ຜູ້ຊ່ຽວຊານເນັ້ນໜັກໃຫ້ຄູເຮັດການມອບໝາຍໃຫ້ມີຄວາມໝາຍຕໍ່ນັກຮຽນ ແລະ ອະທິບາຍຄຸນຄ່າຂອງສິ່ງທີ່ນັກຮຽນຈະຮຽນຮູ້ໂດຍການເຮັດສຳເລັດ.

ສະພາບແວດລ້ອມການຮຽນຮູ້ໃນການຮ່ວມມື

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

Q1. ຖ້ານຳນົດໃຫ້ list ' nums' ທີ່ລຽງລຳດັບແລ້ວ ແລະ ຈຳນວນຖ້ວນ x ໃດໜຶ່ງ, ສ້າງຟັງຊັນສົ່ງຄືນທີ່ຢູ່ຂອງດັດຊະນີທີ່ x ຈະຖືກເອົາລົງໄປເກັບ. ຢ່າງໃດກໍຕາມ, S ຄວນເປັນ list ທີ່ລຽງບຳດັບເຖິງແມ່ນວ່າຫຼັງຈາກເອົາ x ລົງໄປເກັບ.

```python
nums = [10, 20, 40, 50, 60, 80]
x = int(input("Input a number to insert: "))
pos = search_insert_position(nums, x)
print(f"{x} should be inserted at position {pos}.")
nums.insert(pos, x)
print(nums)
```

```
Input a number to insert: 30
30 should be inserted at position 2.
[10, 20, 30, 40, 50, 60, 80]
```