SAMSUNG

Samsung Innovation Campus

Coding, Programming & Data Science

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 6.

Algorithm 3
– Problem Solving with Algorithms

Coding, Programming & Data Science

ຄຳອະທິບາຍບົດ

Chapter objectives

- ✓ ນັກຮຽນຈະສາມາດແກ້ໄຂບັນຫາຕ່າງໆໂດຍໃຊ້ຂັ້ນຕອນວິທີ. ນັກຮຽນຈະສາມາດເຂົ້າໃຈເຕັກນິກການອອກແບບຂັ້ນຕອນວິທີ ເຊັ່ນ: ວິທີການຂອງ greedy, ວິທີການ divide-and-conquer, ວິທີການ dynamic programming ແລະ backtracking ແລະ ສາມາດໃຊ້ພວກມັນເພື່ອແກ້ໄຂບັນຫາໃນຊີວິດຈິງ.

Chapter contents

- ✓ ຫົວຂໍ້ທີ 30. Greedy Approach
- ✓ ຫົວຂໍ້ທີ 31. Divide-and-Conquer
- ✓ ຫົວຂໍ້ທີ 32. Dynamic Programming
- ✓ ຫົວຂໍ້ທີ 33. Backtracking

Unit 30.

Greedy Approach

Learning objectives

- ✓ ນັກຮຽນສາມາດເຂົ້າໃຈວິທີການຂອງ greedy ແລະ ແກ້ໄຂບັນຫາທີ່ໃຫ້ໄດ້ໂດຍໃຊ້ວິທີການ greedy.
- ✓ ນັກຮຽນສາມາດເຂົ້າໃຈວ່າ ບັນຫາເຊັ່ນ: ການແລກປ່ຽນຫຼຽນ ແລະ ເລືອກກິດຈະກຳສາມາດແກ້ໄຂໄດ້ໂດຍໃຊ້ວິທີການຂອງ greedy.
- ✓ ນັກຮຽນສາມາດອອກແບບ ແລະ ສ້າງຂັ້ນຕອນວິທີ greedy ແບບຊ້ຳໆ.

Learning overview

- ✓ ຮຽນຮູ້ວິທີແກ້ໄຂບັນຫາກ່ຽວກັບຂັ້ນຕອນວິທີໂດຍນຳໃຊ້ວິທີການຂອງ greedy.
- ✓ ຮຽນຮູ້ວິທີການແກ້ໄຂບັນຫາການແລກປ່ຽນຫຼຽນໂດຍໃຊ້ວິທີການຂອງ greedy.
- ✓ ຮຽນຮູ້ວິທີການແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳໂດຍໃຊ້ວິທີການຂອງ greedy.

Concepts you will need to know from previous units

- ✓ ສາມາດເຂົ້າໃຈ ແລະ ສ້າງຄຳນິຍາມຂອງຟັງຊັນ recursive ແລະ ເງື່ອນໄຂທີ່ຈະອອກຈາກ recursive.
- ✓ ສາມາດໃຊ້ຂັ້ນຕອນວິທີການຈັດລຽງລຳດັບເພື່ອຈັດລຽງຊຸດຂໍ້ມູນທີ່ກຳນົດໄວ້.
- ✓ ສາມາດວິເຄາະປະສິດທິພາບຂອງຂັ້ນຕອນວິທີ recursive.

Keywords

- Greedy Algorithm
- Activity Selection Problem
- Coin Change Problem
- Recursion
- Iteration

Mission

1. Real world problem

1.1. ຈຳນວນການຈັດປະຊຸມທີ່ເປັນໄດ້ທັງໝົດມີເທົ່າໃດ?

- ສົມມຸດວ່າມີບໍລິສັດທີ່ມີຫ້ອງປະຊຸມພຽງແຕ່ຫ້ອງດຽວ.
- ໃນບໍລິສັດມີຫຼາຍທີມງານ ແລະ ເຂົາເຈົ້າກຳລັງວາງແຜນທີ່ຈະຈັດກອງປະຊຸມຫຼາຍໆຄັ້ງຢູ່ໃນຫ້ອງປະຊຸມນີ້.
- ເນື່ອງຈາກທັງສອງທີມບໍ່ສາມາດໃຊ້ຫ້ອງປະຊຸມດັ່ງກ່າວໃນເວລາດຽວກັນໄດ້, ຜູ້ຈັດການຫ້ອງປະຊຸມພະຍາຍາມຈັດສັນຫ້ອງປະຊຸມເພື່ອໃຫ້ມີຈຳນວນກອງປະຊຸມສູງສຸດພາຍໃຕ້ເງື່ອນໄຂທີ່ວ່າ ກອງປະຊຸມຕ້ອງບໍ່ຊ້ຳກັນ.
- ມັນເປັນໄປໄດ້ບໍທີ່ຈະຮູ້ຈຳນວນສູງສຸດຂອງກອງປະຊຸມທີ່ເປັນໄປໄດ້ເມື່ອພວກເຮົາຮູ້ເວລາເລີ່ມຕົ້ນ ແລະ ເວລາສິ້ນສຸດສຳຫຼັບ N ຈຳນວນຄຳຮ້ອງຂໍການນຳໃຊ້ຫ້ອງປະຊຸມ?
- ບັນຫານີ້ເອີ້ນວ່າບັນຫາການເລືອກກິດຈະກຳ.
- ມາສ້າງຂັ້ນຕອນວິທີເພື່ອແກ້ໄຂບັນຫາການຈັດສັນຫ້ອງປະຊຸມ.

2. Mission

2.1. ບັນຫາການເລືອກກິດຈະກຳ

- ສົມມຸດວ່າ N=7 ເປັນຄຳຮ້ອງຂໍນຳໃຊ້ຫ້ອງປະຊຸມໄດ້ຖືກມອບໃຫ້ດັ່ງຕໍ່ໄປນີ້.
- ການຮ້ອງຂໍຫ້ອງປະຊຸມແຕ່ລະຄັ້ງແມ່ນໃຫ້ເວລາເລີ່ມຕົ້ນ ແລະ ເວລາສິ້ນສຸດ.
- ຈຳນວນສູງສຸດຂອງກອງປະຊຸມທີ່ບໍ່ຊ້ຳກັນແມ່ນເທົ່າໃດ?

id	0	1	2	3	4	5	6
start	1	3	2	1	5	8	5
finish	2	4	5	6	6	9	9

3. ການແກ້ໄຂບັນຫາ

3.1. ວິທີການທີ່ການເລືອກກິດຈະກຳ

ການເລືອກການປະຊຸມສຸດທ້າຍແມ່ນ [0, 1, 4, 5] ແລະ ຈຳນວນການປະຊຸມທີ່ເລືອກໄດ້ສູງສຸດແມ່ນ 4.

```
start = [1, 3, 2, 0, 5, 8, 5]
finish = [2, 4, 5, 6, 6, 9, 9]
meetings = activity_selection1(start, finish)
maximum = len(meetings)
print(meetings, maximum)
```

```
[0, 1, 4, 5] 4
```

3. ການແກ້ໄຂບັນຫາ

3.2. Code ສຸດທ້າຍຂອງການເລືອກກິດຈະກຳ

```
1  def activity_selection1(start, finish):
2      result = []
3      i = 0
4      result.append(i)
5      for j in range(1, len(start)):
6          if finish[i] <= start[j]:
7              result.append(j)
8              i = j
9      return result
```

Key concept

1. ວິທີການ Greedy

1.1. ວິທີການ Greedy ແມ່ນຫຍັງ?

- ວິທີການຂອງ greedy ແມ່ນເຕັກນິກການອອກແບບຂັ້ນຕອນວິທີທີ່ມີຄໍາຕອບໂດຍການເລືອກທາງເລືອກ "ທີ່ດີທີ່ສຸດ" ໃນປັດຈຸບັນຈາກລໍາດັບຂອງທາງເລືອກຕາມເງື່ອນໄຂທີ່ກໍານົດໄວ້ກ່ອນໜ້ານີ້.
- ມັນເປັນເຕັກນິກການອອກແບບຂັ້ນຕອນວິທີທີ່ມີປະສິດທິພາບ ແລະ ງ່າຍທີ່ສຸດສໍາລັບການແກ້ໄຂບັນຫາ ການເພີ່ມປະສິດທິພາບ ເຊັ່ນ: ບັນຫາການແລກປ່ຽນແປງຫຼຽນ ແລະ ບັນຫາການເລືອກກິດຈະກໍາ.

Focus ວິທີການ greedy ແກ້ໄຂບັນຫາເປັນສາມຂັ້ນຕອນຕໍ່ໄປນີ້.

- ຂະບວນການຄັດເລືອກ: ເລືອກອົງປະກອບຕໍ່ໄປເພື່ອເພີ່ມໃສ່ຊຸດຂໍ້ມູນ.
- ການກວດສອບຄວາມເປັນໄປໄດ້: ກວດເບິ່ງວ່າຂໍ້ມູນຊຸດໃຫມ່ແມ່ນເຫມາະສົມທີ່ຈະເປັນຄໍາຕອບ.
- ການກວດສອບການແກ້ໄຂ: ກໍານົດວ່າຂໍ້ມູນຊຸດໃຫມ່ແມ່ນຄໍາຕອບຂອງບັນຫາ.

1. ວິທີການ Greedy

1.2. ບັນຫາການແລກປ່ຽນຫຼຽນ

- ການນຳໃຊ້ວິທີການ greedy, ໃຫ້ແກ້ໄຂບັນຫາການແລກປ່ຽນຫຼຽນດັ່ງຕໍ່ໄປນີ້.
  - ມີຫຼຽນສີ່ປະເພດໃນເກົາຫຼີ, ດັ່ງທີ່ສະແດງຂ້າງລຸ່ມນີ້.
  - ຈຳນວນເງິນຂອງແຕ່ລະຫຼຽນແມ່ນ 500 ວອນ, 100 ວອນ, 50 ວອນ, ແລະ 10 ວອນ.
- ເມື່ອແຈກຢາຍການແລກປ່ຽນຫຼຽນ ພວກເຮົາພະຍາຍາມໃຫ้ຈຳນວນຫຼຽນໜ้ອຍທີ່ສຸດ.
  - ໃຫ้ສົມມຸດວ່າຈຳນວນຫຼຽນແມ่นพรึงพໍ่ต่ำการแลกปรึง.

1. ວິທີການ Greedy

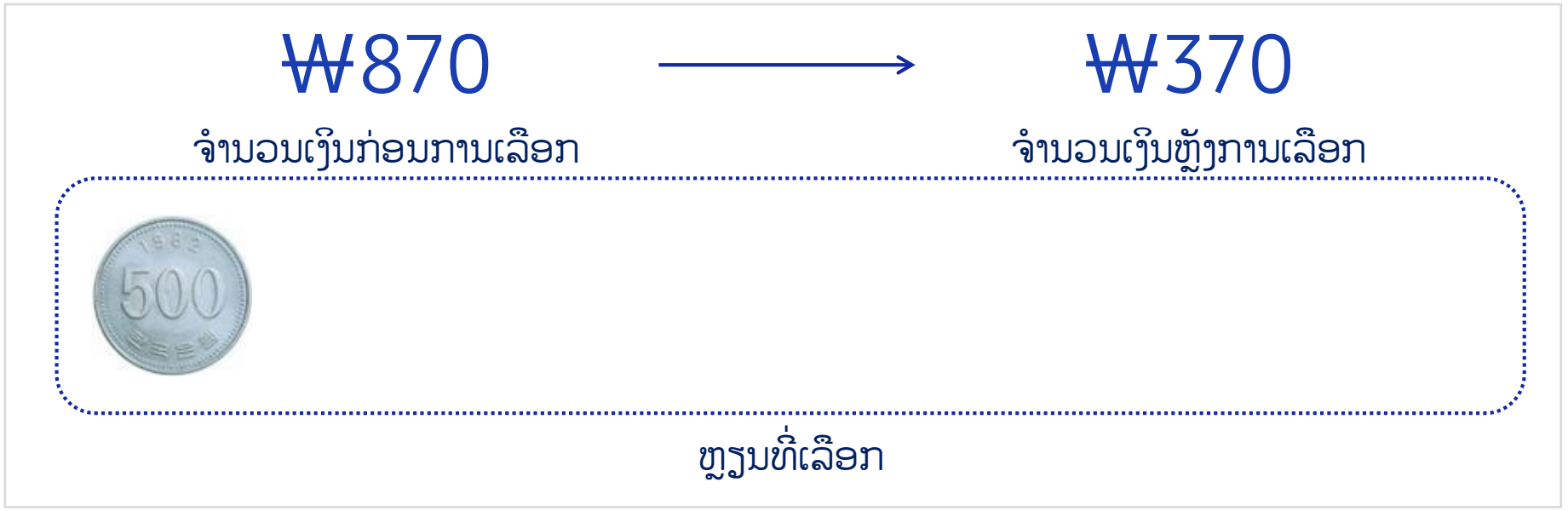
1.2. ບັນຫາການແລກປ່ຽນຫຼຽນ

- ຕົວຢ່າງ, ສົມມຸດວ່າການແລກປ່ຽນທີ່ຕ້ອງການສົ່ງຄືນແມ່ນ 870 ວອນ.
- ທ່ານສາມາດເຮັດການແລກປ່ຽນດ້ວຍຫຼຽນຈໍາ 87 ຫຼຽນ ໂດຍໃຊ້ຫຼຽນມູນຄ່າ 10 ວອນ. ຢ່າງໃດກໍ່ຕາມ, ໃນກໍລະນີນີ້, ມີຈໍານວນຫຼຽນຫຼາຍເກີນໄປສໍາລັບການປ່ຽນແປງ.
- ຈະສາມາດແລກປ່ຽນຫຼຽນມຸນຄ່າ 870 ວອນ ໂດຍໃຊ້ຈໍານວນຫຼຽນທີ່ໜ້ອຍທີ່ສຸດ ດັ່ງທີ່ສະແດງຂ້າງລຸ່ມນີ້?

1. ວິທີການ Greedy

1.2. ບັນຫາການແລກປ່ຽນຫຼຽນ

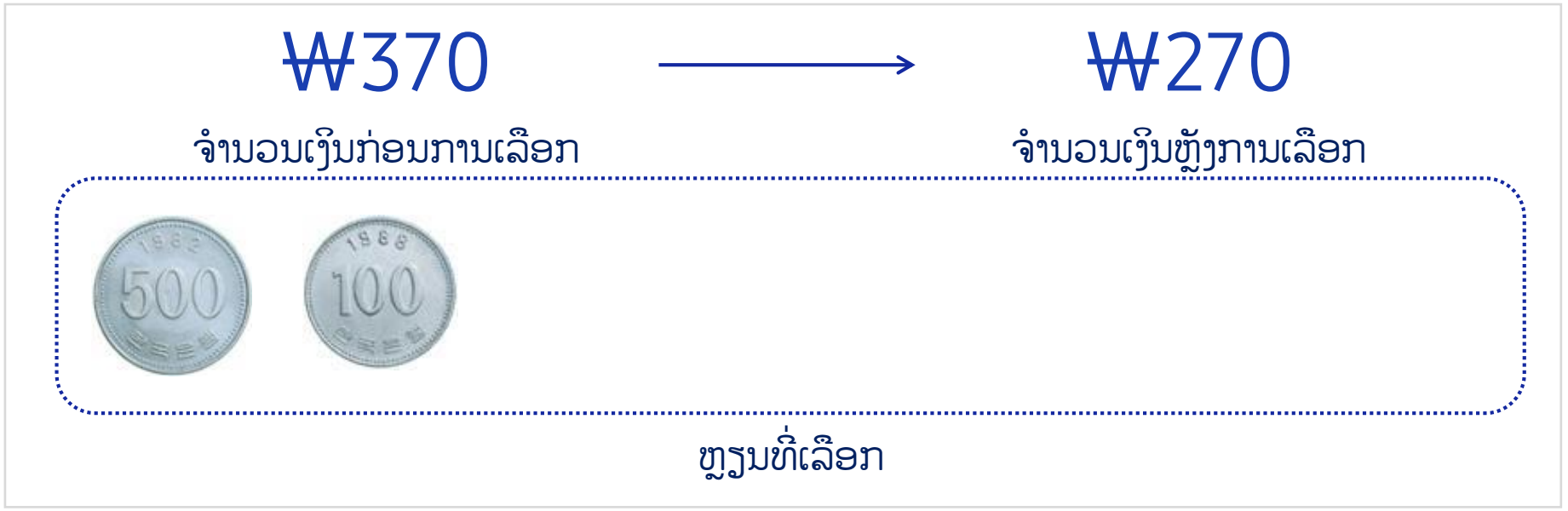
- ກ່ອນອື່ນໝົດ, ໃຫ້ເຮົາເລືອກ 500 ວອນ, ເປັນຫຼຽນທີ່ໃຫຍ່ທີ່ສຸດ. ຈຳນວນການແລກປ່ຽນທີ່ຍັງເຫຼືອ ແມ່ນ 370 ວອນ.
- ດັ່ງນັ້ນ, ຫຼຽນ 500 ວອນບໍ່ສາມາດເລືອກໄດ້ອີກ.



1. ວິທີການ Greedy

1.2. ບັນຫາການແລກປ່ຽນຫຼຽນ

- ເນື່ອງຈາກຫຼຽນ 500 ວອນບໍ່ສາມາດເລືອກໄດ້, ຫຼຽນທີ່ໃຫຍ່ທີ່ສຸດຕໍ່ໄປແມ່ນ 100 ວອນ, ຈະຖືກເລືອກ.
- ການແລກປ່ຽນທີ່ຍັງເຫຼືອແມ່ນ 270 ວອນ, ດັ່ງນັ້ນທ່ານຍັງສາມາດເລືອກອີກ 100 ວອນ.



1. ວິທີການ Greedy

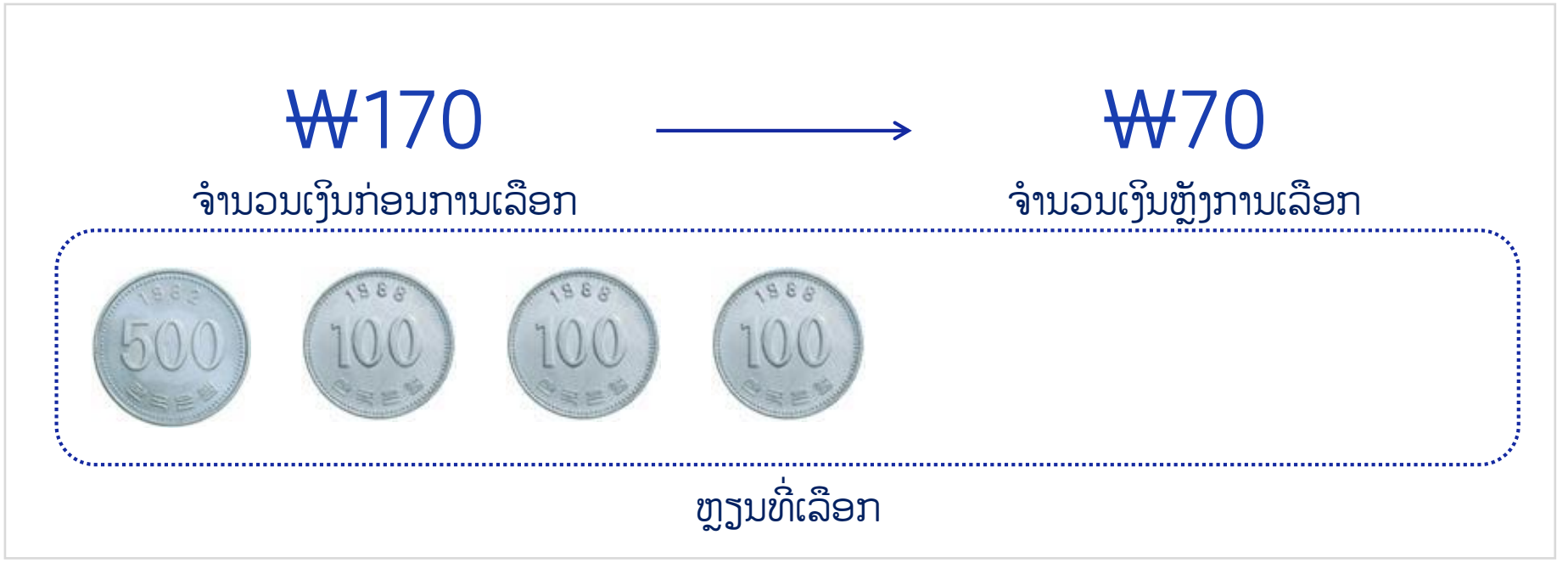
1.2. ບັນຫາການແລກປ່ຽນຫຼຽນ

- ຖ້າທ່ານເລືອກຫຼຽນ 100 ວອນ, ການແລກປ່ຽນທີ່ຍັງເຫຼືອແມ່ນ 170 ວອນ, ດັ່ງນັ້ນທ່ານຍັງສາມາດເລືອກຫຼຽນ 100 ວອນອີກ.

1. ວິທີການ Greedy

1.2. ບັນຫາການແລກປ່ຽນຫຼຽນ

- ຖ້າທ່ານເລືອກຫຼຽນ 100 ວອນ, ການແລກປ່ຽນທີ່ຍັງເຫຼືອຈະເປັນ 70 ວອນ.
- ຕອນນີ້, ຫຼຽນ 100 ວອນບໍ່ສາມາດເລືອກໄດ້ອີກຕໍ່ໄປ.



1. ວິທີການ Greedy

1.2. ບັນຫາການແລກປ່ຽນຫຼຽນ

- ເນື່ອງຈາກທ່ານບໍ່ສາມາດເລືອກຫຼຽນ 100 ວອນໄດ້ອີກ, ໃຫ້ເລືອກເອົາ 50 ວອນ, ເຊິ່ງເປັນຫຼຽນທີ່ໃຫຍ່ທີ່ສຸດຕໍ່ໄປ.
- ການແລກປ່ຽນທີ່ຍັງເຫຼືອແມ່ນ 20 ວອນ ແລະ ຫຼຽນ 50 ວອນແມ່ນບໍ່ແລກປ່ຽນໄດ້ອີກແລ້ວ.

1. ວິທີການ Greedy

1.2. ບັນຫາການແລກປ່ຽນຫຼຽນ

- ຫຼຽນທີ່ໃຫຍ່ທີ່ສຸດຕໍ່ໄປຫຼັງຈາກຫຼຽນ 50 ວອນແມ່ນຫຼຽນ 10 ວອນ, ສະນັ້ນໃຫ້ພວກເຮົາເລືອກ 10 ວອນ.
- ການແລກປ່ຽນທີ່ຍັງເຫຼືອແມ່ນ 10 ວອນ, ດັ່ງນັ້ນທ່ານຍັງສາມາດເລືອກອີກ 10 ວອນ.

1. ວິທີການ Greedy

1.2. ບັນຫາການແລກປ່ຽນຫຼຽນ

- ຖ້າທ່ານເລືອກຫຼຽນ 10 ວອນ, ການປ່ຽນແປງທີ່ຍັງເຫຼືອຈະເປັນ 0 ວອນ.
- ດັ່ງນັ້ນ, ຫຼຽນທີ່ເລືອກທັງໝົດມີມູນຄ່າ 870 ວອນ ແລະ ການແລກປ່ຽນສາມາດສົ່ງຄືນຈຳນວນຫຼຽນ 7 ຫຼຽນ.

1. ວິທີການ Greedy

1.3. ອອກແບບ Greedy Algorithm

ຂະບວນການຂ້າງເທິງນີ້ແມ່ນສະແດງອອກໃນ pseudocode ດັ່ງຕໍ່ໄປນີ້.

```
while (there are more coins and the problem is not solved) {
   grab the largest remaining coin;
   if (adding the coin doesn't make the change exceed the amount owed)
       add the coin to the change;
   if (the total value of the change equals the amount owed)
       the problem is solved;
}
```

1. ວິທີການ Greedy

1.3. ອອກແບບ Greedy Algorithm

ວິທີການ greedy ທີ່ໃຊ້ໃນການແກ້ໄຂບັນຫາການແລກປ່ຽນຫຼຽນປະກອບດ້ວຍສາມຂັ້ນຕອນດັ່ງຕໍ່ໄປນີ້.

- ຂັ້ນຕອນການຄັດເລືອກ:
  - ເລືອກຫຼຽນທີ່ມີຄ່າເງິນສູງສຸດ.
  - ໃນຂັ້ນຕອນນີ້, ອົງປະກອບທີ່ດີທີ່ສຸດທີ່ຈະຖືກເພີ່ມເຂົ້າໃນຊຸດຂໍ້ມູນການແກ້ໄຂບັນຫາໄດ້ຖືກເລືອກ.
- ການກວດສອບຄວາມເປັນໄປໄດ້:
  - ກວດເບິ່ງວ່າມູນຄ່າເງິນຂອງຫຼຽນທີ່ເລືອກແມ່ນໃຫຍ່ກວ່າຈຳນວນການແລກປ່ຽນທັງຫມົດ.
  - ເວົ້າອີກແບບໜຶ່ງວ່າ, ເປັນການກວດເບິ່ງວ່າອົງປະກອບທີ່ດີທີ່ສຸດທີ່ເລືອກຢູ່ໃນຂັ້ນຕອນປະຈຸບັນແມ່ນເຫມາະສົມທີ່ຈະເປັນຄຳຕອບ.
- ການກວດສອບການແກ້ໄຂ:
  - ກວດເບິ່ງວ່າຈຳນວນຫຼຽນທີ່ເລືອກແມ່ນເທົ່າກັບຈຳນວນການແລກປ່ຽນ.
  - ເວົ້າອີກຢ່າງໜຶ່ງວ່າ, ເປັນການເພີ່ມອົງປະກອບທີ່ດີທີ່ສຸດທີ່ເລືອກຢູ່ໃນຂັ້ນຕອນປະຈຸບັນແມ່ນການກວດສອບເພື່ອເບິ່ງວ່າມັນເປັນຄຳຕອບ.

1. ວິທີການ Greedy

1.3. ອອກແບບ Greedy Algorithm

- ໃນບັນຫາການແລກປ່ຽນຫຼຽນ, ຂະບວນການຂອງການຄັດເລືອກເອົາຫຼຽນສອດຄ່ອງກັບຂັ້ນຕອນການຄັດເລືອກ.

```
while (there are more coins and the problem is not solved) {
  grab the largest remaining coin;                          selection procedure
   if (adding the coin doesn't make the change exceed the amount owed)
       add the coin to the change;
   if (the total value of the change equals the amount owed)
       the problem is solved;
}
```

1. ວິທີການ Greedy

1.3. ອອກແບບ Greedy Algorithm

- ຂະບວນການກວດສອບວ່າຫຼຽນທີ່ເລືອກບໍ່ເກີນຈຳນວນທີ່ຍັງເຫຼືອຂອງການແລກປ່ຽນ ເປັນຂັ້ນຕອນການກວດສອບຄວາມເປັນໄປໄດ້.

```
while (there are more coins and the problem is not solved) {
   grab the largest remaining coin;
   if (adding the coin doesn't make the change exceed the amount owed)      feasibility check
        add the coin to the change;
   if (the total value of the change equals the amount owed)
        the problem is solved;
}
```

1. ວິທີການ Greedy

1.3. ອອກແບບ Greedy Algorithm

ຂະບວນການກວດສອບວ່າຊຸດຫຼຽນທີ່ເລືອກໃນປັດຈຸບັນເທົ່າກັບຈຳນວນການແລກປ່ຽນທັງໝົດເປັນ ຂອງຂະບວນການກວດສອບການແກ້ໄຂ.

```
while (there are more coins and the problem is not solved) {
    grab the largest remaining coin;
    if (adding the coin doesn't make the change exceed the amount owed)
        add the coin to the change;
    if (the total value of the change equals the amount owed)
        the problem is solved;
}
```

solution check

Let’s code

1. ວິທີການ Greedy

1.1. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

- ໃຫ້ຂຽນຂັ້ນຕອນວິທີເພື່ອແກ້ໄຂບັນຫາການແລກປ່ຽນຫຼຽນ.
- ການປ້ອນຂໍ້ມູນຂອງຂັ້ນຕອນວິທີແມ່ນປະເພດຫຼຽນ ແລະ ຈຳນວນການແລກປ່ຽນ ແລະຜົນໄດ້ຮັບແມ່ນລາຍການຂອງຫຼຽນ.
- ໃນເວລານີ້, ສົມມຸດວ່າຈຳນວນເງິນຂອງຫຼຽນແມ່ນໄດ້ຮັບໃນລຳດັບຫຼຸດລົງມາ.

```
1  def coin_change(coins, amount):
2      changes = []
3      largest = 0
4      while amount > 0:
5          if amount < coins[largest]:
6              largest += 1
7          else:
8              changes.append(coins[largest])
9              amount -= coins[largest]
10     return changes
```

Line 2-3

- ບັນຊີລາຍຊື່ການແລກປ່ຽນ, ເຊິ່ງເຮັດຫນ້າທີ່ເປັນກະເປົ໋າເງິນຫຼຽນ, ໄດ້ຖືກກຳໜົດຄ່າເລີ່ມຕົ້ນເປັນ [ ].
- ເນື່ອງຈາກຫຼຽນຖືກຈັດລຽງຕາມລຳດັບຈາກໃຫຍ່ໄປຫານ້ອຍ, ເລີ່ມຕົ້ນຈຳນວນສູງສຸດຂອງຕຳແໜ່ງຫຼຽນ 'ໃຫຍ່ທີ່ສຸດ' ທີ່ສາມາດຖືກຈັດໃສ່ໃນກະເປົ໋າເງິນໃນປະຈຸບັນເປັນອົງປະກອບທຳອິດຂອງລາຍການຫຼຽນ.

1. ວິທີການ Greedy

1.1. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ເນື່ອງຈາກຈຳນວນຫຼຽນຖືກຈັດລຽງຕາມລຳດັບ, ວິທີການ greedy ສາມາດຖືກນຳໃຊ້ດັ່ງຕໍ່ໄປນີ້.

```
1  def coin_change(coins, amount):
2      changes = []
3      largest = 0
4      while amount > 0:
5          if amount < coins[largest]:
6              largest += 1
7          else:
8              changes.append(coins[largest])
9              amount -= coins[largest]
10     return changes
```

Line 4-9

- ປະໂຫຍກ while loop ທີ່ເຮັດຊ້ຳຄືນຈົນກ່ວາການແລກປ່ຽນຍັງສາມາດເຮັດໄດ້.
- ຖ້າການແລກປ່ຽນໃຫຍ່ກວ່າຫຼືເທົ່າກັບຈຳນວນ 'ໃຫຍ່ທີ່ສຸດ' ໃນປັດຈຸບັນ, ລົບຈຳນວນອອກ ຈາກ Line 8-9.
- ຖ້າການແລກປ່ຽນໜ້ອຍກວ່າຈຳນວນ 'ທີ່ໃຫຍ່ທີ່ສຸດ' ໃນປັດຈຸບັນ, ເພີ່ມຄ່າທີ່ໃຫຍ່ທີ່ສຸດ ແລະ ດຳເນີນການກັບຈຳນວນຫຼຽນຕໍ່ໄປ.
- ເມື່ອອອກຈາກ loop, ໃຫສົ່ງຄືນການແລກປ່ຽນເພາະວ່າຈຳນວນຫຼຽນໃນການແລກປ່ຽນແມ່ນຈຳນວນການແລກປ່ຽນ.

1. ວິທີການ Greedy

1.2. Code ທີ່ສັ່ງເຮັດວຽກໂປຣແກຣມ

- ການສ້າງຂັ້ນຕອນວິທີກ່ອນໜ້ານີ້ດຳເນີນການດັ່ງຕໍ່ໄປນີ້.
- ລາຍການ ແລະ ຈຳນວນຂອງຫຼຽນແມ່ນຜົນໄດ້ຮັບສຳລັບຈຳນວນການແລກປ່ຽນທີ່ໄດ້ຮັບຈາກຜູ້ໃຊ້.
- ໃນຂະນະນີ້, ສົມມຸດວ່າມູນຄ່າຂໍ້ມູນປ້ອນເຂົ້າແມ່ນຈຳນວນທີ່ສາມາດແລກປ່ຽນຈາກຊຸດຂອງຫຼຽນ.

```
1 coins = [500, 100, 50, 10]
2 amount = int(input("Input the amount: "))
3 changes = coin_change(coins, amount)
4 print(changes, len(changes))
```

```
Input the amount: 870
[500, 100, 100, 100, 50, 10, 10] 7
```

2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.1. ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

- ໃຫ້ໃຊ້ວິທີການ greedy ກັບບັນຫາການເລືອກກິດຈະກຳທີ່ໄດ້ກ່າວມາຂ້າງເທິງ.
- ສົມມຸດວ່າການຮ້ອງຂໍການນຳໃຊ້ຫ້ອງປະຊຸມ N=7 ໄດ້ຖືກຈັດສັນໃຫ້ດັ່ງຕໍ່ໄປນີ້.
- ການຮ້ອງຂໍໃຊ້ຫ້ອງປະຊຸມແຕ່ລະຄັ້ງແມ່ນໃຫ້ເວລາເລີ່ມຕົ້ນ ແລະ ເວລາສິ້ນສຸດ. ຈຳນວນສູງສຸດຂອງ ການປະຊຸມທີ່ບໍ່ຊ້ຳກັນໃນທີ່ນີ້ແມ່ນເທົ່າໃດ?

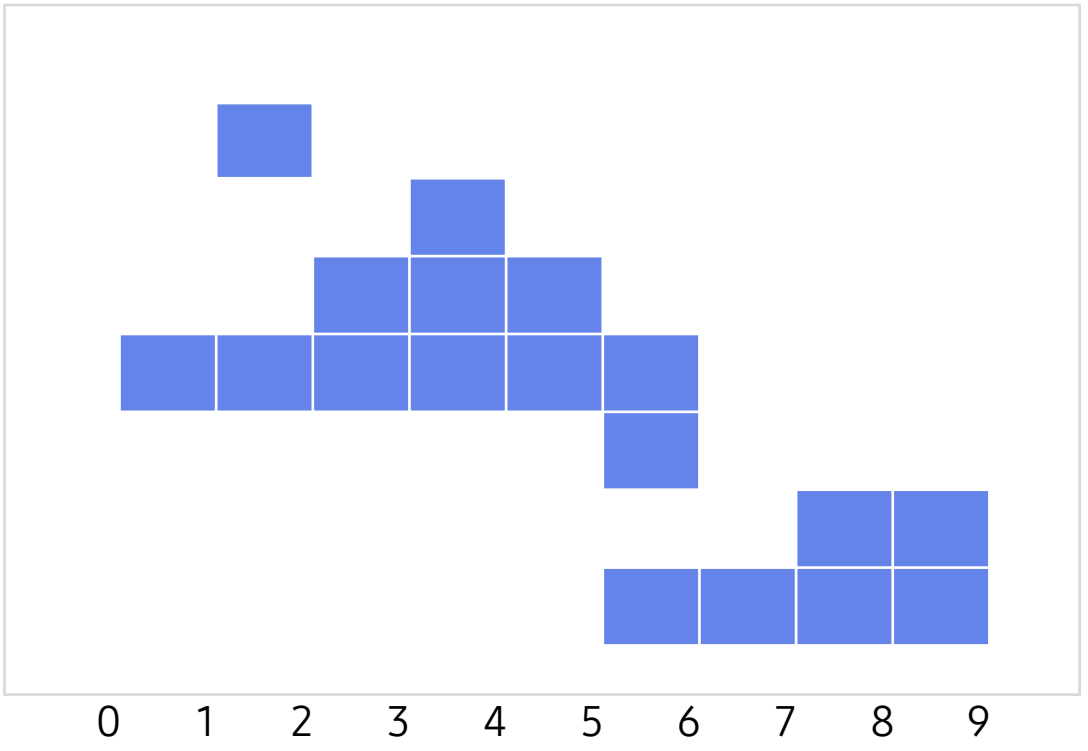
id	0	1	2	3	4	5	6
start	1	3	2	1	5	8	5
finish	2	4	5	6	6	9	9

2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.1. ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

ແຕ້ມຮູບຄຳຮ້ອງຂໍໃຊ້ຫ້ອງປະຊຸມເຫຼົ່ານີ້ຢູ່ໃນຕາຕະລາງເວລາມີດັ່ງນີ້.

id	start	finish
0	1	2
1	3	4
2	2	5
3	0	6
4	5	6
5	7	9
6	5	9

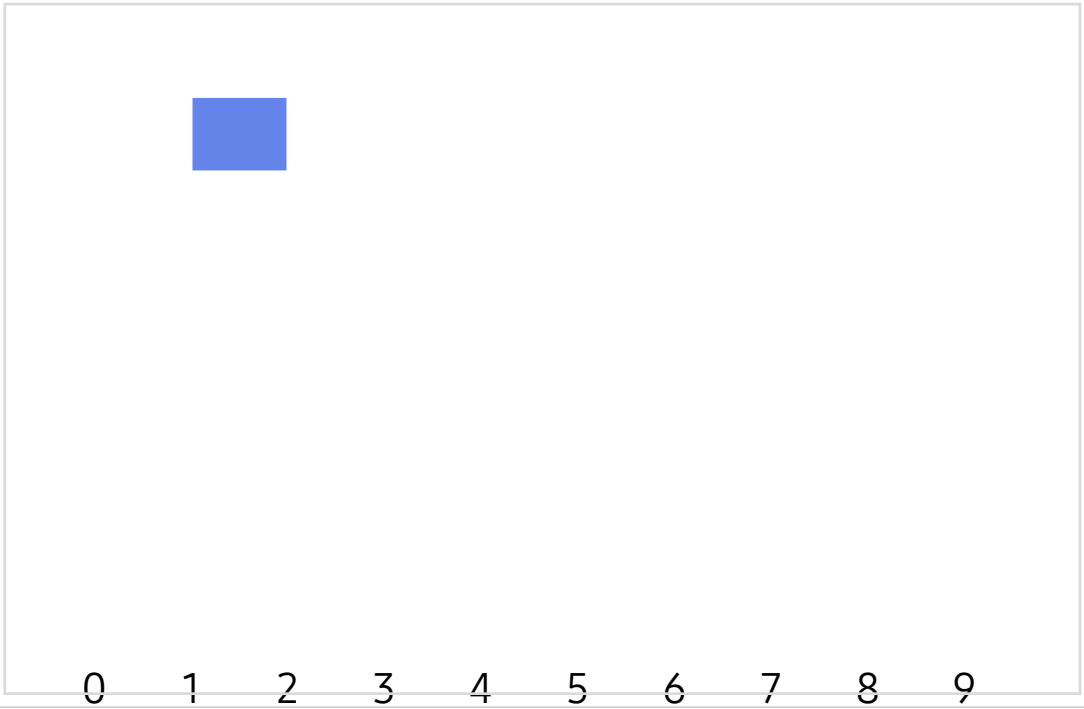


2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.1. ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

- ໃນຕອນທຳອິດ, ກອງປະຊຸມທັງຫມົດສາມາດຈັດຂຶ້ນໄດ້, ດັ່ງນັ້ນການປະຊຸມທີ 0 ສາມາດຈັດຂຶ້ນໄດ້.
- ໃນເວລານີ້, ໃຫ້ສັງເກດວ່າກອງປະຊຸມໄດ້ຖືກຈັດລຽງຕາມລຳດັບຂອງເວລາສິ້ນສຸດ. ເວົ້າອີກຢ່າງໜຶ່ງວ່າ, ກອງປະຊຸມຄັ້ງທີ 0 ແມ່ນໃຊ້ເວລາສັ້ນທີ່ສຸດ.

id	start	finish
0	1	2

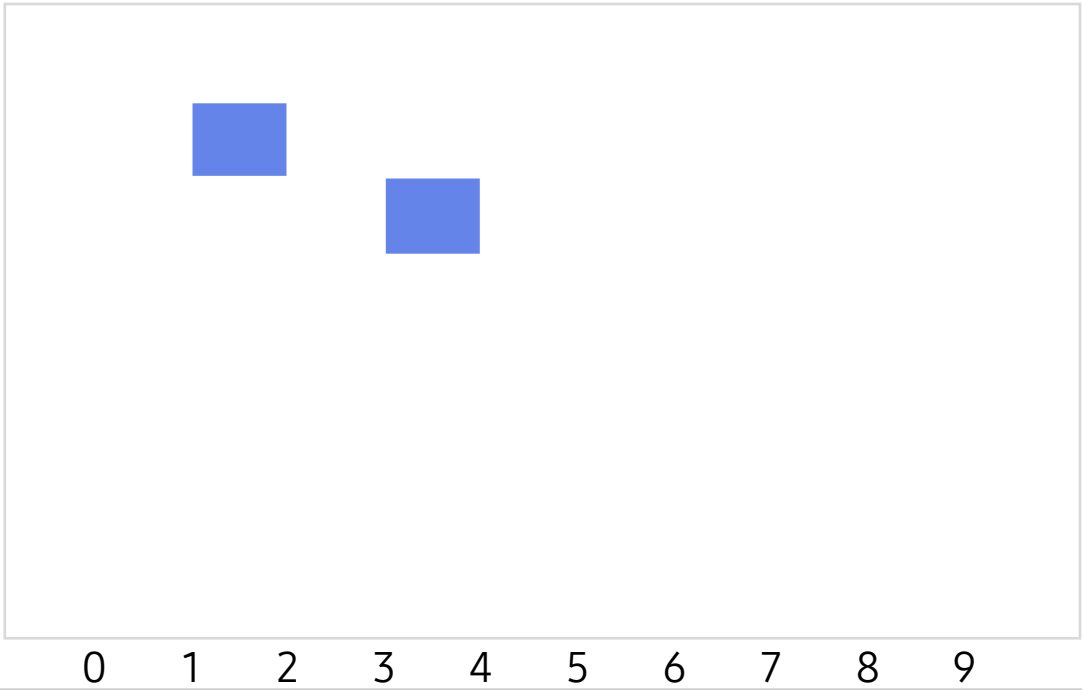


2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.1. ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

- ເວລາສິ້ນສຸດຂອງກອງປະຊຸມແມ່ນຊ້າກວ່າກອງປະຊຸມຄັ້ງທີ 0.
- ດັ່ງນັ້ນ, ຖ້າເວລາເລີ່ມຕົ້ນຢູ່ຫຼັງເວລາສຸດທ້າຍຂອງການປະຊຸມທີ່ເລືອກແລ້ວ, ແມ່ນສາມາດເລືອກການປະຊຸມນັ້ນໄດ້.
- ມັນສາມາດເລືອກໄດ້ເພາະວ່າເວລາເລີ່ມຕົ້ນ 3 ຂອງການປະຊຸມທີ 1 ແມ່ນຫຼາຍກວ່າເວລາສິ້ນສຸດ 2 ຂອງການປະຊຸມທີ 0.

id	start	finish
0	1	2
1	3	4

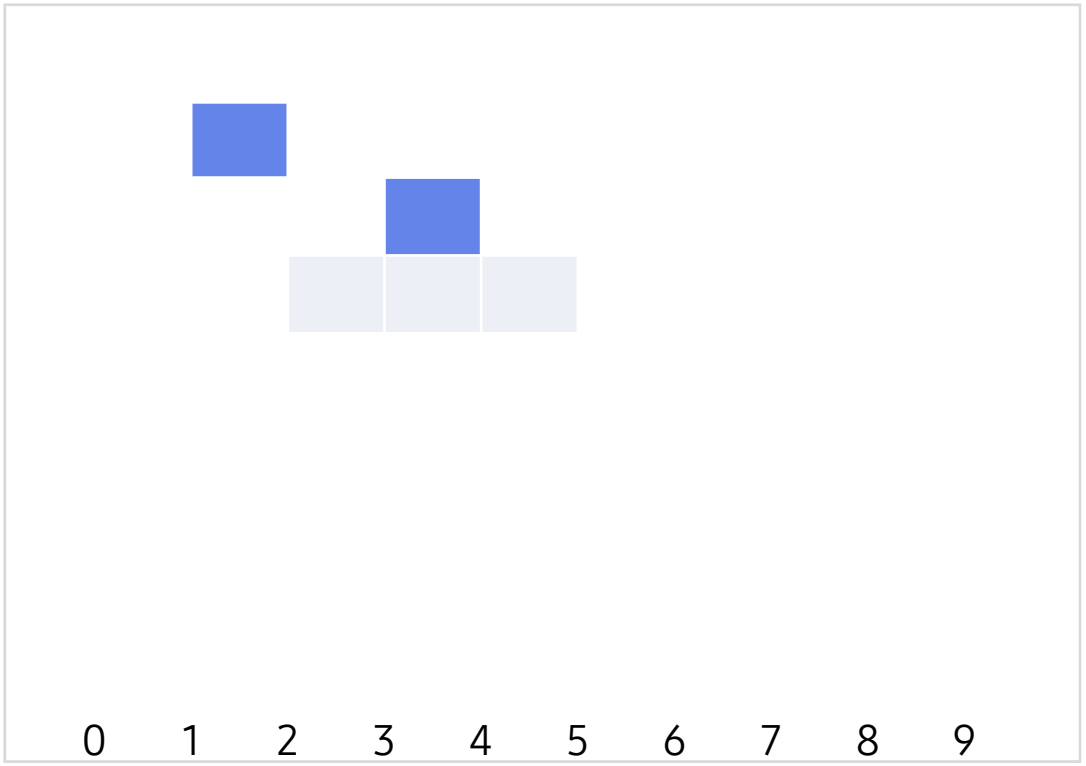


2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.1. ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

ເວລາເລີ່ມຕົ້ນຂອງກອງປະຊຸມຄັ້ງທີສອງແມ່ນຫນ້ອຍກວ່າເວລາສິ້ນສຸດຂອງກອງປະຊຸມຄັ້ງທີ 1, ດັ່ງນັ້ນຈຶ່ງບໍ່ສາມາດເລືອກໄດ້.

id	start	finish
0	1	2
1	3	4
2	2	5

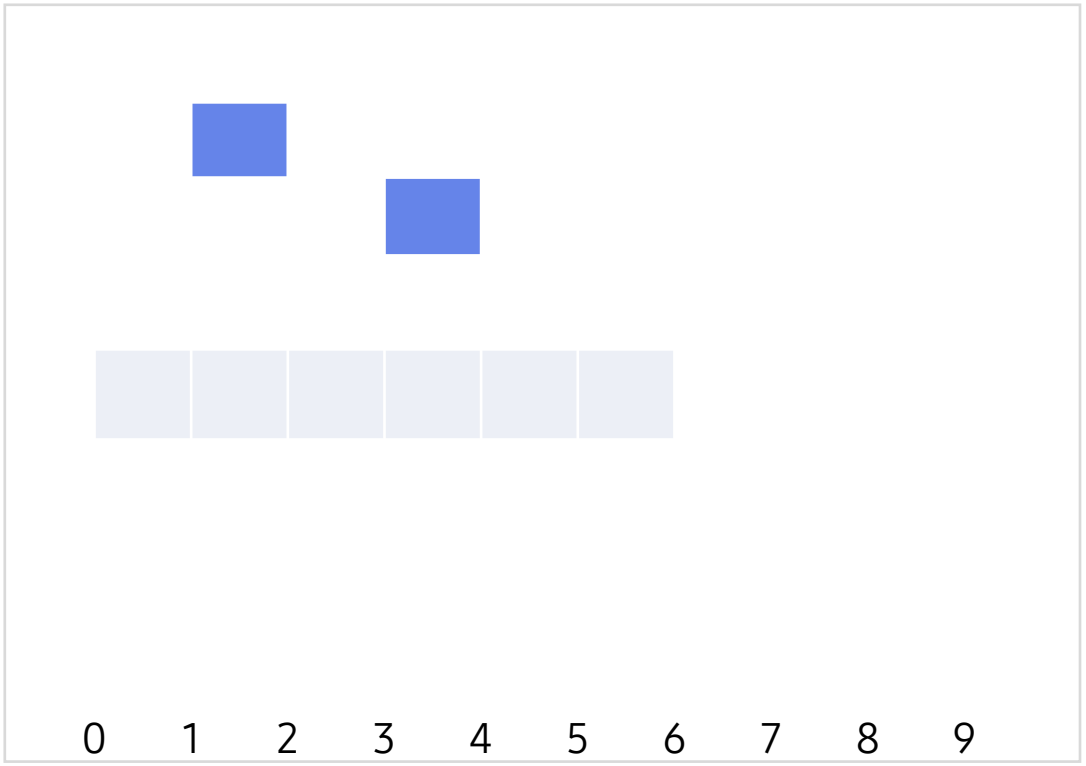


2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.1. ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

ບໍ່ສາມາດເລືອກເອົາກອງປະຊຸມ 3 ເພາະວ່າເວລາເລີ່ມຕົ້ນຂອງມັນ ແມ່ນຫນ້ອຍກວ່າເວລາສິ້ນສຸດຂອງກອງປະຊຸມ 1, ເຊິ່ງແມ່ນ 4.

id	start	finish
0	1	2
1	3	4
2	2	5
3	0	6

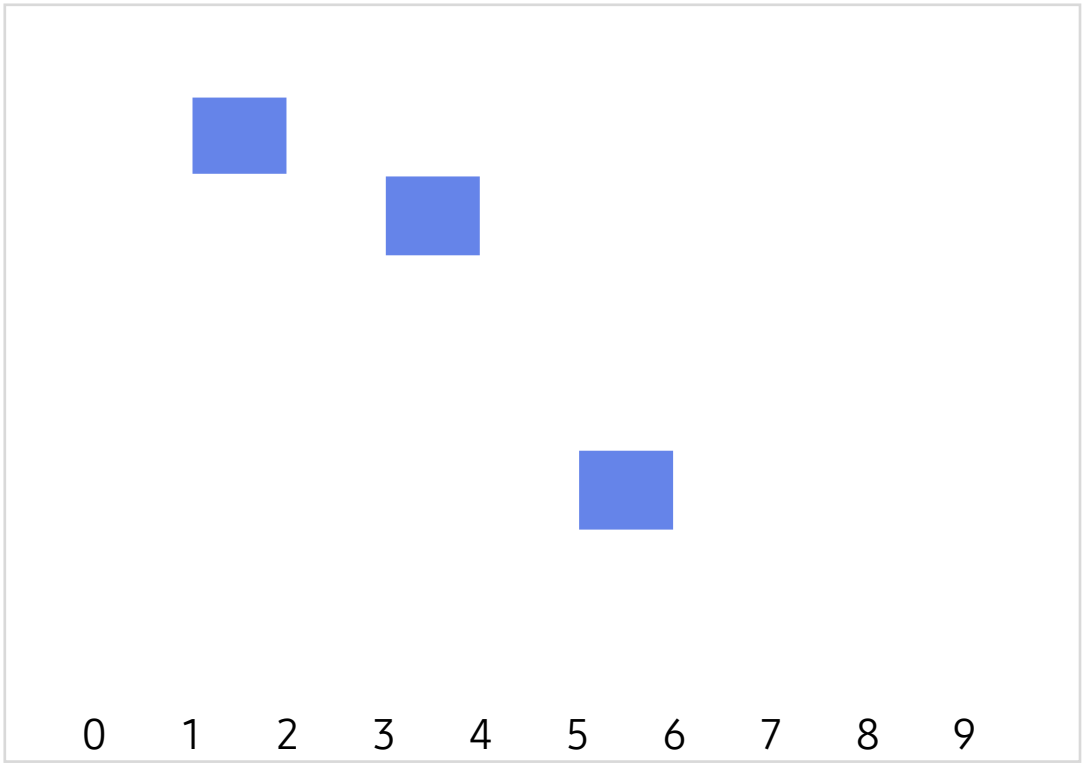


2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.1. ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

- ເວລາເລີ່ມຕົ້ນຂອງກອງປະຊຸມຄັ້ງທີ 4, 5, ແມ່ນຫຼາຍກວ່າເວລາສິ້ນສຸດຂອງກອງປະຊຸມຄັ້ງທີ 1, ດັ່ງນັ້ນຈຶ່ງສາມາດເລືອກໄດ້.

id	start	finish
0	1	2
1	3	4
2	2	5
3	0	6
4	5	6

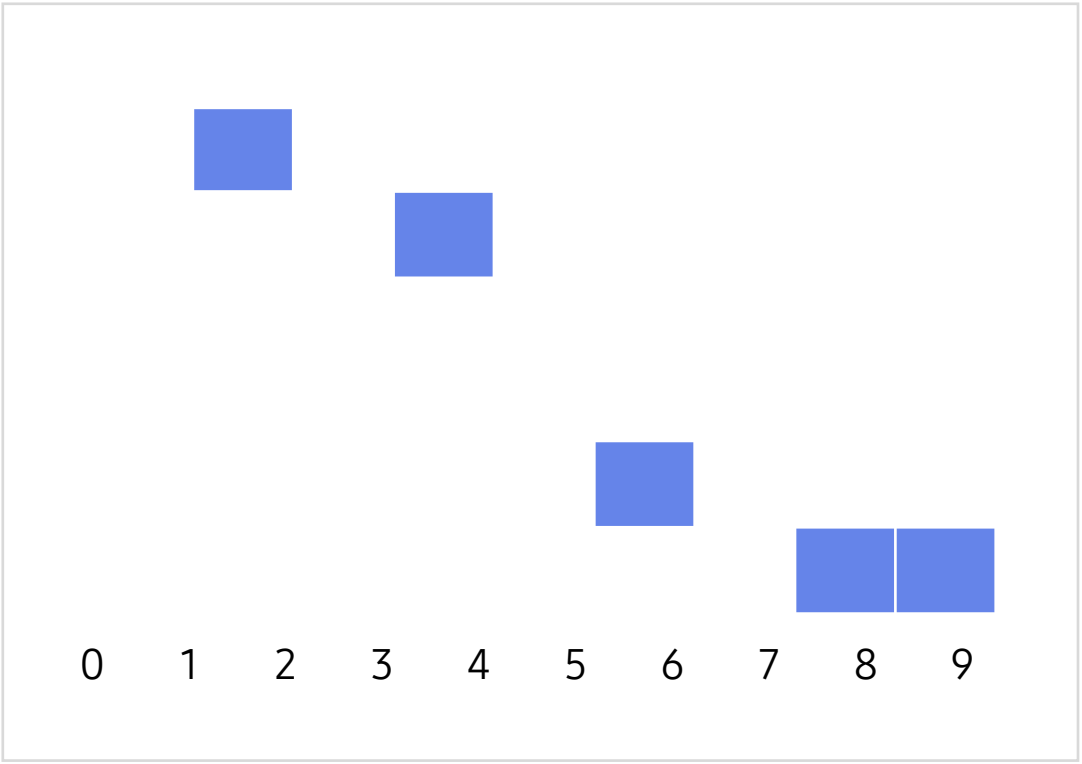


2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.1. ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

ເວລາເລີ່ມຕົ້ນຂອງກອງປະຊຸມທີ 5, ແມ່ນຫຼາຍກວ່າເວລາສິ້ນສຸດຂອງກອງປະຊຸມ 4, ເຊິ່ງແມ່ນ 6, ດັ່ງນັ້ນມັນຍັງສາມາດເລືອກໄດ້.

id	start	finish
0	1	2
1	3	4
2	2	5
3	0	6
4	5	6
5	7	9

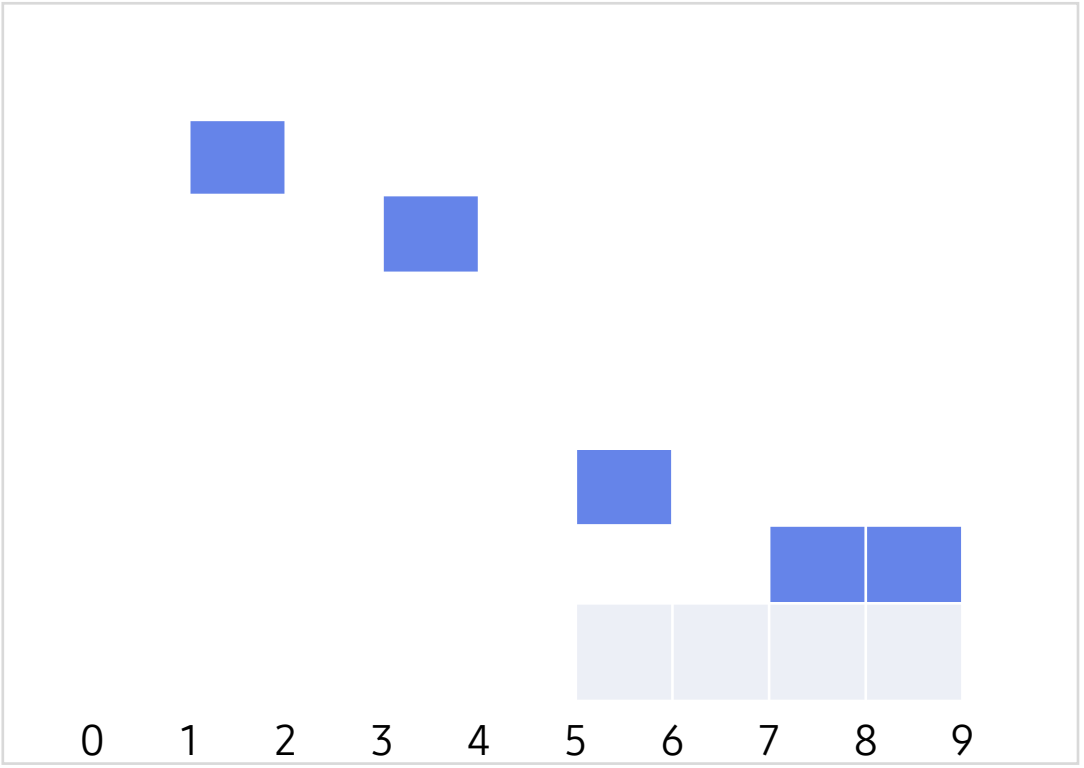


2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.1. ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

ເວລາເລີ່ມຕົ້ນຂອງກອງປະຊຸມ 6, ເຊິ່ງແມ່ນ 5, ແມ່ນຫນ້ອຍກວ່າເວລາເລີ່ມຕົ້ນຂອງກອງປະຊຸມ 5, ດັ່ງນັ້ນມັນບໍ່ສາມາດເລືອກໄດ້.

id	start	finish
0	1	2
1	3	4
2	2	5
3	0	6
4	5	6
5	7	9
6	5	9

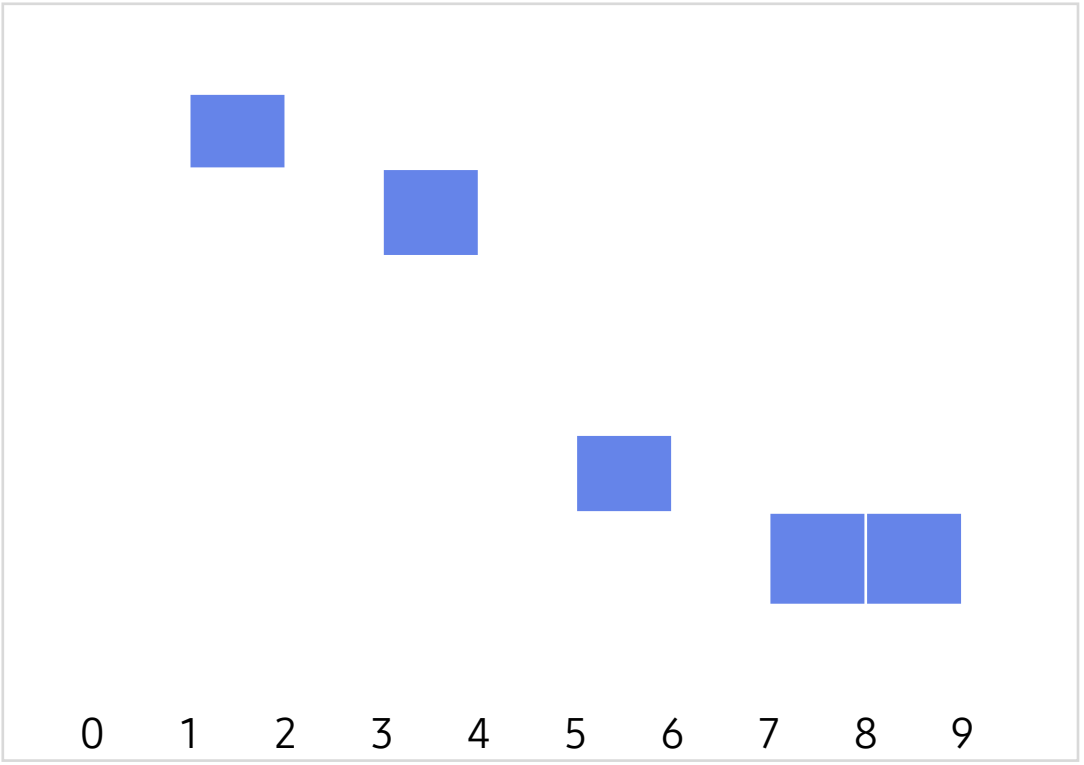


2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.1. ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

ກອງປະຊຸມຄັດເລືອກສຸດທ້າຍແມ່ນ [0, 1, 4, 5] ແລະຈຳນວນກອງປະຊຸມທີ່ເລືອກໄດ້ສູງສຸດແມ່ນ 4.

id	start	finish
0	1	2
1	3	4
2	2	5
3	0	6
4	5	6
5	7	9
6	5	9



2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.2. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

ໃຫ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳໂດຍໃຊ້ວິທີການທີ່ແນະນຳກ່ອນຫນ້ານີ້.

```
1  def activity_selection1(start, finish):
2      result = []
3      i = 0
4      result.append(i)
5      for j in range(1, len(start)):
6          if finish[i] <= start[j]:
7              result.append(j)
8              i = j
9      return result
```

Line 1-4

- ສຳລັບບັນຫາການຄັດເລືອກກິດຈະກຳ, ເວລາເລີ່ມຕົ້ນກອງປະຊຸມ 'start' ແລະ ເວລາສິ້ນສຸດກອງປະຊຸມ 'finish' ແມ່ນເປັນຂໍ້ມູນປ້ອນເຂົ້າ.
- ລາຍການ 'result' ປະກອບດ້ວຍຈຳນວນການປະຊຸມ.
- ດັ່ງນັ້ນ, ດັດຊະນີ I ໃນເບື້ອງຕົ້ນຈະມີຄ່າເປັນ 0 ແລະ ເອົາໃສ່ເຂົ້າໄປໃນຜົນໄດ້ຮັບ.

2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.2. ສ້າງ ແລະ ຂຽນໂປຣແກຣມ

```
1  def activity_selection1(start, finish):
2      result = []
3      i = 0
4      result.append(i)
5      for j in range(1, len(start)):
6          if finish[i] <= start[j]:
7              result.append(j)
8              i = j
9      return result
```

Line 5-9

- ນັບຕັ້ງແຕ່ກິດຈະກຳທີ 0 ໄດ້ຖືກເລືອກແລ້ວ, ດັ່ງນັ້ນຈະເລືອກການປະຊຸມຈາກ 1 ໄປຫາ len(start) -1.
- ຖ້າ finish[i] ໜ້ອຍກວ່າຫຼືເທົ່າກັບການ start[j], ມັນໝາຍຄວາມວ່າກອງປະຊຸມທີ j ບໍ່ທັບຊ້ອນກັບກອງປະຊຸມທີ່ມີຢູ່ແລ້ວ.
- ດັ່ງນັ້ນ, j ເຊິ່ງຊອດຄ່ອງກັບເງື່ອນໄຂນີ້, ໄດ້ຖືກເພີ່ມເຂົ້າໃນຜົນໄດ້ຮັບ.
- ຕອນນີ້ອັບເດດ I ເປັນຄ່າ j ເພື່ອກຳນົດເວລາສຸດທ້າຍຂອງກອງປະຊຸມເປັນກອງປະຊຸມ j.

2. ການເອົາ ວິທີການ Greedy ໄປໃຊ້ແກ້ໄຂບັນຫາການເລືອກກິດຈະກຳ

2.3. Code ที่ใช้สั่งการโปรแกรม

- ບັນຫາການຄັດເລືອກກິດຈະກຳສາມາດໄດ້ຮັບການປະຕິບັດດັ່ງຕໍ່ໄປນີ້.
- ໃນເວລານີ້, start ແລະ finish ສະແດງເຖິງເວລາເລີ່ມຕົ້ນຂອງກອງປະຊຸມແລະເວລາສິ້ນສຸດກອງປະຊຸມ, ຕາມລຳດັບ ແລະ ຜົນໄດ້ຮັບຂອງກອງປະຊຸມແມ່ນຕົວເລກດັດສະນີຂອງກອງປະຊຸມທີ່ເລືອກ ແລະ maximum ແມ່ນຈຳນວນການປະຊຸມສູງສຸດ.

```
start = [1, 3, 2, 0, 5, 8, 5]
finish = [2, 4, 5, 6, 6, 9, 9]
meetings = activity_selection1(start, finish)
maximum = len(meetings)
print(meetings, maximum)
```

```
[0, 1, 4, 5] 4
```

Pop quiz

Q1. ໃນບັນຫາການແລກປ່ຽນຫຼຽນ, ສົມມຸດວ່າມີຫຼຽນ 400 ອອນ ປະກອບຢູ່ນຳ.
ຈົ່ງບອກຜົນໄດ້ຮັບຂອງຂັ້ນຕອນວິທີ coin_change() ຈະກຳນົດການແລກປ່ຽນຫຼຽນ 800 ອອນ.

```
1 coins = [500, 400, 100, 50, 10]
2 amount = int(input("Input the amount: "))
3 changes = coin_change(coins, amount)
4 print(changes, len(changes))
```

```
Input the amount: 800
```

Pair programming

Pair Programming Practice

ແນວທາງ, ກົນໄກ ແລະ ແຜນສຸກເສີນ

ການກະກຽມການສ້າງໂປຣແກຣມຮ່ວມກັນເປັນຄູ່ກ່ຽວຂ້ອງກັບການໃຫ້ຄຳແນະນຳແລະກົນໄກເພື່ອຊ່ວຍໃຫ້ນັກຮຽນຈັບຄູ່ຢ່າງຖືກຕ້ອງແລະ ໃຫ້ເຂົາເຈົ້າເຮັດວຽກເປັນຄູ່. ຕົວຢ່າງ, ນັກຮຽນຄວນປ່ຽນ "ເຮັດ." ການກະກຽມທີ່ມີປະສິດຕິຜົນຕ້ອງໃຫ້ມີແຜນການສຸກເສີນໃນກໍລະນີທີ່ຄູ່ຮ່ວມງານຫນຶ່ງບໍ່ຢູ່ຫຼືຕັດສິນໃຈທີ່ຈະບໍ່ເຂົ້າຮ່ວມດ້ວຍເຫດຜົນໃດຫນຶ່ງ ຫຼືດ້ວຍເຫດຜົນອື່ນ. ໃນກໍລະນີເຫຼົ່ານີ້, ມັນເປັນສິ່ງສໍາຄັນທີ່ຈະເຮັດໃຫ້ມັນຊັດເຈນວ່ານັກຮຽນທີ່ມີປະຕິບັດໜ້າທີ່ຢ່າງຫ້າວຫັນຈະບໍ່ຖືກລົງໂທດຍ້ອນວ່າການຈັບຄູ່ບໍ່ໄດ້ຜົນດີ.

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. ສົມມຸດວ່າຈຳນວນຫຼຽນໃນບັນຫາການແລກປ່ຽນຫຼຽນແມ່ນບໍ່ຈຳກັດ. ຕົວຢ່າງ, ຖ້າທ່ານມີຫຼຽນຕໍ່ໄປນີ້ຢູ່ໃນກະເປົ໋າເງິນຂອງທ່ານ, ທ່ານຄວນແຈກຢາຍ 710 ວອນເປັນການແລກປ່ຽນແນວໃດ?

ກະເປົ໋າເງິນ

₩710

ຈຳນວນແລກປ່ຽນ

ຫຼຽນທີ່ເລືອກ

Q2. ໃຫ້ຈຳນວນຫຼຽນ ແລະ ຈຳນວນການແລກປ່ຽນ, ຊອກຫາຈຳນວນຫຼຽນໜ້ອຍທີ່ສຸດທີ່ທີ່ທ່ານສາມາດແລກປ່ຽນໄດ້.

```
1 coins = list(map(int, input("Input the coins: ").split()))
2 coins.sort(reverse = True)
3 print(coins)
4 amount = int(input("Input the amount: "))
5 changes = coin_change2(coins, amount)
6 print(changes, len(changes))
```

```
Input the coins: 500 50 50 100 50 10 10
[500, 100, 50, 50, 50, 10, 10]
Input the amount: 710
[500, 100, 50, 50, 10] 5
```

Hint | ສັງເກດເຫັນວ່າທ່ານຈັດລຽງຫຼຽນຕາມລຳດັບຈາກໃຫຍ່ຫານ້ອຍໃນແຖວທີ 2 ດ້ວຍ coins.sort(reverse = True).