SAMSUNG

# Samsung Innovation Campus

## Coding, Programming & Data Science

Together for Tomorrow!
Enabling People
Education for Future Generations

# Module 5

# Algorithm 2
# - Sorting Algorithms

Coding, Programming & Data Science

# Chapter Description

## Chapter objectives

- ✓ ນັກຮຽນສາມາດເຂົ້າໃຈບັນຫາການຈັດລຽງ ແລະ ແກ້ໄຂບັນຫາການຈັດລຽງໂດຍໃຊ້ການຈັດລຽງແບບ Bobble, ການຈັດລຽງການແບບ Selection, ການຈັດລຽງແບບ Insertion, ການຈັດລຽງແບບ merge ແລະ ວິທີການຈັດລຽງແບບ Quick. ນອກຈາກນັ້ນ, ຂັ້ນຕອນການຈັດລຽງສາມາດຖືກປະຕິບັດໃນ Python ແລະ ຜູ້ຮຽນສາມາດເຂົ້າໃຈ ແລະ ປຽບທຽບ time complexity ຂອງແຕ່ລະວິທີການຈັດລຽງ.

## Chapter contents

- ✓ Unit 27. ການຈັດລຽງແບບ Bobble, ການຈັດລຽງແບບ Selection, ການຈັດລຽງແບບ Insertion
- ✓ Unit 28. ການຈັດລຽງແບບ Merge
- ✓ Unit 29. ວິທີການຈັດລຽງແບບ Quick

# Chapter 27.

ການຈັດລຽງລຳດັບແບບ Bobble, ການຈັດລຽງລຳດັບແບບ Selection, ການຈັດລຽງລຳດັບແບບ Insertion

## Learning objectives

- ✓ ເຂົ້າໃຈບັນຫາການຈັດລຽງ ແລະ ສາມາດແກ້ໄຂບັນຫາການຈັດລຽງໂດຍ ໃຊ້ເຕັກນິກການຈັດລຽງແບບຕ່າງໆໄດ້.
- ✓ ເຂົ້າໃຈ ແລະ ສາມາດອະທິບາຍຄວາມແຕກຕ່າງລະຫວ່າງການຈັດລຽງແບບ Bubble, ການຈັດລຽງແບບ Selection, ແລະການຈັດລຽງແບບ Insertion.
- ✓ ເຂົ້າໃຈ ແລະ ສາມາດອະທິບາຍ time complexity ທີ່ໃຊ້ໃນການ ຈັດລຽງແບບ Bubble, ການຈັດລຽງການແບບ Selection ແລະ ການຈັດລຽງແບບ Insertion.

## Learning overview

- ✓ ສ້າງ Algorithm ການຈັດລຽງແບບ Bubble, ເຊິ່ງໃຊ້ຈັດລຽງຂໍ້ມູນສອງຕົວທີ່ຢູ່ຕິດກັນ ໂດຍການສະຫຼັບຕຳແໜ່ງພວກມັນ.
- ✓ ສ້າງ Algorithm ການຈັດລຽງແບບ Selection, ເຊິ່ງຄົ້ນຫາ ຄ່າຂໍ້ມູນທີ່ນ້ອຍທີ່ສຸດ ເພື່ອສ້າງລາຍການຈັດລຽງ.
- ✓ ສ້າງ Algorithm ການຈັດລຽງແບບ Insertion, ເຊິ່ງແຊກອົງປະກອບໃໝ່ເຂົ້າໃນລາຍການທີ່ຈັດຮຽງ.

## Concepts you will need to know from previous units

- ✓ ນຳໃຊ້ຕົວປະຕິບັດການ (Operators) ສົມທຽບ ໃນການປຽບທຽບ ສອງຈຳນວນ.
- ✓ ນຳໃຊ້ຟັງຊັນສະຫຼັບຄ່າ (swap function) ເພື່ອສະລັບຕຳແໜ່ງຂອງ ສອງຈຳນວນ.
- ✓ ນຳໃຊ້ Big O notation ເພື່ອວິເຄາະ time complexity ຂອງ Algorithm.

# Keywords

- Sorting Problem
- Bubble Sort
- Selection Sort
- Insertion Sort
- Quadratic Time

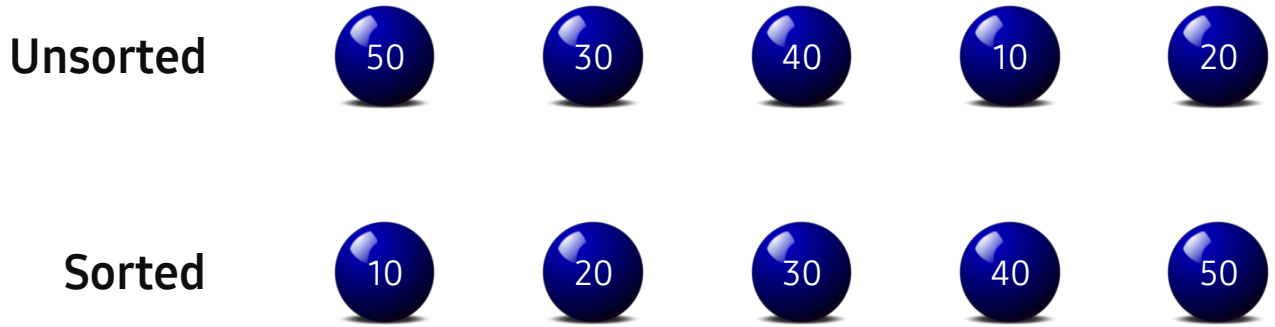
# Mission

# 1. Real world problem

## 1.1. ບັນຫາການຈັດລຽງ

ໃນຊີວິດປະຈຳວັນຂອງພວກເຮົາ, ພວກເຮົາມັກຈະພົບກັບບັນຫາການຈັດຕຳແໜ່ງ

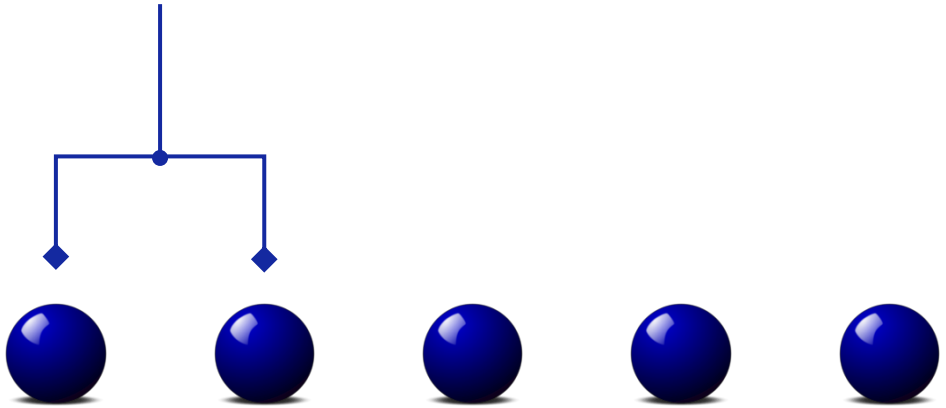
**ຕົວຢ່າງ** ສົມມຸດວ່າທ່ານໄດ້ຮັບໝ່ອຍເຫຼັກສີຟ້າມາ 5 ໝ່ອຍ ທີ່ສະແດງໃຫ້ເຫັນຂ້າງລຸ່ມນີ້. ໂດຍໝ່ອຍເຫຼັກເຫຼົ່ານີ້ແມ່ນມີຮູບຮ່າງ ແລະ ຂະໜາດ ດຽວກັນ ແຕ່ມີນ້ຳໜັກທີ່ແຕກຕ່າງກັນ. ສະນັ້ນ, ມີວິທີໃດແດ່ທີ່ຄວນໃຊ້ ເພື່ອຈັດລຽງໝ່ອຍເຫຼັກເຫຼົ່ານີ້ຕາມລຳດັບຈາກ ນ້ອຍ ຫາ ໃຫຍ່ ຕາມນ້ຳໜັກ ຈາກຮູບຂ້າງລຸ່ມນີ້?



# 2. Mission

## 2.1. ການຈັດລຽງແບບ Bubble : ເມື່ອເຮົາສາມາດສັບປ່ຽນບ່ອນພຽງແຕ່ສອງອົງປະກອບທີ່ຢູ່ຕິດກັນ

- ຈົນຕະນາການວ່າເຄື່ອງມືທີ່ພວກເຮົາສາມາດນຳໃຊ້ໄດ້ແມ່ນອຸປະກອນທີ່ສາມາດເຮັດໄດ້ດັ່ງຕໍ່ໄປນີ້ :
  - ເຮົາສາມາດເອົາໜ່ວຍເຫຼັກສອງໜ່ວຍຢູ່ຕິດກັນ ແລະ ປຽບທຽບນ້ຳໜັກຂອງພວກມັນໄດ້.
  - ໜ່ວຍເຫຼັກສອງໜ່ວຍສາມາດຖືກວາງລົງໂດຍການສັບປ່ຽນຕຳແໜ່ງ.
- ເນື່ອງຈາກວ່າຄອມພິວເຕີບໍ່ສາມາດຈື່ນ້ຳໜັກຂອງໜ່ວຍເຫຼັກໄດ້, ມີພຽງໜ່ວຍເຫຼັກສອງໜ່ວຍທີ່ຢູ່ຕິດກັນສາມາດຖືກສັບປ່ຽນບ່ອນກັນໃນແຕ່ລະຄັ້ງ. ໃນການປຽບທຽບແຕ່ລະຄັ້ງ ສາມາດໃຊ້ຍຸດທະສາດເພື່ອຍ້າຍລູກເຫຼັກທີ່ຫນັກກວ່າໄປທາງດ້ານຂວາ.
- ດ້ວຍຍຸດທະສາດເທິງ, ຕ້ອງໄດ້ມີການປຽບທຽບໜ່ວຍເຫຼັກທັງໝົດຈຳນວນຈັກຄັ້ງເພື່ອເອົາໄປວາງໃຫ້ລຽງລຳດັບຕາມນ້ຳໜັກ?



# 2. Mission

## 2.2. ການຈັດລຽບແບບ Selection: ເມື່ອສາມາດນຳໃຊ້ຫຼັກການຄວາມສົມດູນ

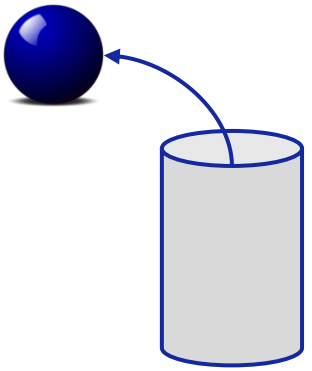
- ນຳໃຊ້ຫຼັກການຄວາມສົມດຸນ. ໂດຍມີຂັ້ນຕອນການປະຕິບັດດັ່ງນີ້:
  - ເຮົາສາມາດເອົາໜ່ວຍເຫຼັກສອງໜ່ວຍທີ່ບໍ່ຢູ່ຕິດກັນ ແລະ ປຽບທຽບນ້ຳໜັກຂອງພວກມັນໄດ້.
  - ໜ່ວຍເຫຼັກທັງສອງສາມາດຖືກສິ່ງກັບຄືນໂດຍການສັບປ່ຽນບ່ອນກັນ.
- ຫຼັກການຄວາມສົມດຸນຂອງຂະຫນາດ ຍັງບໍ່ສາມາດຈິດຈຳນ້ຳຫນັກໄດ້, ແຕ່ມັນສາມາດສັບປ່ຽນບ່ອນທັງສອງໜ່ວຍເຫຼັກທີ່ບໍ່ຢູ່ຕິດກັນໄດ້. ໂດຍທີ່ເຮົາສາມາດເລືອກຫຼັກການໃນການປຽບທຽບໜ່ວຍທຳອິດກັບໜ່ວຍເຫຼັກທີ່ມີນ້ຳຫນັກໜ້ອຍທີ່ສຸດເພື່ອໃຫ້ສາມາດສັບປ່ຽນບ່ອນກັນໄດ້..
- ດ້ວຍຫຼັກການນີ້, ຕ້ອງໄດ້ມີການປຽບທຽບໜ່ວຍເຫຼັກທັງໝົດຈຳນວນຈັກຄັ້ງເພື່ອເອົາໄປວາງໃຫ້ລຽງລຳດັບຕາມນ້ຳໜັກ?

# 2.Mission

## 2.3. ການຈັດລຽງແບບ Insertion: ເມື່ອໄດ້ຮັບໜ່ວຍເຫຼັກພຽງແຕ່ໜຶ່ງໜ່ວຍໃນແຕ່ລະຄັ້ງ

- ໃຊ້ຂະຫນາດຄວາມສົມດຸນ. ເວລາທີ່ມີຂໍ້ຈໍາກັດ :
  - ມີໜ່ວຍເຫຼັກຢູ່ໃນຖັງ ແລະ ພວກມັນສາມາດເອົາອອກໄດ້ເທື່ອລະອັນ.
  - ທຸກຄັ້ງທີ່ໜ່ວຍເຫຼັກຖືກເອົາອອກ, ມັນຄວນຈະຖືກເອົາໄປວາງໄວ້ໃນຕໍາແຫນ່ງທີ່ຖືກຈັດລຽງສະເຫມີ.
- ໃນກໍລະນີນີ້, ບໍ່ສາມາດປຽບທຽບໜ່ວຍເຫຼັກທັງໝົດໃນເວລາດຽວກັນ, ເນື່ອງຈາກວ່າມີໜ່ວຍເຫຼັກຖືກຈັດລຽງແລ້ວເທົ່ານັ້ນທີ່ຖືກເອົາອອກຈາກຖັງແຕ່ລະຄັ້ງ, ໜ່ວຍເຫຼັກໃໝ່ຈະຖືກຄົ້ນຫາຕາມລໍາດັບ..
- ດ້ວຍວິທີນີ້, ຕ້ອງໄດ້ມີການປຽບທຽບໜ່ວຍເຫຼັກທັງໝົດຈໍານວນຈັກຄັ້ງເພື່ອເອົາໄປວາງໃຫ້ລຽງລໍາດັບຕາມນໍ້າໜັກ?

ໄດ້ຈັດລຽງແລ້ວ
(Already Sorted)



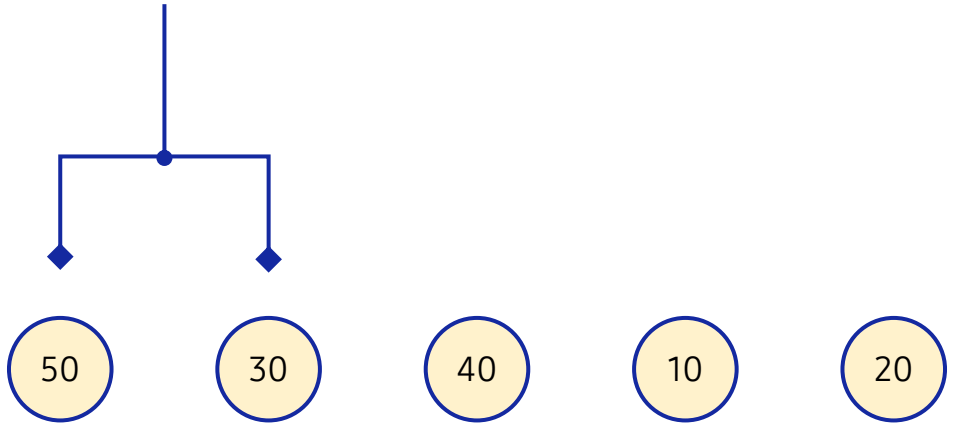
# Key concept

# 1. ຫຼັກການຈັດລຽງແບບ Bubble

## 1.1. ຕົວຢ່າງການຈັດລຽງແບບ Bubble

ການຈັດລຽງແບບ Bubble ເປັນ Algorithm ຈັດລຽງລຳດັບທີ່ປຽບທຽບສອງອົງປະກອບທີ່ຢູ່ຕິດກັນ ແລະ ສັບປ່ຽນບ່ອນພວກມັນຖ້າບໍ່ຖືກລຳດັບ.

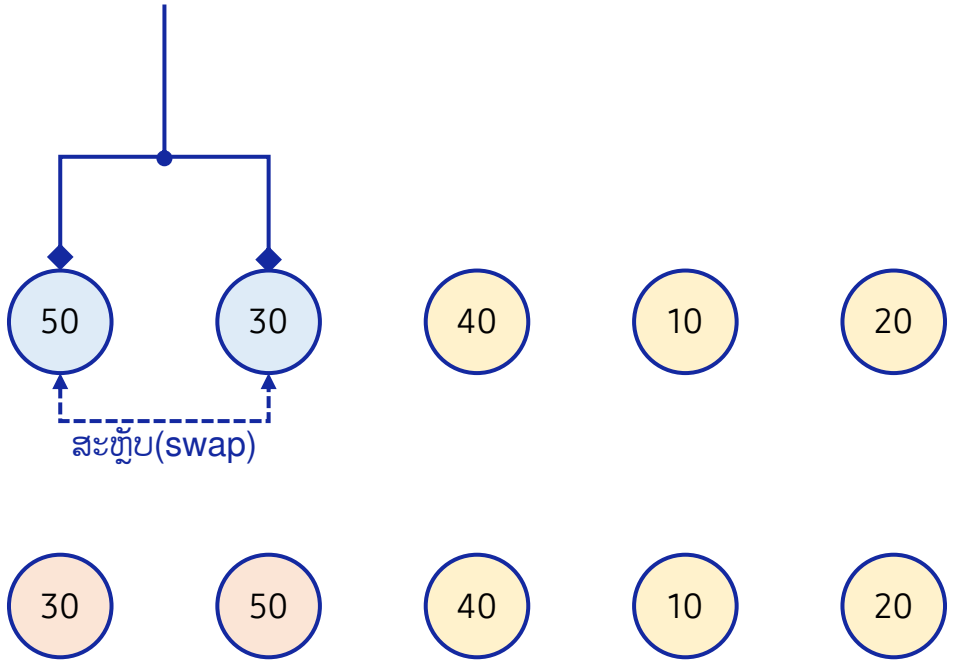
Ex ສົມມຸດວ່າ ຖ້າຕ້ອງການທີ່ຈະຈັດລຽງລຳດັບຂໍ້ມູນຊຸດນີ້ [50, 30, 40, 10, 20] ໃຫ້ປະຕິບັດດັ່ງຕໍ່ໄປນີ້.



# 1. ການຈັດລຽງແບບ Bubble

## 1.2. ຂະບວນການຂອງການຈັດລຽງແບບ bubble

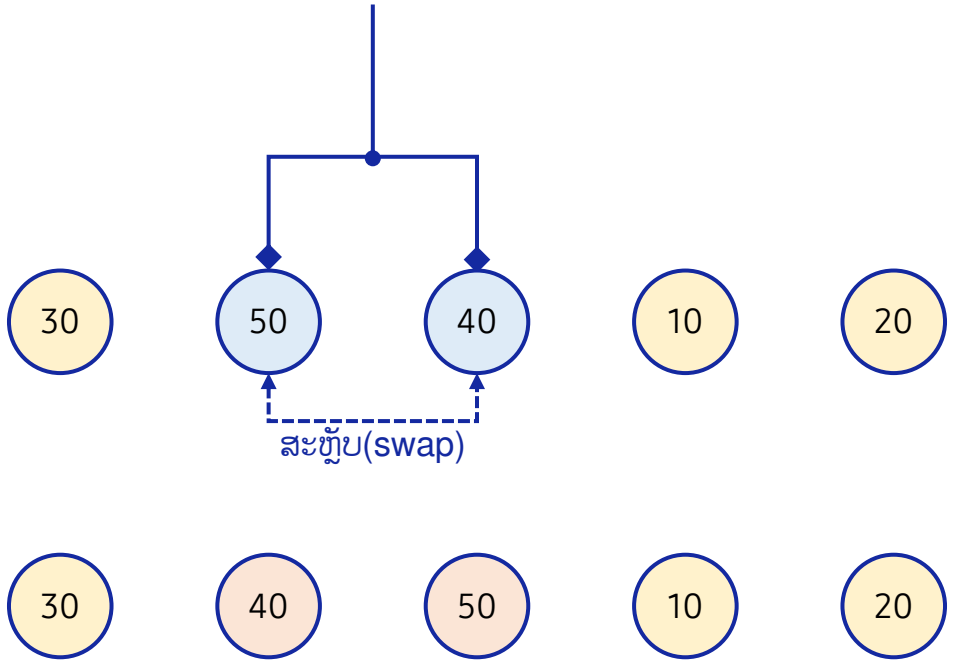
ກ່ອນອື່ນເລີ່ມຈາກ, ປຽບທຽບອົງປະກອບທີ 1 ທີ່ມີຄ່າເທົ່າກັບ 50 ກັບອົງປະກອບທີ 2 ທີ່ມີຄ່າເທົ່າກັບ 30. ເນື່ອງຈາກ 50 ໃຫຍ່ກວ່າ 30, ສອງອົງປະກອບນີ້ຈຶ່ງຕ້ອງສັບປ່ຽນບ່ອນກັນ.



# 1. ການຈັດລຽງແບບ Bubble

## 1.2. ຂະບວນການຂອງການຈັດລຽງແບບ bubble

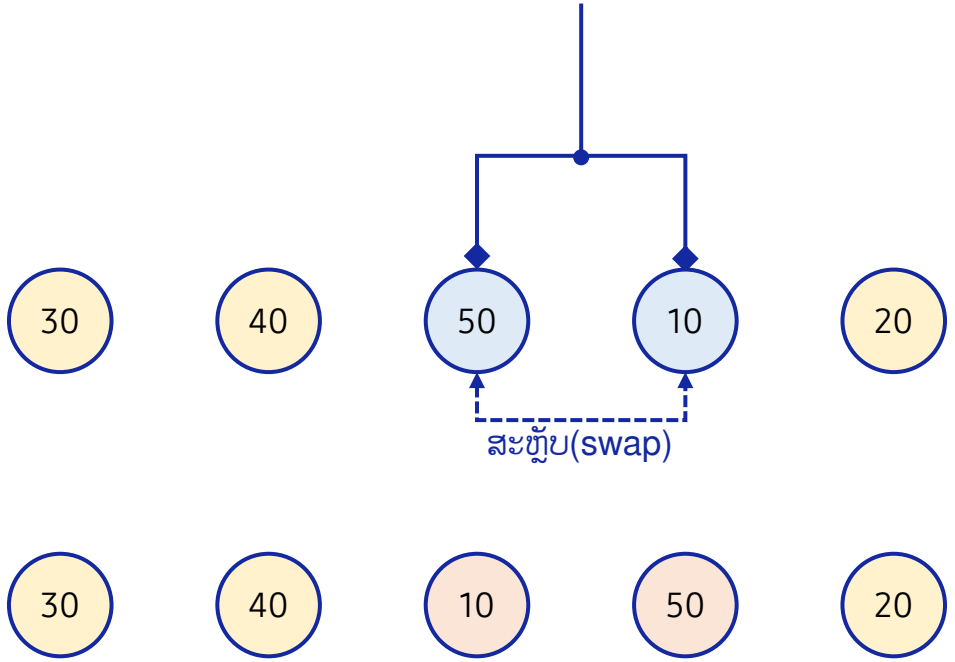
ຕໍ່ໄປ, ປຽບທຽບອົງປະກອບທີ 2 ທີ່ມີຄ່າເທົ່າກັບ 50 ກັບອົງປະກອບທີ 3 ທີ່ມີຄ່າເທົ່າກັບ 40. ເນື່ອງຈາກ 50 ໃຫຍ່ກວ່າ 40, ສອງອົງປະກອບນີ້ຈຶ່ງຕ້ອງສັບປ່ຽນບ່ອນກັນ.



# 1. ການຈັດລຽງແບບ Bubble

## 1.2. ຂະບວນການຂອງການຈັດລຽງແບບ bubble

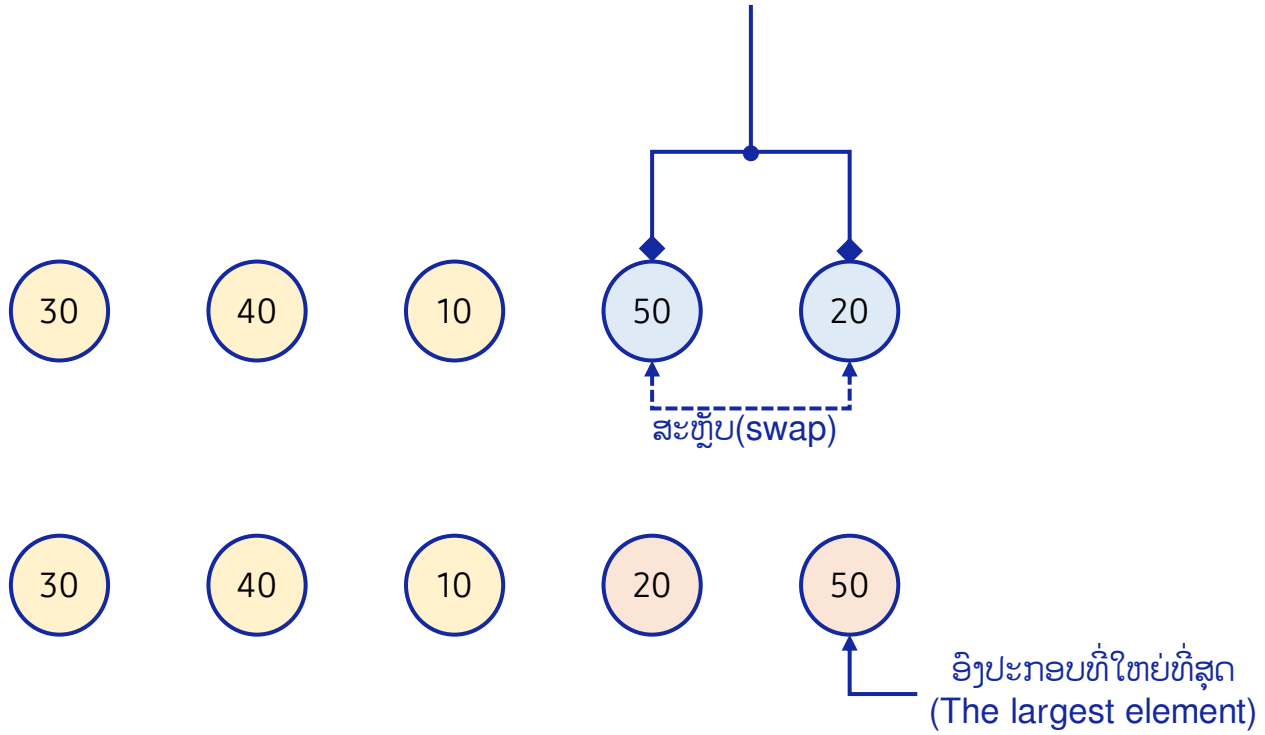
ຕໍ່ໄປ, ປຽບທຽບອົງປະກອບທີ 3 ທີ່ມີຄ່າເທົ່າກັບ 50 ກັບອົງປະກອບຕຳແໜ່ງທີ 4 ທີ່ມີຄ່າເທົ່າກັບ 10. ເນື່ອງຈາກ 50 ໃຫຍ່ກວ່າ 10, ສອງອົງປະກອບນີ້ຈຶ່ງຕ້ອງສັບປ່ຽນບ່ອນກັນ.



# 1. ການຈັດລຽງແບບ Bubble

## 1.2. ຂະບວນການຂອງການຈັດລຽງແບບ bubble

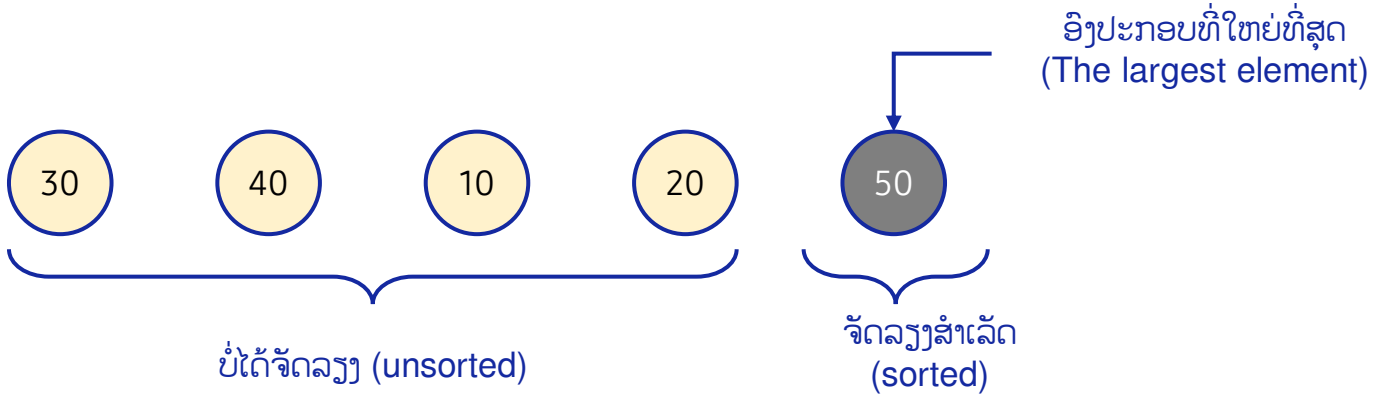
ລຳດັບຖັດໄປ, ປຽບທຽບອົງປະກອບທີ 4 ທີ່ມີຄ່າເທົ່າກັບ 50 ກັບອົງປະກອບຕຳແໜ່ງສຸດທ້າຍ ທີ່ມີຄ່າເທົ່າກັບ 20. ແຕ່ 50 ໃຫຍ່ກວ່າ 20, ສອງອົງປະກອບນີ້ຈຶ່ງຕ້ອງສັບປ່ຽນບ່ອນກັນ.



# 1. ການຈັດລຽງແບບ Bubble

## 1.2. ຂະບວນການຂອງການຈັດລຽງແບບ bubble

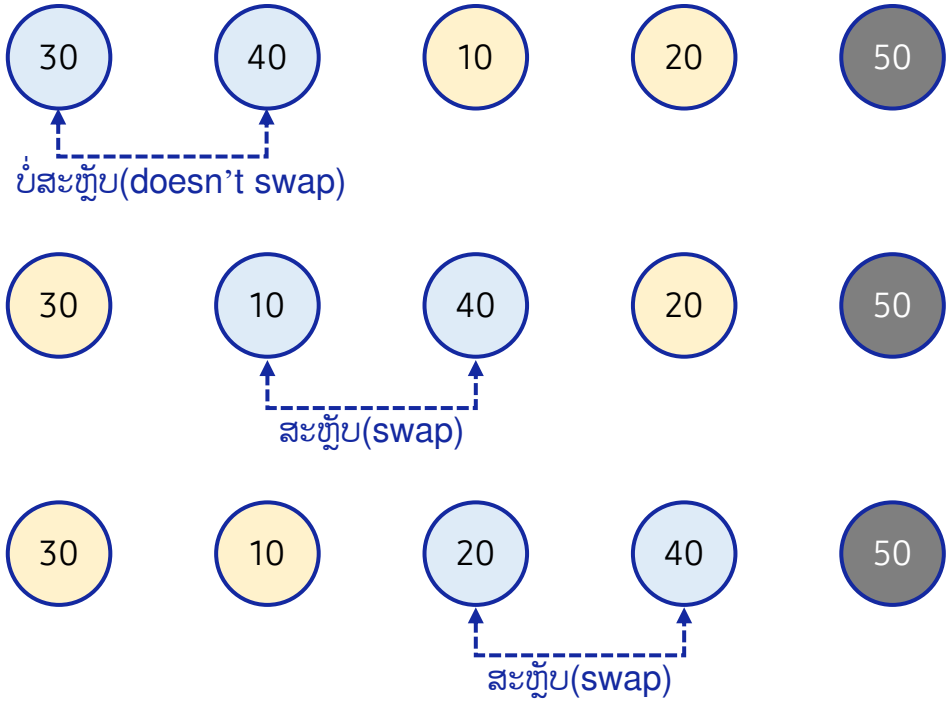
ໃຫ້ສັງເກດວ່າຫຼັງຈາກປະຕິບັດຮອບທຳອິດ, ອົງປະກອບທີ່ໃຫຍ່ທີ່ສຸດໃນລາຍການຄື 50 ຈຶ່ງຖືກຈັດຢູ່ເບື້ອງຂວາສຸດ. ສະນັ້ນອົງປະກອບສຸດທ້າຍ 50 ຈາກການຈັດລຽງໃນຮອບທີ 1 ຈະບໍ່ໄດ້ຈັດລຽງອີກໃນຮອບທີ 2.



# 1. ການຈັດລຽງແບບ Bubble

## 1.2. ຂະບວນການຂອງການຈັດລຽງແບບ bubble

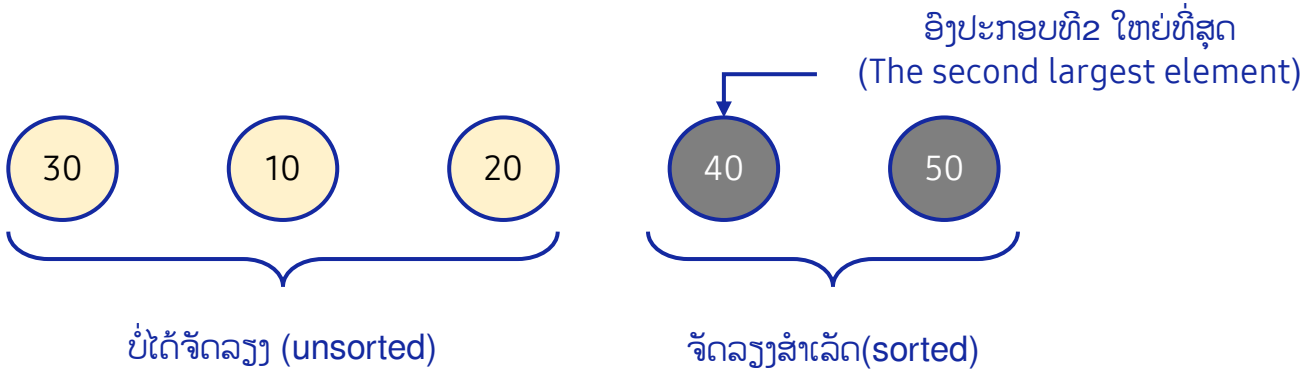
- ສືບຕໍ່ການຈັດລຽງແບບ bubble ໃນລາຍການທີ່ຍັງບໍ່ໄດ້ຈັດລຽງ ຍົກເວັ້ນອົງປະກອບສຸດທ້າຍ..



# 1. ການຈັດລຽງແບບ Bubble

## 1.2. ຂະບວນການຂອງການຈັດລຽງແບບ bubble

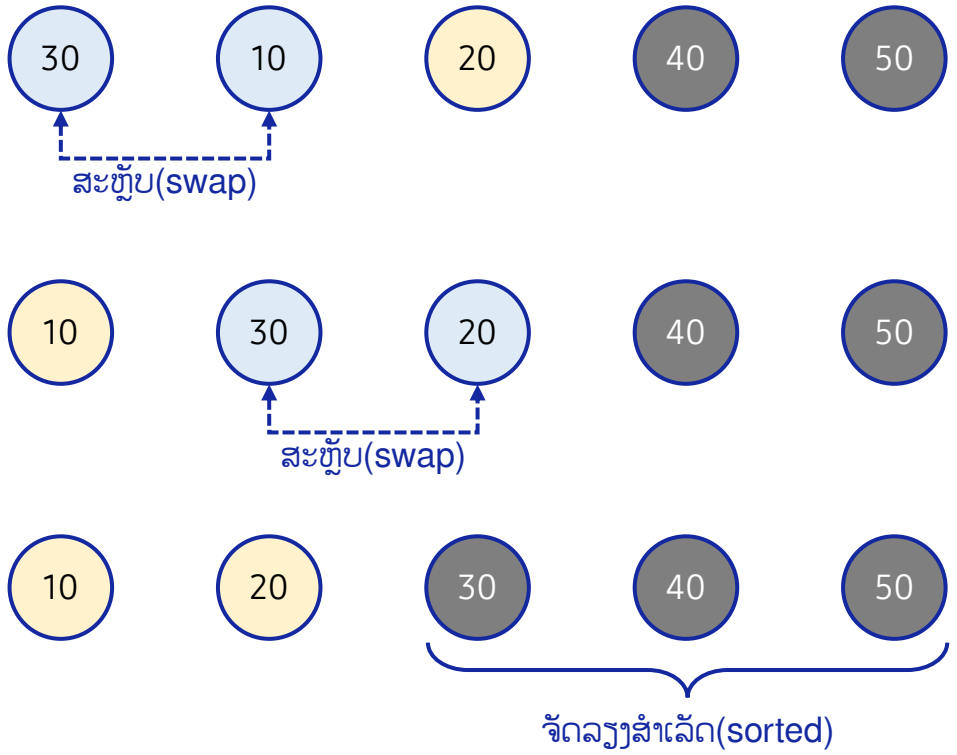
ໃຫ້ສັງເກດວ່າຜ່ານການຈັດລຽງຮອບທີ 2 ເຫັນວ່າອົງປະກອບທີ່ໃຫຍ່ທີ່ສຸດແມ່ນ 40, ເຊິ່ງເປັນອົງປະກອບທີ່ຈະບໍ່ໄດ້ຖືກປຽບທຽບໃນການຈັດລຽງຮອບຕໍ່ໄປ.



# 1. ການຈັດລຽງແບບ Bubble

## 1.2. ຂະບວນການຂອງການຈັດລຽງແບບ bubble

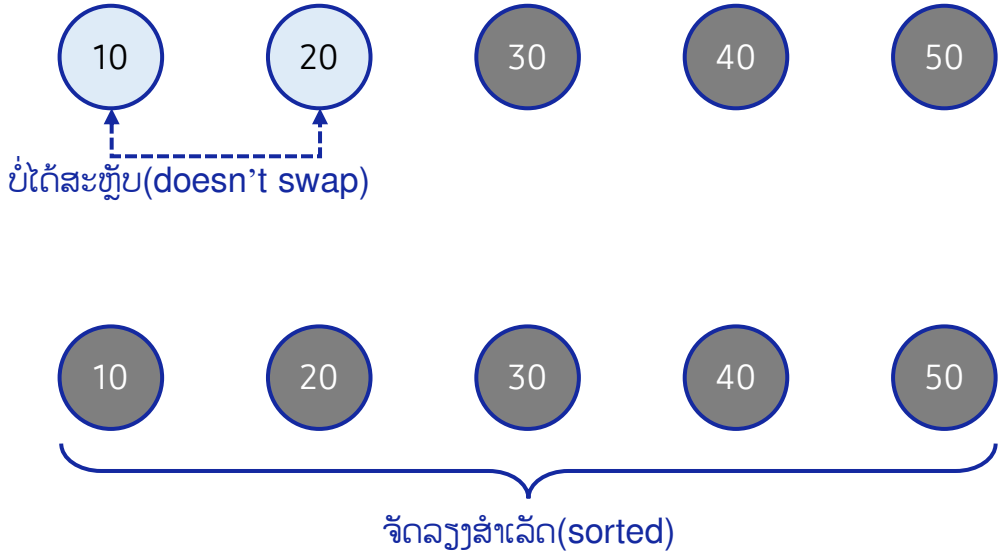
ใช้อิทิภาพจัดลงงแบบดงอกัน ในรอบที่ 3.



# 1. ການຈັດລຽງແບບ Bubble

## 1.2. ຂະບວນການຂອງການຈັດລຽງແບບ bubble

ດ້ວຍວິທີການດຽວກັນນີ້, ການຈັດລຽງແບບ bubble ສາມາດດຳເນີນການຕໍ່ຈົນກວ່າທຸກອົງປະກອບຈະຖືກຈັດລຽງສຳເລັດ.



# 2. Selection Sort

## 2.1. ຕົວຢ່າງການຈັດລຽງແບບ Selection

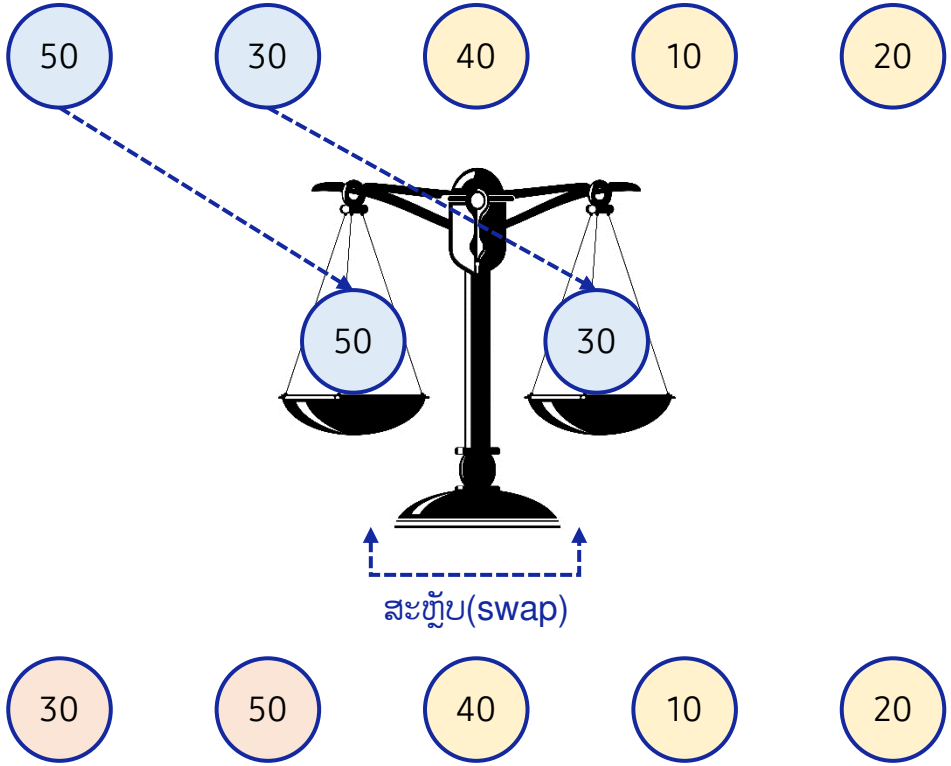
ການຈັດລຽງແບບ Selection ແມ່ນວິທີການຈັດລຽງທີ່ຊອກຫາອົງປະກອບທີ່ນ້ອຍທີ່ສຸດໃນລາຍການຂໍ້ມູນທັງໝົດ ແລ້ວເອົາມາວາງໄວ້ທາງຊ້າຍສຸດ. ການເຮັດແບບນີ້, ແມ່ນອາໃສຫຼັກການປຽບທຽບອົງປະກອບທຳອິດຂອງລາຍການທີ່ບໍ່ໄດ້ຈັດລຽງກັບສ່ວນທີ່ເຫຼືອຂອງລາຍການ ແລະ ສັບປ່ຽນບ່ອນກັບອົງປະກອບທີ່ນ້ອຍກວ່າ.

**Ex** ສົມມຸດວ່າ ຕ້ອງການທີ່ຈະຈັດລຽງລຳດັບຂໍ້ມູນຊຸດນີ້ [50, 30, 40, 10, 20] ໃຫ້ປະຕິບັດດັ່ງຕໍ່ໄປນີ້.

# 2. ການຈັດລຽງແບບSelection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

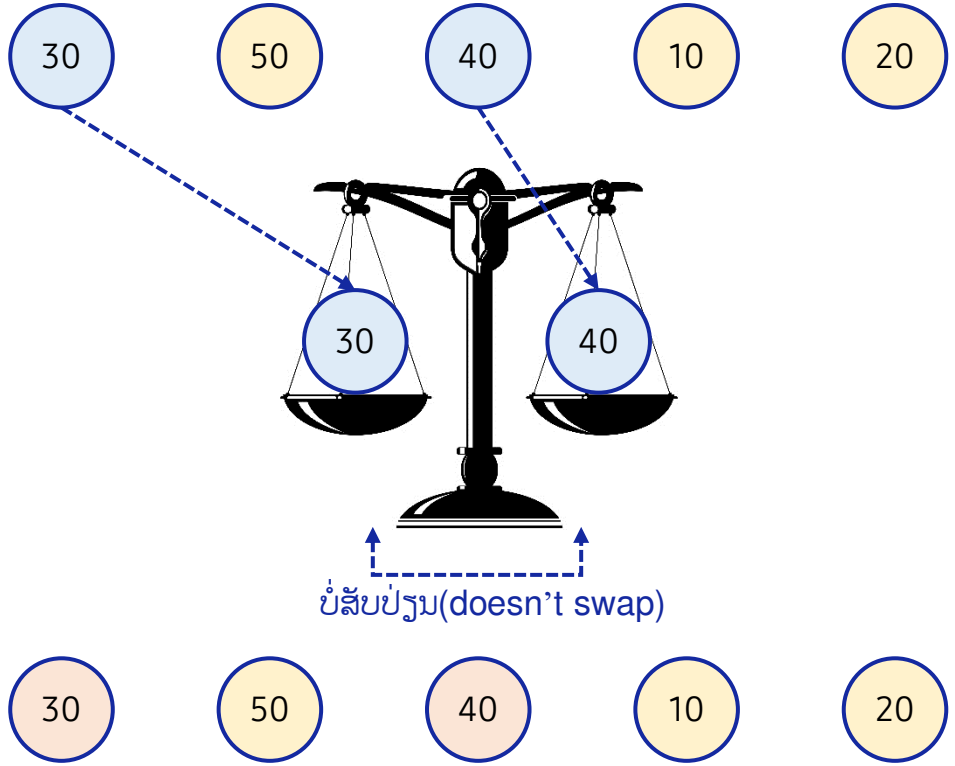
- ທຳອິດ, ປຽບທຽບອົງປະກອບທີ 1 ທີ່ມີຄ່າເທົ່າກັບ 50 ກັບອົງປະກອບທີ 2 ທີ່ມີຄ່າເທົ່າກັບ 30. ເນື່ອງຈາກ 50 ແມ່ນໃຫຍ່ກວ່າ 30, ສອງອົງປະກອບນີ້ຈຶ່ງຕ້ອງສັບປ່ຽນບ່ອນກັນ.



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

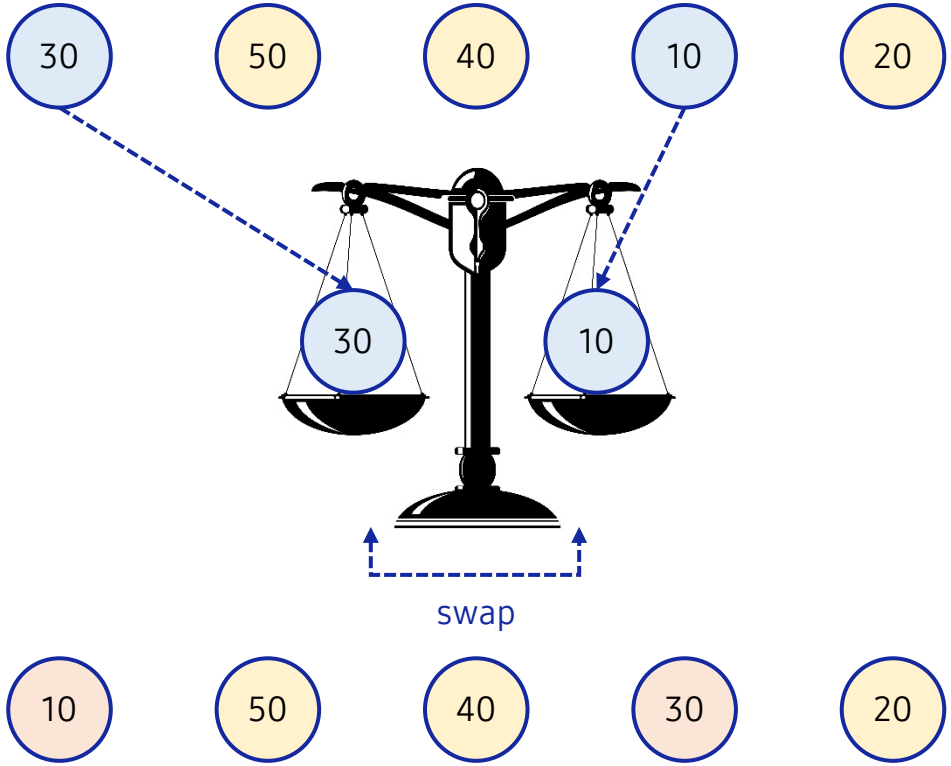
- ຕໍ່ໄປ, ປຽບທຽບອົງປະກອບທີ 1 ທີ່ມີຄ່າເທົ່າກັບ 30 ກັບອົງປະກອບທີ 3 ທີ່ມີຄ່າເທົ່າກັບ 40. ເນື່ອງຈາກ 30 ນ້ອຍກວ່າ 40, ຖືກລໍາດັບແລ້ວ ຈຶ່ງບໍ່ໄດ້ສັບປ່ຽນບ່ອນກັນ



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

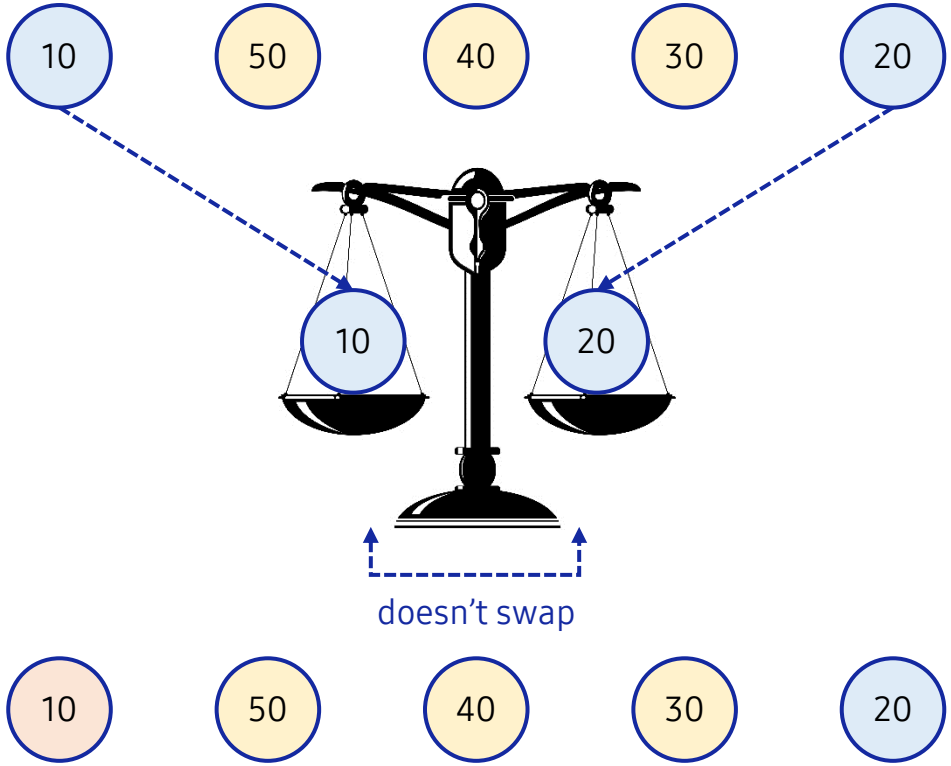
- ຖັດມາ, ປຽບທຽບອົງປະກອບທີ 1 ທີ່ມີຄ່າເທົ່າກັບ 30 ກັບອົງປະກອບທີ 4 ທີ່ມີຄ່າເທົ່າກັບ 10. ເນື່ອງຈາກ 10 ນ້ອຍກວ່າ 30, ສອງອົງປະກອບນີ້ຈຶ່ງຕ້ອງສັບປ່ຽນບ່ອນກັນ.



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

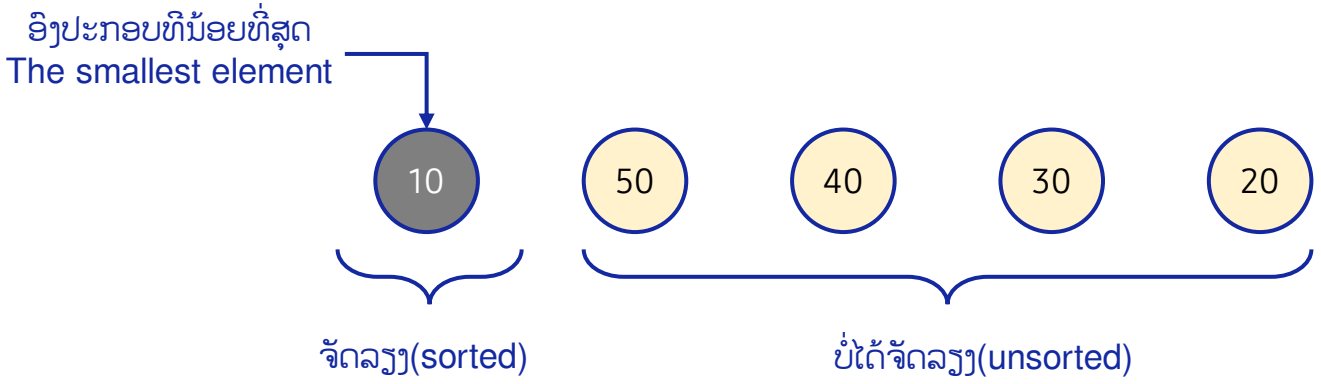
- ຕໍ່ໄປ, ປຽບທຽບອົງປະກອບທີ 1 10 ທີ່ມີຄ່າເທົ່າກັບ 10 ກັບອົງປະກອບທີ 5 1 ທີ່ມີຄ່າເທົ່າກັບ 20. ເນື່ອງຈາກ 10 ນ້ອຍກວ່າ 20, ຖືກລໍາດັບແລ້ວ ຈຶ່ງບໍ່ໄດ້ສັບປ່ຽນບ່ອນກັນ.



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

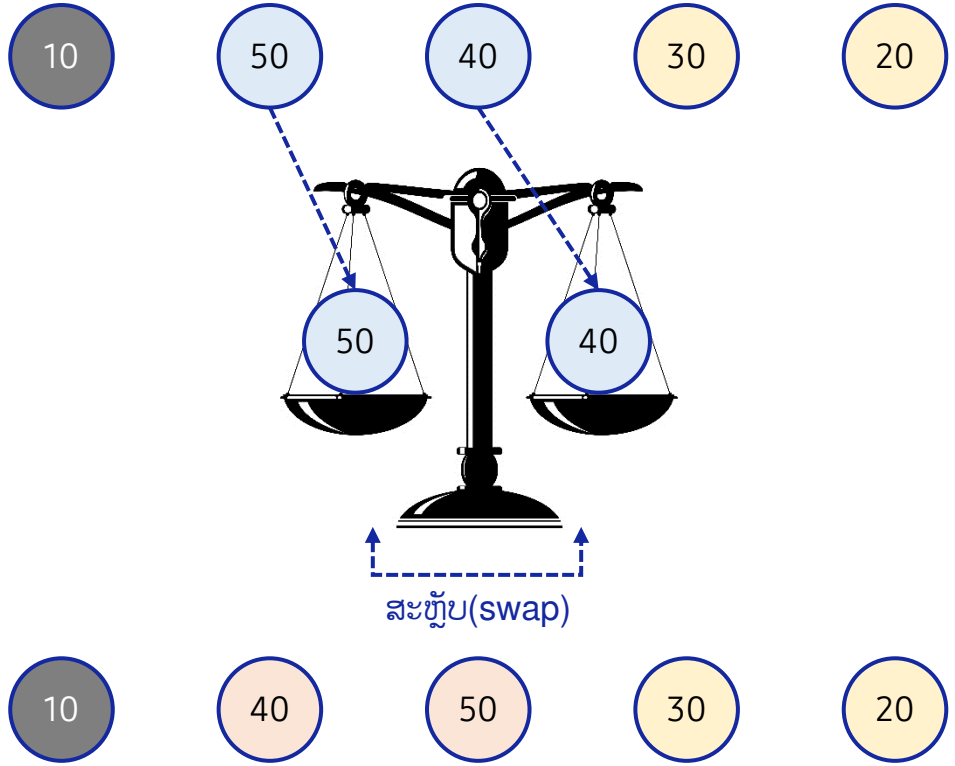
ໃຫ້ສັງເກດວ່າໃນ ວິທີການຈັດລຽງແບບ Selection, ອົງປະກອບທີ່ນ້ອຍທີ່ສຸດ ມີຄ່າເທົ່າກັບ 10 ຖືກວາງໄວ້ເປັນໂຕທຳອິດໃນລາຍການທີ່ຖືກຈັດລຽງແລ້ວ. ເນື່ອງຈາກອົງປະກອບທີ 1 ແມ່ນລາຍການທີ່ຈັດລຽງແລ້ວ, ຈຶ່ງບໍ່ໄດ້ເອົາມາຄັດເລືອກອີກໃນຮອບຕໍ່ໄປ.



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

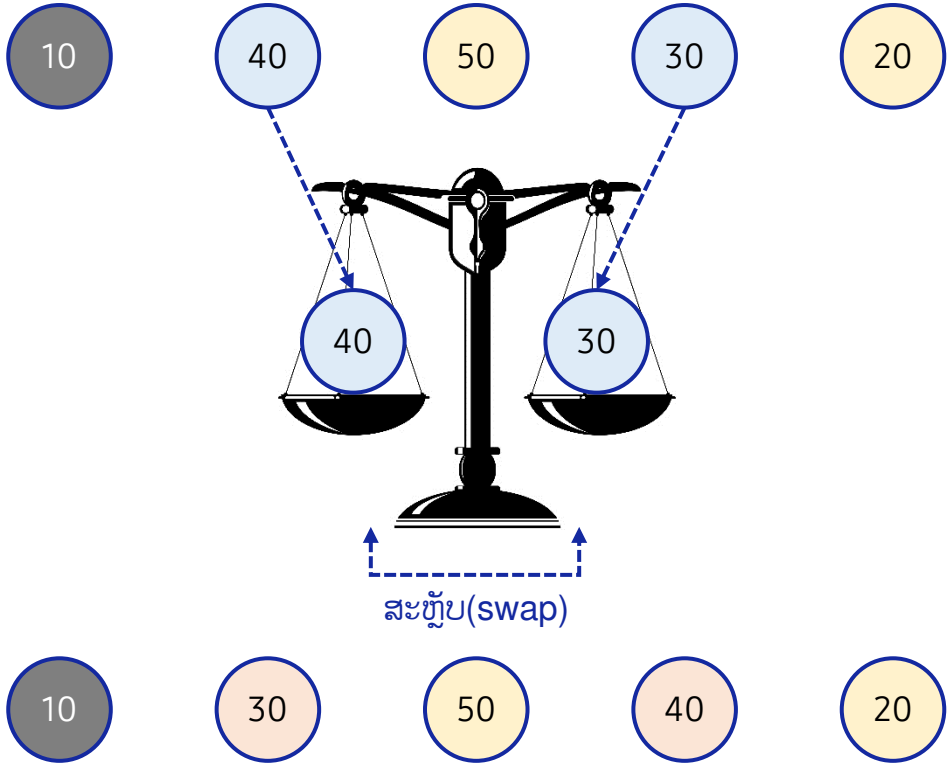
ການຈັດລຽງແບບ Selection ດຳເນີນການຕໍ່ກັບຂໍ້ມູນທີ່ຢູ່ໃນລາຍການທີ່ບໍ່ໄດ້ຈັດລຽງ, ນັ້ນແມ່ນປຽບທຽບອົງປະກອບທີ 2 ທີ່ມີຄ່າເທົ່າກັບ 50 ກັບ ອົງປະກອບທີ 3 ທີ່ມີຄ່າເທົ່າກັບ 40. ເນື່ອງຈາກ 40 ນ້ອຍກວ່າ 50, ສອງອົງປະກອບນີ້ຈຶ່ງຕ້ອງສັບປ່ຽນບ່ອນກັນ.



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

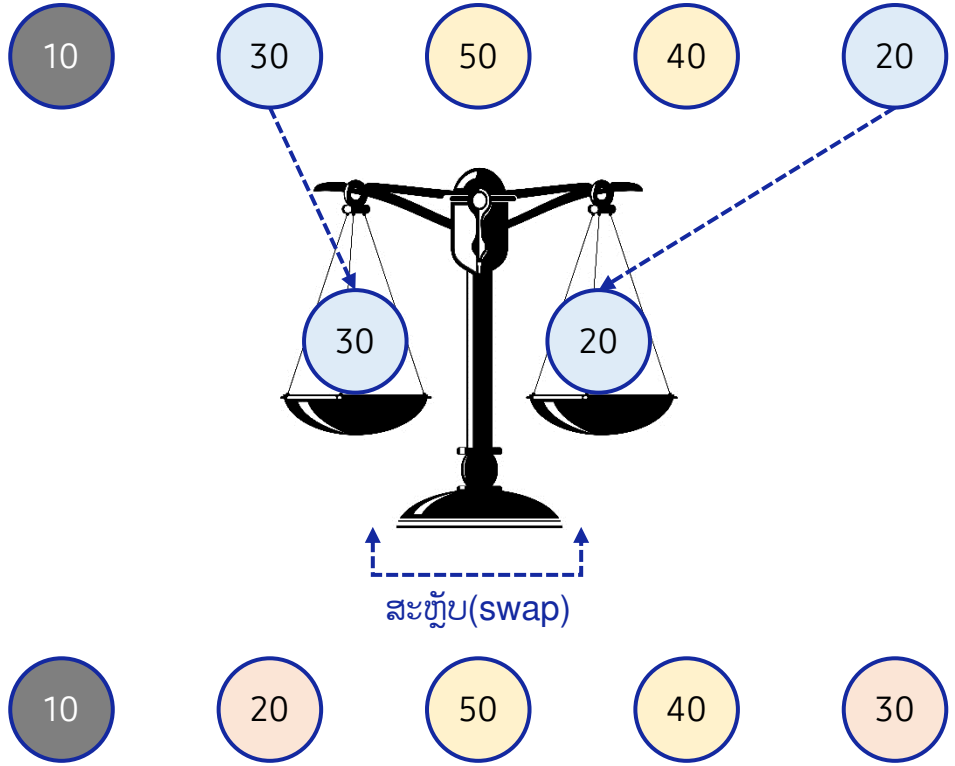
- ການຈັດລຽງແบบ Selection ດຳເນີນການຕໍ່ ເພື່ອປຽบທຽบອົງປะกอบที 2 ที่มีค่าเท่ากับ 40 กับอົງປะกอบที 4 ที่มีค่าเท่ากับ 30, เนื่องจาก 30 น้อยกว่า 40 สองอົງປะกอบนี้จึ่งต้องสับปลี่ยนซ่อนกัน.



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

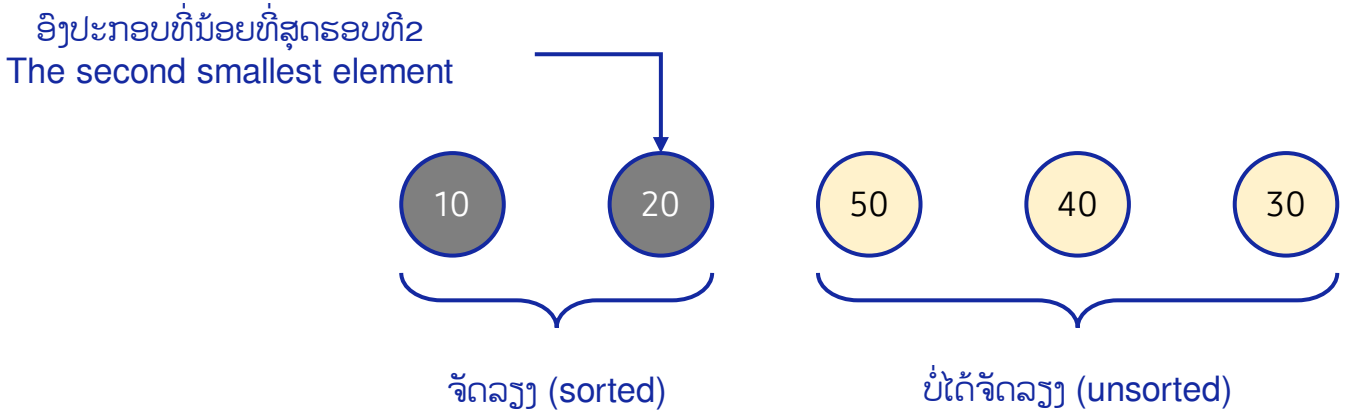
- ການຈັດລຽງແບບ Selection ດຳເນີນການຕໍ່ ເພື່ອປຽບທຽບອົງປະກອບທີ 2 ທີ່ມີຄ່າເທົ່າກັບ 30 ກັບອົງປະກອບທີ 5 ທີ່ມີຄ່າເທົ່າກັບ 20, ເນື່ອງຈາກ 20 ນ້ອຍກວ່າ 30 ສອງອົງປະກອບນີ້ຈຶ່ງຕ້ອງສັບປ່ຽນບ່ອນກັນ.



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

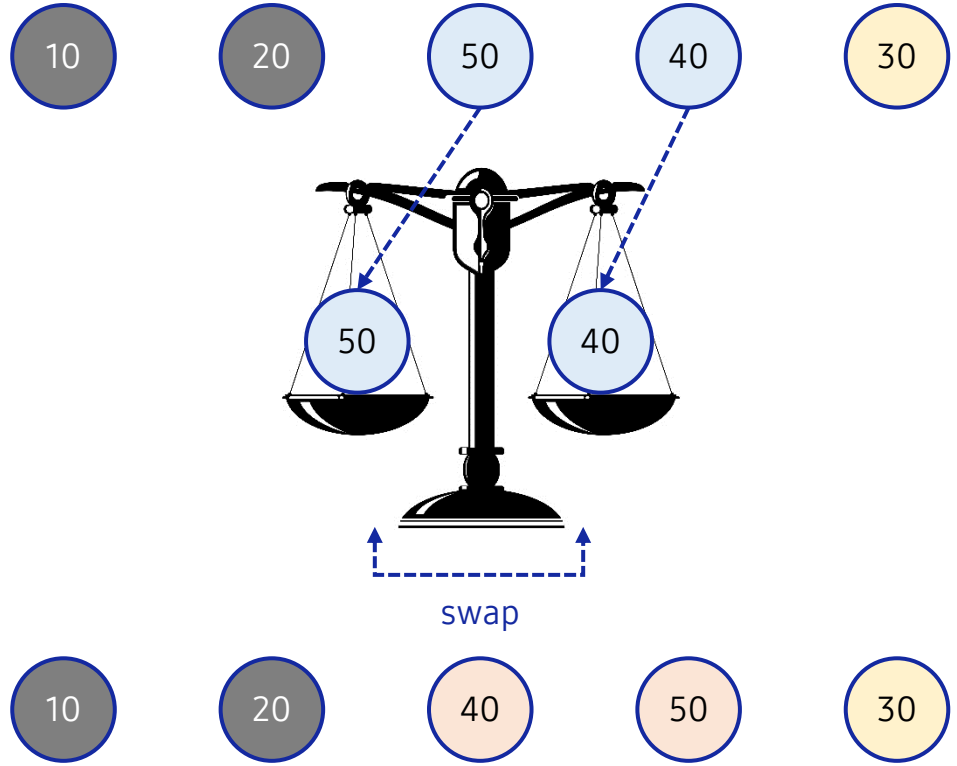
ໃຫ້ສັງເກດວ່າຜ່ານການຈັດລຽງ ໄດ້ອົງປະກອບທີ່ນ້ອຍທີ່ສຸດຂອງ ຮອບທີ 2 ແມ່ນ 20 ແລະ ອົງປະກອບດັ່ງກ່າວຈະບໍ່ຖືກເອົາມາປຽບທຽບອີກໃນຮອບຕໍ່ໄປ.



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

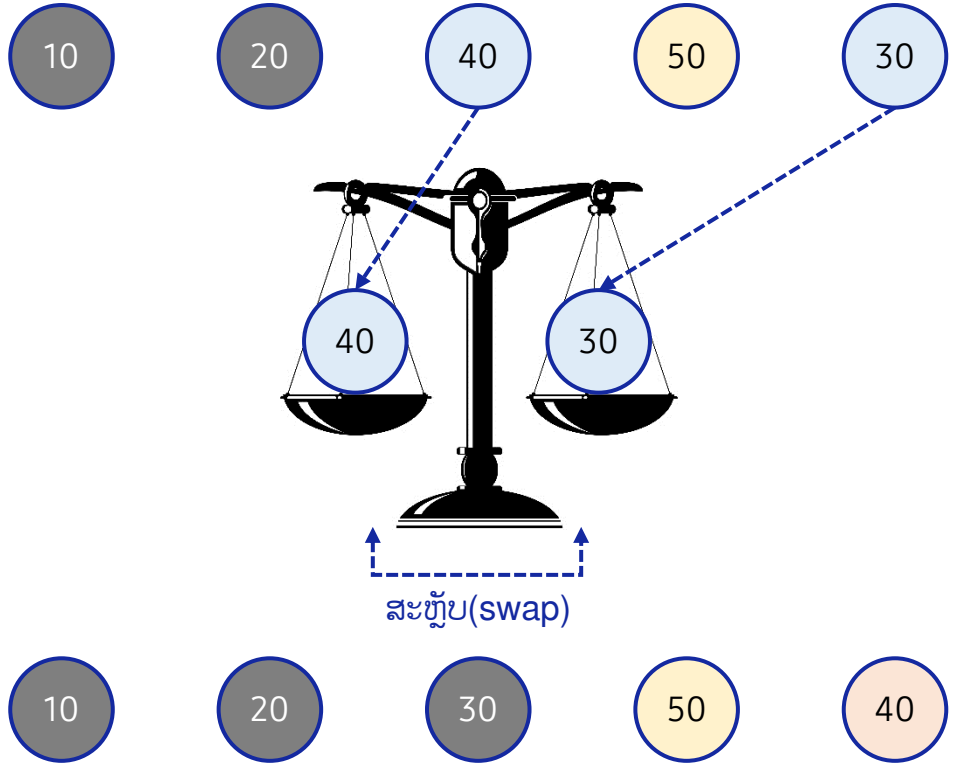
- ດຳເນີນການຕໍ່ໃນຮອບທີ 3 ເພື່ອປຽບທຽບອົງປະກອບທີ 3 ທີ່ມີຄ່າເທົ່າກັບ 50 ກັບອົງປະກອບທີ 4 ທີ່ມີຄ່າເທົ່າກັບ 40, ເນື່ອງຈາກ 40 ນ້ອຍກວ່າ 50 ສອງອົງປະກອບນີ້ຈຶ່ງຕ້ອງສັບປ່ຽນບ່ອນກັນ.



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

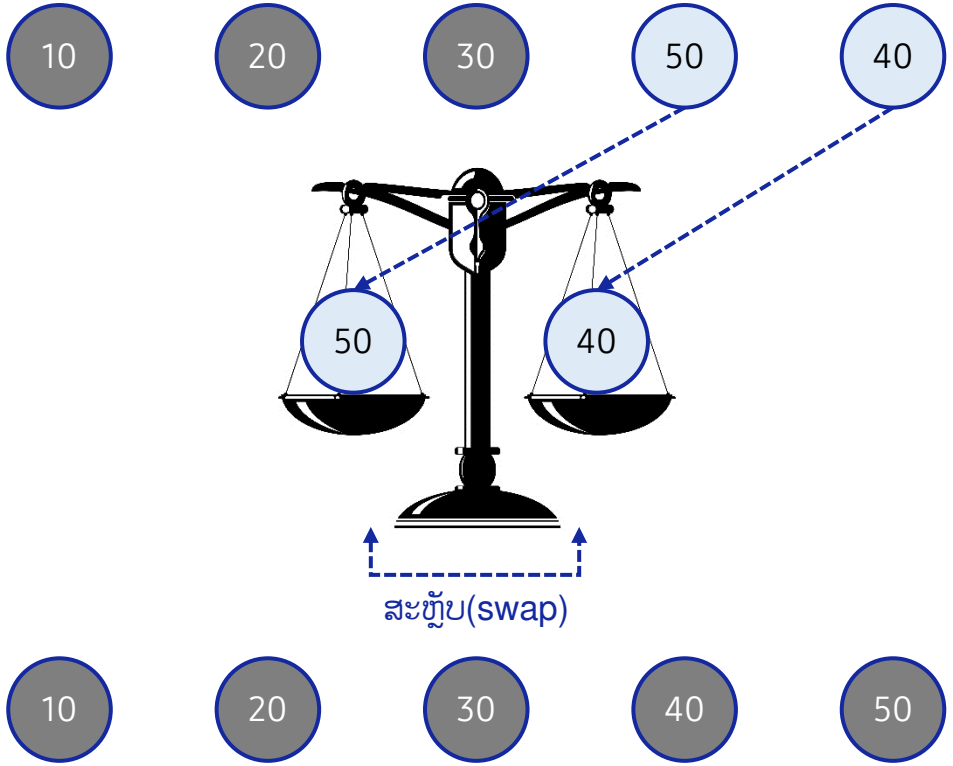
- ໂດຍການປະຕິບັດແບບເດີມ ການຄັດເລືອກດຳເນີນການຕ່ຳ ຕາມອົງປະກອບທີ່ຍັງບໍ່ໄດ້ຄັດເລືອກ, ຍົກເວັ້ນລາຍການຄັດເລືອກແລ້ວ.



# 2. ການຈັດລຽງແບບ Selection

## 2.2. ຂະບວນການຈັດລຽງແບບ Selection

ການຄັດເລືອກດຳເນີນການຕໍ່ ໃນອົງປະກອບທີ່ຍັງບໍ່ໄດ້ຄັດເລືອກໄປຕາມລຳດັບຈົນສຳເລັດ.



# 3. ການຈັດລຽງແບບ Insertion

## 3.1. ຕົວຢ່າງ ການຈັດລຽງແບບ Insertion

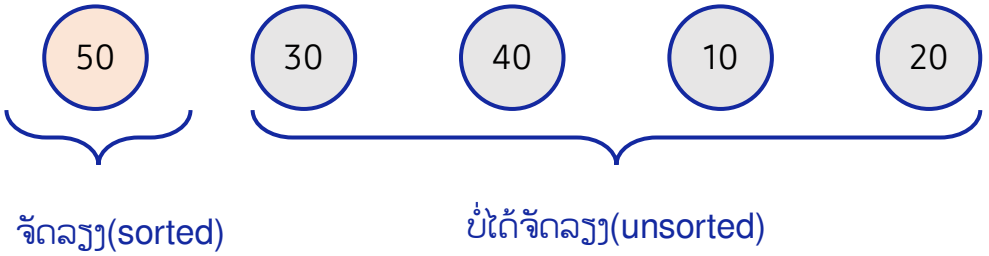
ການຈັດລຽງແບບ Insertion ແມ່ນວິທີການຈັດລຽງທີ່ເພີ່ມອົງປະກອບຈາກລາຍການທີ່ບໍ່ໄດ້ຈັດລຽງ ໄປຫາລາຍການທີ່ຈັດລຽງເທື່ອລະອັນ. ເພື່ອດຳເນີນການ ລັກສະນະດັ່ງກ່າວ, ຈະໃຊ້ຫຼັກການຊອກຫາຕຳແໜ່ງທີ່ຈະເພີ່ມ ໂດຍການປຽບທຽບອົງປະກອບທີ່ຈະເພີ່ມກັບສ່ວນທີ່ເຫຼືອຂອງລາຍການທີ່ຈັດລຽງຕາມລຳດັບ.

**ຕົວຢ່າງ** ສົມມຸດວ່າພວກເຮົາຕ້ອງການທີ່ຈະຈັດລຽງລຳດັບຂໍ້ມູນໃນລາຍການນີ້ [50, 30, 40, 10, 20] ໃຫ້ປະຕິບັດດັ່ງຕໍ່ໄປນີ້.

# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

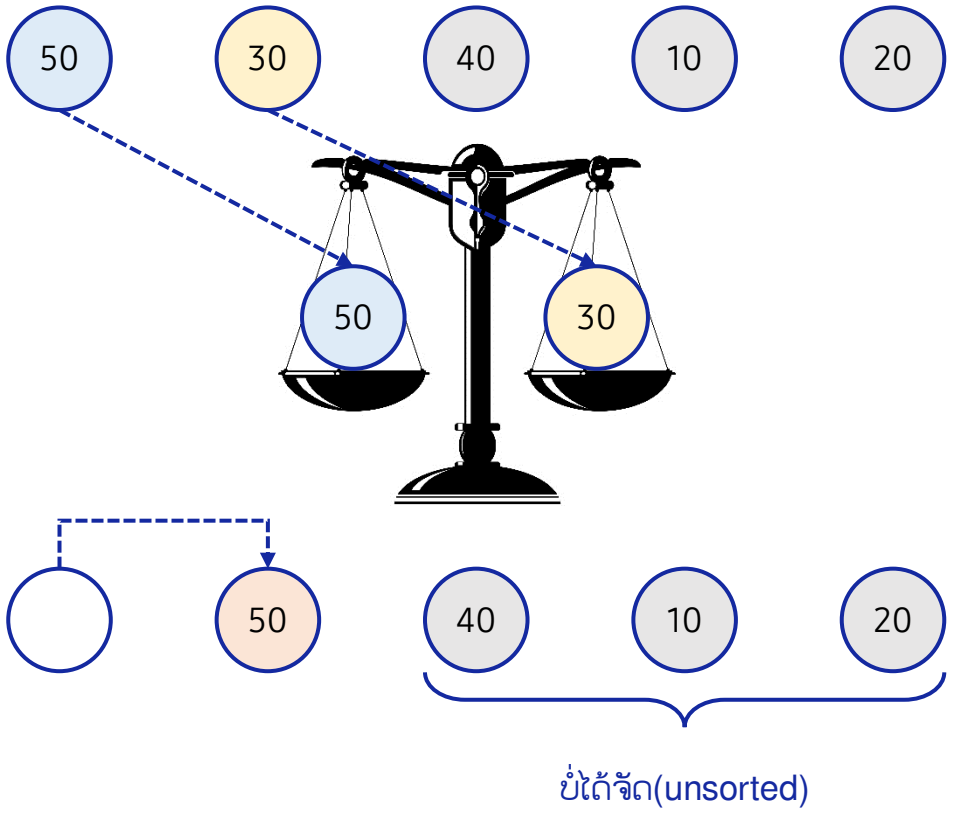
ເລີ່ມຈາກ, ອົງປະກອບທີ 1 ທີ່ມີຄ່າເທົ່າກັບ 50 ຖືວ່າເປັນລາຍການຄັດເລືອກຮອບທີ 1 ເລີຍ.

# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

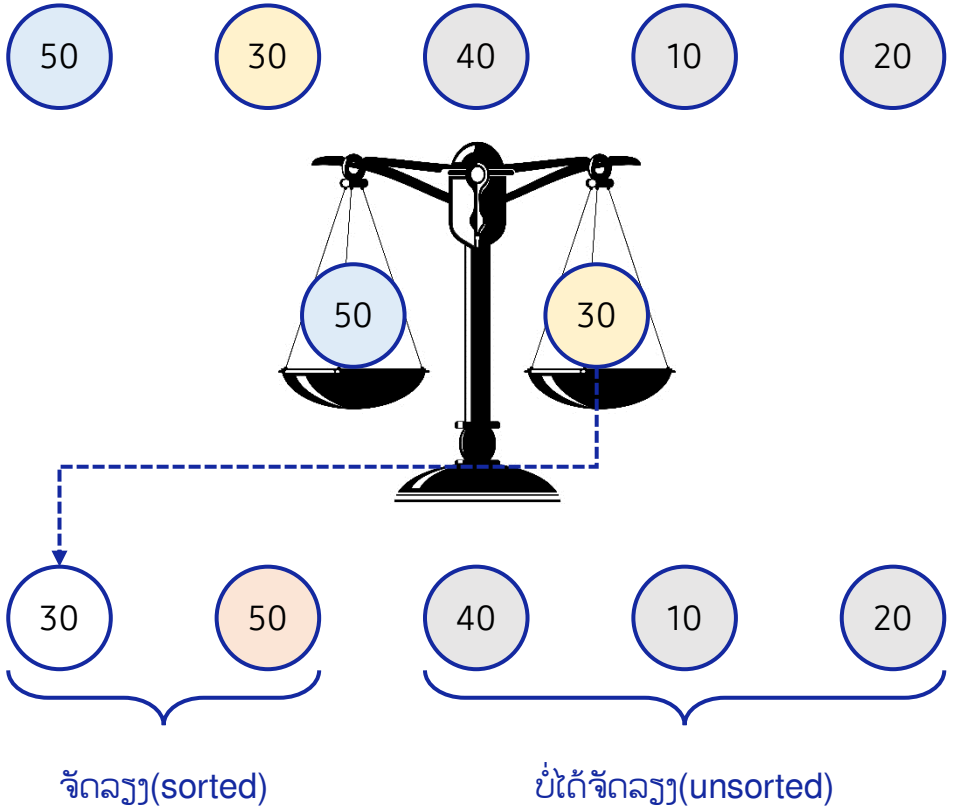
ເນື່ອງຈາກອົງປະກອບທີ 2 ມີຄ່າເທົ່າກັບ 30 ແມ່ນນ້ອຍກວ່າ 50 ຕາມການຈັດລຽງຄັ້ງທີ 2 , 50 ຈຶ່ງຖືກເລື່ອນມາທາງດ້ານຂວາ.



# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

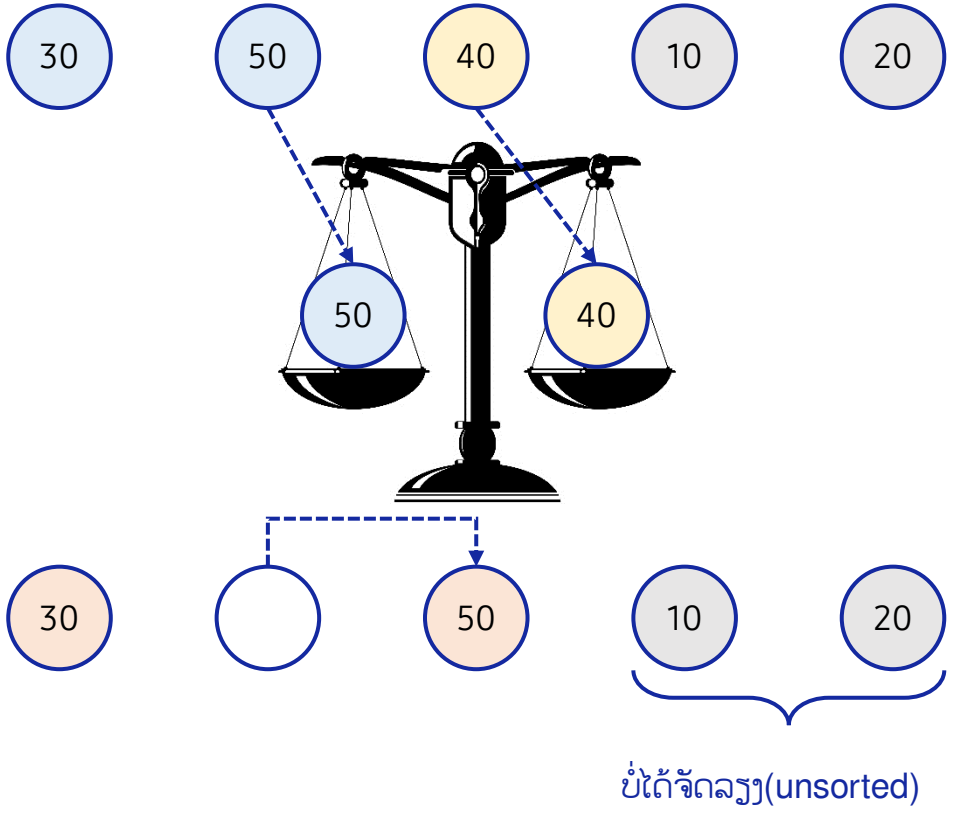
- ຫຼັງຈາກເລື່ອນ 50 ໄປຕຳແໜ່ງທີ 2, 30 ຖືກເອົາມາວາງໄວ້ໃນຕຳແໜ່ງທີ 1 ແທນບ່ອນ 50 ເພາະວ່າບໍ່ມີອົງປະກອບອື່ນທີ່ຈະປຽບທຽບ



# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

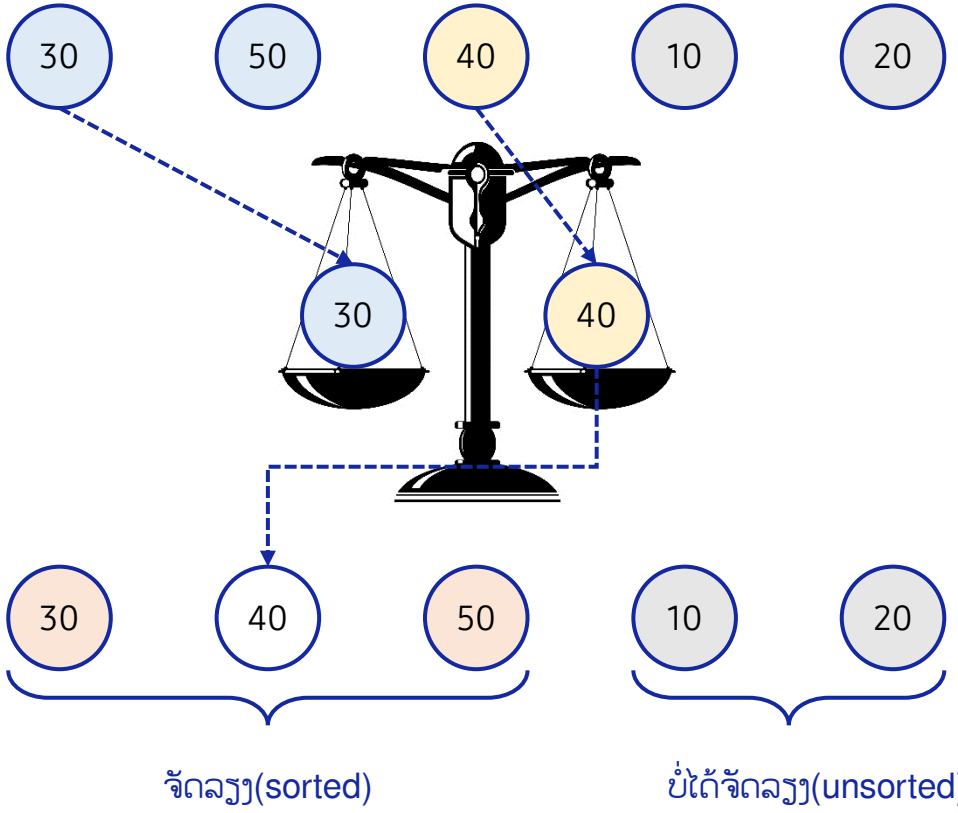
ດຳເນີນການຕໍ່ໄປ, ເນື່ອງຈາກອົງປະກອບທີ 3 40 ນ້ອຍກວ່າ 50 ໃນລາຍການຈັດລຽງລຳດັບ, 50 ຖືກຍ້າຍໄປທາງຂວາໜຶ່ງຕຳແໜ່ງ



# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

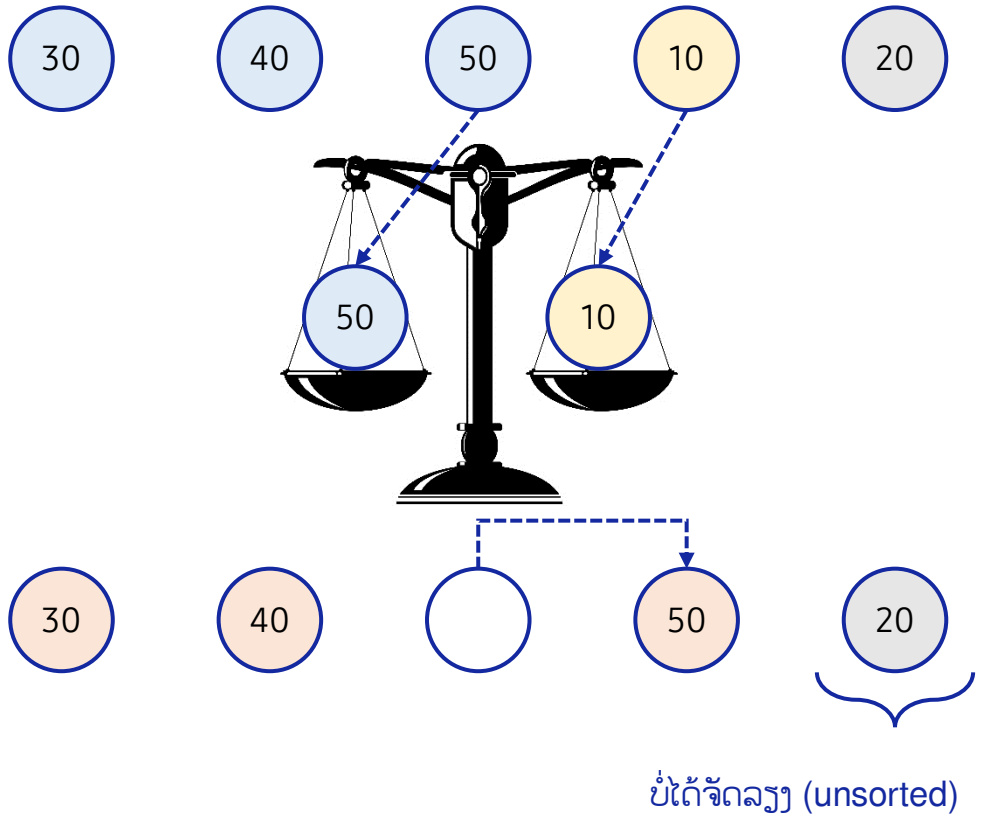
ຕໍ່ໄປ ໃຫ້ປຽບທຽບ ຕຳແໜ່ງທີ 1 ທີ່ມີຄ່າເທົ່າກັບ 30 ກັບ ຕຳແໜ່ງທີ 2 ທີ່ມີຄ່າເທ່າກັບ 40 ພາຍໃນລາຍການທີ່ຖືກຈັດລຽງແລ້ວ, ເນື່ອງຈາກ 30 ນ້ອຍກວ່າ 40, 40 ຈຶ່ງຖືກຈັດຢູ່ຕຳແໜ່ງເກົ່າ.



# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

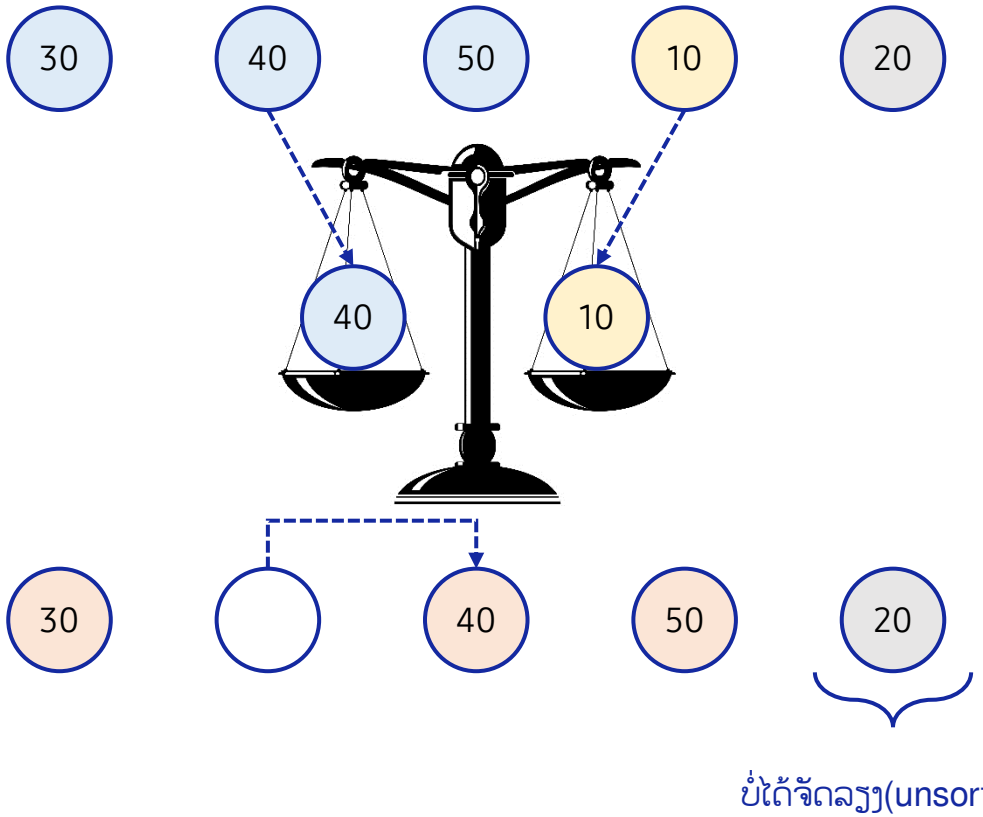
ໃນການຈັດລຽງຮອບທີ 2, ເນື່ອງຈາກອົງປະກອບກ່ອນໜ້າແມ່ນ 50 ໃຫຍ່ກວ່າ 10, 50 ຈຶ່ງຖືກເລື່ອນໄປທາງດ້ານຂວາຕຳແໜ່ງໜຶ່ງ.



# 3. ภามจัดล฽งแบบ Insertion

## 3.2. ขะบอมภามจัดล฽งแบบ Insertion

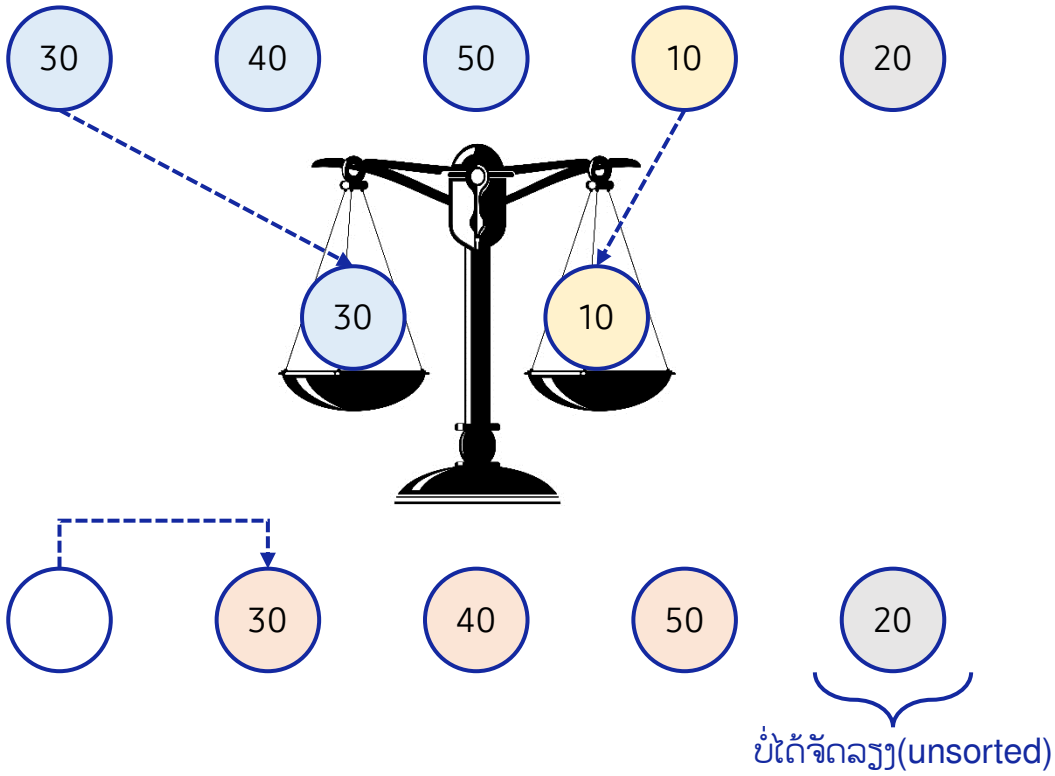
ในลายภามจัดล฽ง, จาภามจัดล฽งที่ผ่ามมา 40 ใหย่กว่า 10, 40 จึ่งถืกเลื่อมไปทาง้ามขอาต๋าแหฺ่งหนึ่ง.



# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

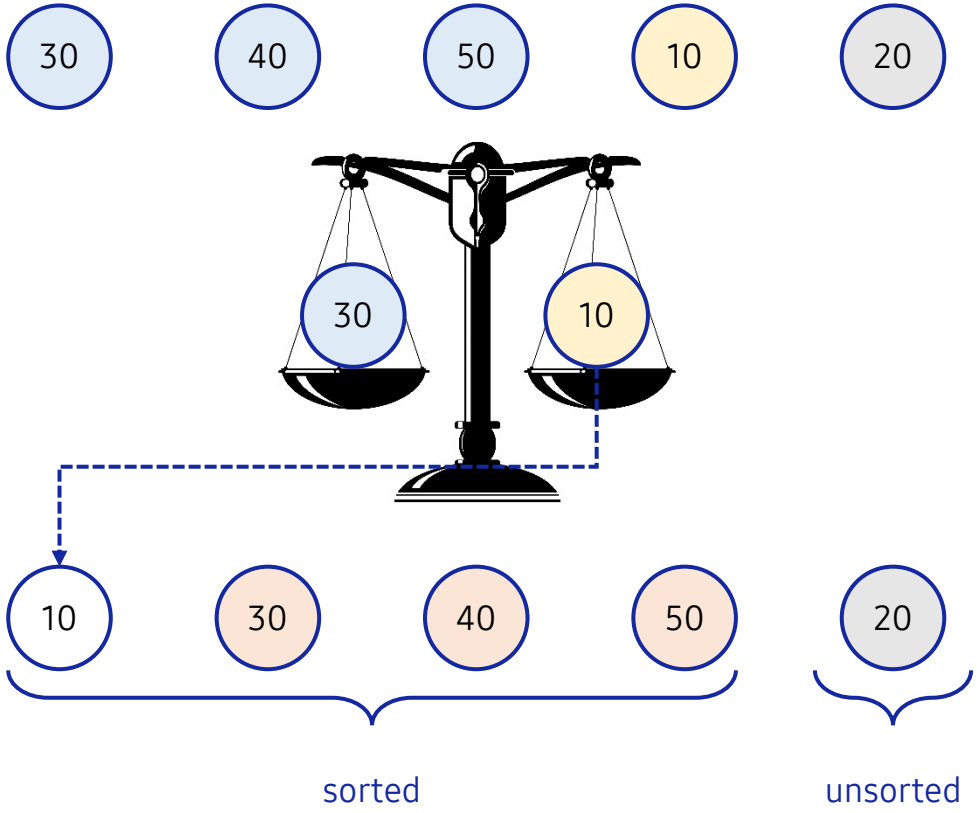
ໃນລາຍການຈັດລຽງ, ອົງປະກອບຜ່ານມາແມ່ນ 30 ໃຫຍ່ກວ່າ 10, 30 ຈຶ່ງຖືກເລື່ອນໄປທາງດ້ານຂວາ.



# 3. ການຈັດລຽງແບບ Insertion

## 3.2. . ຂະບວນການຈັດລຽງແບບ Insertion

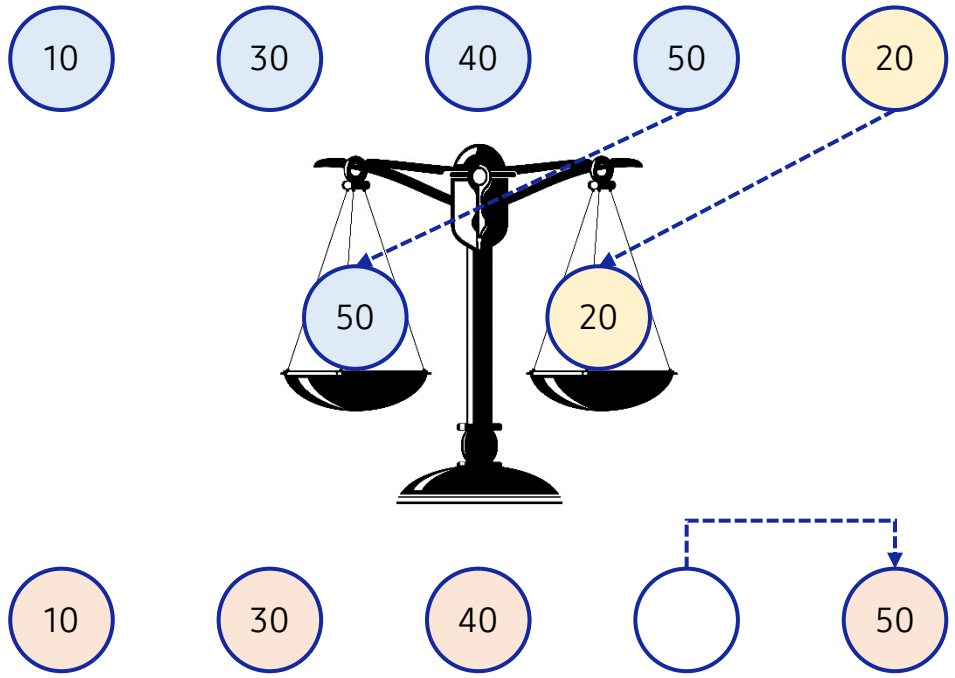
- 10 ໄດ້ຖືກເອົາແຊກເຂົ້າໃສ່ຕຳແໜ່ງທຳອິດ ເນື່ອງຈາກວ່າບໍ່ມີອົງປະກອບທີ່ຈະປຽບທຽບຕໍ່ໄປແລ້ວ



# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

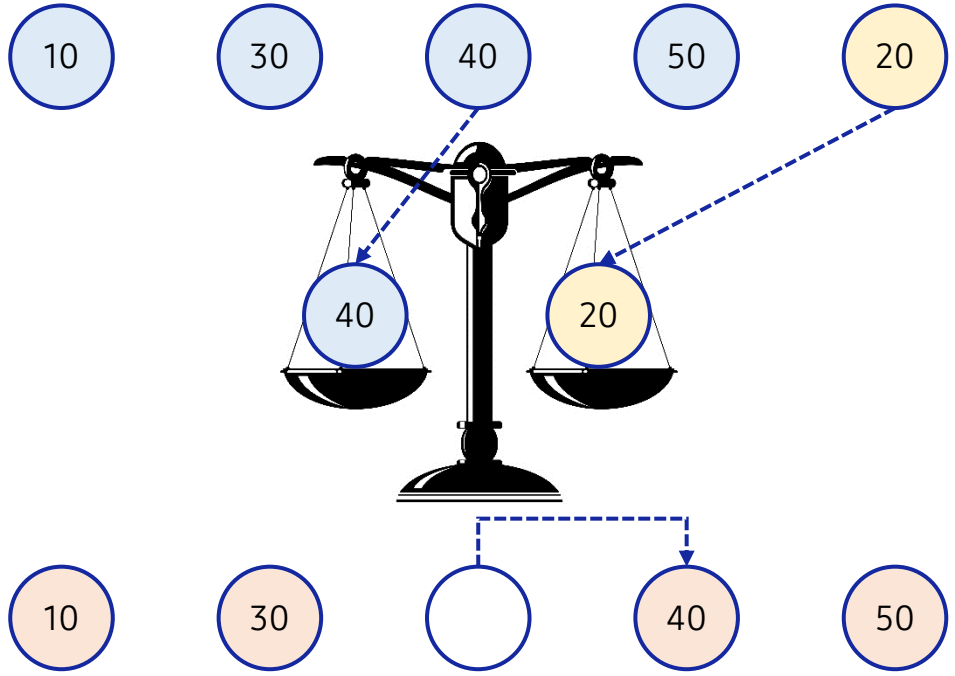
ເມື່ອເຖິງອົງປະກອບສຸດທ້າຍ 20 ນ້ອຍກວ່າ 50 ໃນລາຍການທີ່ຖືກຈັດລຽງ, 50 ຈຶ່ງຖືກເລື່ອນໄປທາງດ້ານຂວາ.



# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

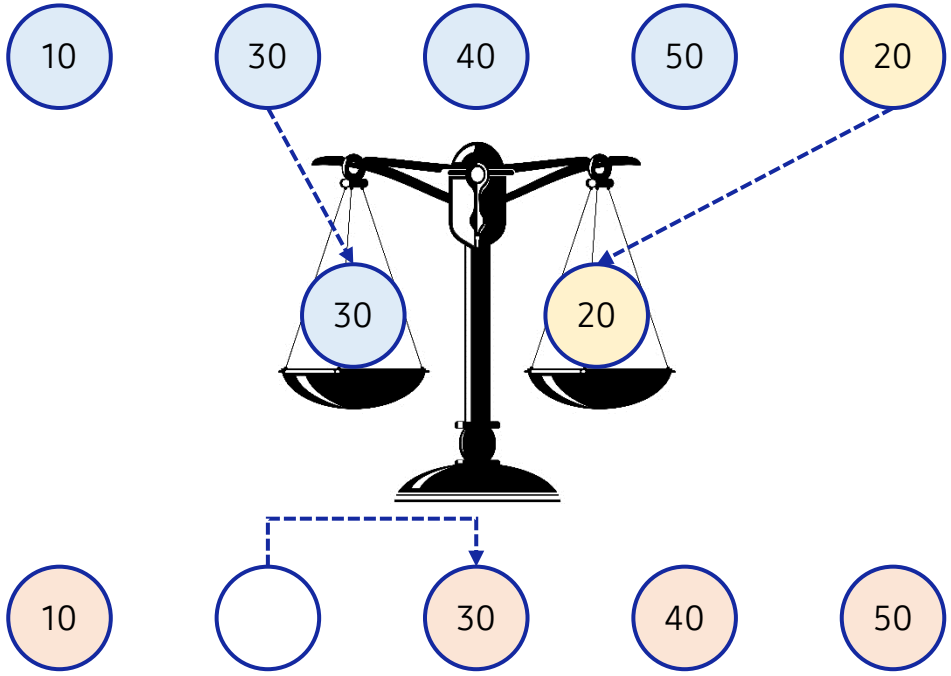
ໃນລາຍການທີ່ຖືກຈັດລຽງແລ້ວ, ເນື່ອງຈາກວ່າຕຳແໜ່ງຕໍ່ລຶງໄປແມ່ນມີຄ່າເທົ່າກັບ 40  ທີ່ໃຫຍ່ກວ່າ 20, 40 ຈຶ່ງຖືກເລື່ອນໄປທາງຂວາໜຶ່ງຕຳແໜ່ງ.



# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

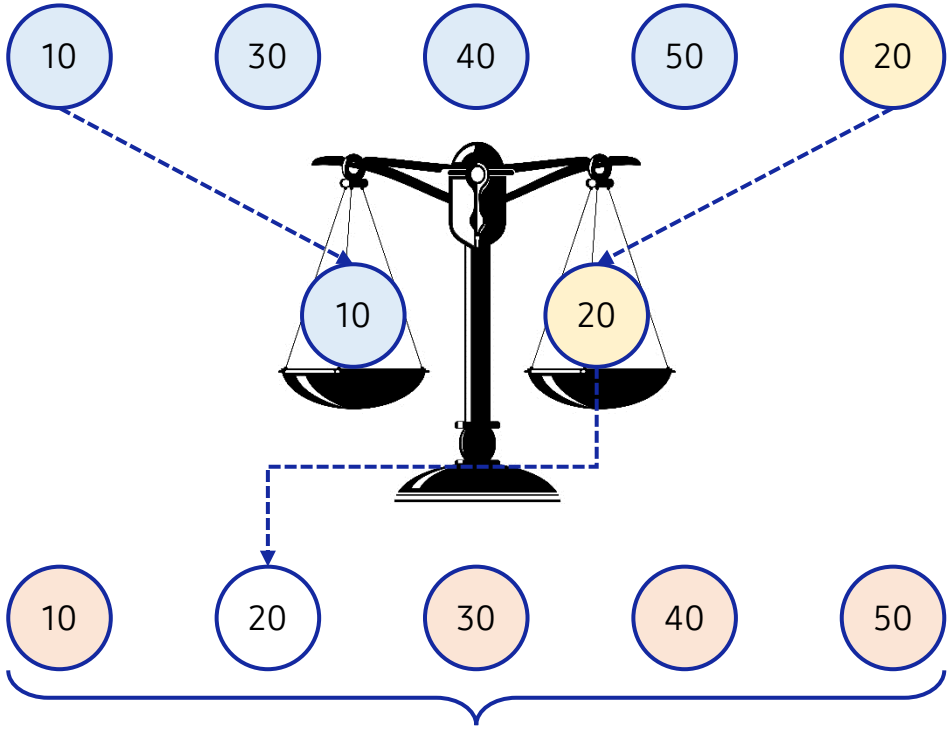
ໃນລາຍການຈັດລຽງແລ້ວ, ເນື່ອງຈາກກ່ອນໜ້ານີ້ອົງປະກອບ 30 ໃຫຍ່ກວ່າ 20, 30 ຈຶ່ງຖືກເລື່ອນໄປທາງດ້ານຂວາໜຶ່ງຕຳແໜ່ງ.



# 3. ການຈັດລຽງແບບ Insertion

## 3.2. ຂະບວນການຈັດລຽງແບບ Insertion

ພາຍໃນລາຍການຈັດລຽງທີ່ຜ່ານມາ, 10 ນ້ອຍກວ່າ 20, 20 ຈຶ່ງຢູ່ຕຳແໜ່ງທີ່ຖືກຕ້ອງ. ສັງເກດເຫັນວ່າ ລາຍການທັງຫມົດແມ່ນທີ່ຖືກຈັດລຽງແລ້ວ.



# Let’s code

# 1. ການສະຫຼັບຕຳແໜ່ງ

## 1.1. ສະຫຼັບ: ສັບປ່ຽນບ່ອນຢູ່ຂອງສອງອົງປະກອບ

Algorithm ການຈັດລຽງປະກອບມີພື້ນຖານການດຳເນີນການຢູ່ 2 ຢ່າງ ໄດ້ແກ່  ການປຽບທຽບ ແລະ ການປ່ຽນບ່ອນ. ໂດຍທີ່ການສະຫຼັບຕຳແໜ່ງນັ້ນສາມາດສ້າງຟັງຊັນການເຮັດວຽກດັ່ງລຸ່ມນີ້.

```
1 def swap1(S, x, y):
2     t = S[x]
3     S[x] = S[y]
4     S[y] = t
```

```
5 5 = [10, 20]
6 6rint(S)
7 7wap1(S, 0, 1)
8 8rint(S)
```

```
[10, 20]
[20, 10]
```

**Line 1 4**

- ฟังຊັน swap()ใช้ลายการข้อมูน S ส่วน ตำแหน่ง x และ y ของลายการอิงประกอบที่ป้อนเข้ามา.
- จัดเก็บข้อมูนใน S[x] ในตัวแปรชื่อถาว t และ ข้อมูนตำแหน่ง S[y] ไปเป็น S[x].
- ถ่าข้อมูนใน S[x] จัดเก็บในตัวแปรชื่อถาว t ก่อนย้ายไปเก็บที่ S[y].

# 1.ການສະຫຼັບຕຳແໜ່ງ

## 1.2. ການສະຫຼັບຕຳແໜ່ງໃນພາສາ Python

ໃນພາສາ Python, ເຮົາສາມາດຈັດເກັບຄ່າຂໍ້ມູນໄວ້ໃນຫຼາຍຕົວປ່ຽນໃນເວລາດຽວກັນ, ສະນັ້ນ ສາມາດສະຫຼັບຄ່າຕົວປ່ຽນດັ່ງກ່າວດ້ວຍວິທີພື້ນຖານຕໍ່ໄປນີ້.

```
1  def swap2(S, x, y):
2      S[x], S[y] = S[y], S[x]
```

```
3  S = [10, 20]
4  print(S)
5  swap2(S, 0, 1)
6  print(S)
```

```
[10, 20]
[20, 10]
```

**Line 1 2**

- ທົດແທນຄ່າຂອງ S[y] ແລະ S[x] ເປັນ S[x] ແລະ S[y], ຕາມລຳດັບ.
- ໃນເວລານີ້, ຄວນສັງເກດວ່າສອງຄ່າສາມາດແລກປ່ຽນໄດ້ເພາະວ່າຄ່າທີ່ຢູ່ເບື້ອງຂວາຈະຖືກປ່ຽນເປັນ tuple ແລະ ຖືກມອບໝາຍໃຫ້ຄ່າຢູ່ເບື້ອງຊ້າຍ..

# 2. ການຈັດລຽງແບບ Bubble

## 2.1. ຂັ້ນຕອນປະຕິບັດສ້າງການຈັດລຽງແບບ Bubble

ຈັດລຽງແບບ Bubble ແມ່ນ Algorithm ທີ່ໃຊ້ປຽບທຽບສອງອົງປະກອບທີ່ມີຕຳແໜ່ງຢູ່ຕິດກັນ ໂດຍການຫຼັບຄ່າໄປເລື້ອຍໆ ດັ່ງຕ່ໍໄປນີ້.

```
1 def bubblesort(S):
2     n = len(S)
3     for i in range(n):
4         print(S)
5         for j in range(n - 1):
6             if S[j] > S[j + 1]:
7                 S[j], S[j + 1] = S[j + 1], S[j]
```

**Line 3 7**

- ສຳຫຼັບທຸກໆຄ່າຂອງ i, າຂອງອົງປະກອບທີ່ຢູ່ໃກ້ຄຽງແມ່ນປຽບທຽບສຳລັບອົງປະກອບ j ທາງດ້ານຂວາຂອງ i
- ຖ້າສອງອົງປະກອບທີ່ຢູ່ຕິດກັນບໍ່ຖືກຕ້ອງ, ສອງຄ່າຈະຖືກສະຫຼັບໄປຈົນເຖິງຄູ່ສຸດທ້າຍ.

# 2. ການຈັດລຽງແບບ Bubble

## 2.2. ຜົນການເຮັດວຽກ ການຈັດລຽງແບບ Bubble

ຈັດລຽງແບບ Bubble ແມ່ນ ອານກໍຣິດທຶມທີ່ໃຊ້ປຽບທຽບສອງອົງປະກອບທີ່ມີຕຳແໜ່ງຢູ່ຕິດກັນ ໂດຍການຫຼັບຄ່າໄປເລື້ອຍໆ ດັ່ງຕ່ໄປນີ້.

```
1  def bubblesort(S):
2      n = len(S)
3      for i in range(n):
4          print(S)
5          for j in range(n - 1):
6              if S[j] > S[j + 1]:
7                  S[j], S[j + 1] = S[j + 1], S[j]
```

```
8  S = [50, 30, 40, 10, 20]
9  bubblesort(S)
10 print(S)
```

```
[50, 30, 40, 10, 20]
[30, 40, 10, 20, 50]
[30, 10, 20, 40, 50]
[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50]
```

**Line 4**

- ເພື່ອກວດສອບການເຮັດວຽກຂອງການຈັດລຽງແບບ bubble, ສະຖານະຂອງ S ແມ່ນຜົນໄດ້ຮັບທຸກໆຄັ້ງທີ i-th ຖືກວົນຮອບປະຕິບັດ.

# 3. ການຈັດລຽງແບບ Selection

## 3.1. ຂັ້ນຕອນປະຕິບັດສ້າງການຈັດລຽງແບບ Selection ໂດຍບໍ່ມີການສະສັບ ປ່ຽນຕຳແໜ່ງ

ການຈັດລຽງແບບ Selection ແມ່ນວິທີການທີ່ເລືອກຄ່າທີ່ນ້ອຍທີ່ສຸດຈາກລາຍການ ແລະ ເພີ່ມມັນເຂົ້າໄປໃນລາຍການຄັດເລືອກດັ່ງຕໍ່ໄປນີ້.

```
1  def selectionsort1(S):
2      R = []
3      while len(S) > 0:
4          print(R, S)
5          smallest = S.index(min(S))
6          R.append(S[smallest])
7          S.pop(smallest)
8      return R
```

**Line 1 8**

- ໃຊ້ລາຍການທີ່ບໍ່ໄດ້ທັນຈັດລຽງໃນ S ເປັນພາລາແມັດເຕີປ້ອນເຂົ້າ ແລະ ສົ່ງກັບອອກມາເປັນລາຍການທີ່ຍັນຈຸໄວ້ໃນ R.
- ຄົ້ນຫາອົງປະກອບທີ່ໃຫຍ່ທີ່ສຸດໃນລາຍການ S, ລຶບອອກ, ເພີ່ມອົງປະກອບທີ່ລຶບອອກມາໄວ້ຢູ່ລາຍການ R, ແລະ ທວນຊ້ຳ ຈົນກວ່າ S ຈະບໍ່ມີອົງປະກອບຈັກຕົວ(ຄ່າຫວ່າງເປົ່າ).

# 3. ການຈັດລຽງແບບ Selection

## 3.2. ຜົນການເຮັດວຽກ ການຈັດລຽງແບບ Selectionໂດຍບໍ່ມີການສະສັບປ່ຽນຕຳແໜ່ງ

1 ກວດສອບການດຳເນີນການຂອງຂັ້ນຕອນການຄັດເລືອກ. ລາຍການອົງປະກອບຖືກເພີ່ມໃສ່ R ຕາມລຳດັບຂອງອົງປະກອບທີ່ນ້ອຍທີ່ສຸດໃນ S.

```
1  def selectionsort1(S):
2      R = []
3      while len(S) > 0:
4          print(R, S)
5          smallest = S.index(min(S))
6          R.append(S[smallest])
7          S.pop(smallest)
8      return R
```

```
9  S = [50, 30, 40, 10, 20]
10 R = selectionsort1(S)
11 print(R)
```

```
[] [50, 30, 40, 10, 20]
[10] [50, 30, 40, 20]
[10, 20] [50, 30, 40]
[10, 20, 30] [50, 40]
[10, 20, 30, 40] [50]
[10, 20, 30, 40, 50]
```

**Line 4**

- ເພື່ອກວດສອບການດຳເນີນການຂອງການຄັດເລືອກ, ສະຖານະຂອງ S ແມ່ນຜົນໄດ້ຮັບທຸກຄັ້ງໃນຂະນະທີ່ loop ຖືກປະຕິບັດ.

# 3. ການຈັດລຽງແບບ Selection

## 3.3. ການປະຕິບັດ In-Place ຂອງການຈັດລຽງແບບຄັດເລືອກ

- In-Place ໝາຍເຖິງຂັ້ນຕອນການຈັດລຽງ ທີ່ຈັດລຽງຜ່ານການປະຕິບັດການສະຫຼັບໂດຍບໍ່ຕ້ອງໃຊ້ໜ່ວຍຄວາມຈຳເພີ່ມເຕີມ. selectionsort1() ແມ່ນ ອາມໍຣິດທຶມ ທີ່ປະຕິບັດກ່ອນໜ້ານີ້ໃຊ້ຫນ່ວຍຄວາມຈຳ R ເພີ່ມເຕີມ, ດັ່ງນັ້ນ ຈຶ່ງບໍ່ແມ່ນການຈັດລຽງ In-Place.
- ເພື່ອດຳເນີນຂັ້ນຕອນການຈັດລຽງ In-Place algorithm ສຳຫຼັບການຈັດລຽງແບບ Selection, ຂັ້ນຕອນການເຮັດວຽກໂດຍໃຊ້ການສະຫຼັບ ດັ່ງທີ່ສະແດງກ່ອນໜ້ານີ້.

```
1  def selectionsort2(S):
2      n = len(S)
3      for i in range(n - 1):
4          print(S)
5          smallest = i
6          for j in range(i + 1, n):
7              if S[j] < S[smallest]:
8                  smallest = j
9          S[i], S[smallest] = S[smallest], S[i]
```

**Line 1 9**

- **Line** 6-8, ຊອກຫາຕຳແໜ່ງຂອງອົງປະກອບທີ່ນ້ອຍທີ່ສຸດ ຫຼັງຈາກອົງປະກອບ i-th.
- **Line** 9, ອົງປະກອບ i-th ແລະ ອົງປະກອບຂອງຕຳແໜ່ງນ້ອຍທີ່ສຸດ ຖືກສັບປ່ຽນ.
- ພາຍນອກ for-loop, i ແມ່ນວົນຮອບຈາກອົງປະກອບທຳອິດຂອງລາຍການ ໄປຫາອົງປະກອບສຸດທ້າຍ.

# 3. ການຈັດລຽງແບບ Selection

## 3.4. ຜົນການເຮັດວຽກ In-Place ຂອງການຈັດລຽງແບບ Selection

ກວດສອບການດຳເນີນງານ ຂອງຂັ້ນຕອນການຄັດເລືອກ. ອົງປະກອບຖືກຈັດລຽງຕາມລຳດັບຂອງອົງປະກອບທີ່ນ້ອຍທີ່ສຸດ ໂດຍບໍ່ຕ້ອງໃຊ້ໜ່ວຍຄວາມຈຳເພີ່ມເຕີມ

```
1  def selectionsort2(S):
2      n = len(S)
3      for i in range(n - 1):
4          print(S)
5          smallest = i
6          for j in range(i + 1, n):
7              if S[j] < S[smallest]:
8                  smallest = j
9          S[i], S[smallest] = S[smallest], S[i]
```

```
10 S = [50, 30, 40, 10, 20]
11 selectionsort2(S)
12 print(S)
```

```
[50, 30, 40, 10, 20]
[10, 30, 40, 50, 20]
[10, 20, 40, 50, 30]
[10, 20, 30, 50, 40]
[10, 20, 30, 40, 50]
```

**Line 4**

- ເພື່ອກວດສອບການເຮັດວຽກຂອງການຄັດເລືອກ, ສະຖານະຂອງ S ແມ່ນພິມໄດ້ຮັບທຸກໆຄັ້ງທີ່ i-th loop ຖືກປະຕິບັດ.

# 4. ການຈັດລຽງແບບ Insertion

## 4.1. ການປະຕິບັດ ຈັດລຽງແບບ Insertionແມ່ນໃຊ້ຫນ່ວຍຄວາມຈຳເພີ່ມ

ການຈັດລຽງແບບ Insertion ແມ່ນວິທີການຈັດລຽງທີ່ເພີ່ມອົງປະກອບໃຫມ່ໃຫ້ກັບຕຳແໜ່ທີ່ຖືກຕ້ອງໃນລາຍການທີ່ຈັດລຽງ, ດັ່ງສະແດງໃນ code ຕໍ່ໄປນີ້.

```
1  def insertionsort1(S):
2      n = len(S)
3      R = []
4      while len(S) > 0:
5          print(R, S)
6          x = S.pop(0)
7          j = len(R) - 1
8          while j >=0 and R[j] > x:
9              j -= 1
10         R.insert(j + 1, x)
11     return R
```

**Line 1 11**

- ນຳໃຊ້ລາຍການທີ່ຍັງບໍ່ທັນໄດ້ຈັດລຽງ S ເປັນພາລາເມເຕີທີ່ສົ່ງເຂົ້າໄປ ແລະ ສົ່ງກັບອອກມາໃນລາຍການອົງປະກອບທີ່ຖືກຈັດບລຽງແລ້ວຄື R.
- ເອົາອົງປະກອບທີ່ຢູ່ໃນລາຍການ S ອອກມາທີລະ 1, ເລື່ອນໄປຂ້າງໜ້າ ຈົນເຖິງຕົວສຸດທ້າຍຂອງອົງປະກອບໃນ R, ແລະ ເພີ່ມລົງໄປໃນຕຳແໜ່ງທີ່ຕ້ອງການແຊກເພີ່ມ.

# 4. ການຈັດລຽງແບບ Insertion

## 4.2. ຜົນການເຮັດວຽກການຈັດລຽງແບບ Insertion ໂດຍການໃຊ້ໜ່ວຍຄວາມຈຳເພີ່ມ

ກວດສອບຂັ້ນຕອນການເຮັດວຽກຂອງການຈັດລຽງແບບແຊກເພີ່ມ. ຈາກການເພີ່ມອົງປະກອບໃນລາຍການ S ໄປຫາ R ຕາມລຳດັບການຈັດລຽງ.

```
1  def insertionsort1(S):
2      n = len(S)
3      R = []
4      while len(S) > 0:
5          print(R, S)
6          x = S.pop(0)
7          j = len(R) - 1
8          while j >=0 and R[j] > x:
```

```
9  S = [50, 30, 40, 10, 20]
10 insertionsort1(S)
11 print(R)
```

```
[] [50, 30, 40, 10, 20]
[50] [30, 40, 10, 20]
[30, 50] [40, 10, 20]
[30, 40, 50] [10, 20]
[10, 30, 40, 50] [20]
[10, 20, 30, 40, 50]
```

**Line5**

- ໃນຄຳສັ່ງເພື່ອກວດກາເບິ່ງການດຳເນີນງານຂອງການຈັດລຽງແບບແຊກເພີ່ມ, ສະຖານະຂອງ S ແມ່ນຜົນໄດ້ຮັບທຸກຄັ້ງທີ່ while loop ຖືກປະຕິບັດ

# 4. ການຈັດລຽງແບບ Insertion

## 4.3. ການປະຕິບັດ In-Place ຂອງການຈັດລຽງແບບ Insertion

. ເພື່ອປະຕິບັດການຈັດລຽງການແຊກເພີ່ມ in-Place ເປັນ algorithm ຈັດລຽງ ທີ່ສາມາດປະຕິບັດໃນລະຫວ່າງອົງປະກອບ ເຄື່ອນຍ້າຍເທື່ອລະ1 ດັ່ງທີ່ສະແດງ ກ່ອນໜ້ານີ້.

```
1  def insertionsort2(S):
2      n = len(S)
3      for i in range(1, n):
4          print(S)
5          x = S[i]
6          j = i - 1
7          while j >= 0 and S[j] > x:
8              S[j + 1] = S[j]
9              j -= 1
10         S[j + 1] = x
```

**Line 1 10**

- **Line** 5, ອົງປະກອບ i-th ຖືກເກັບໄວ້ຊື່ວ່າx ໃນ x.
- ໃນແຖວ 6-9, j ຫຼຸດຄ່າລົງ ລະຫວ່າງການເຄື່ອນຍ້າຍອົງປະກອບ ຈົນກວ່າຈະພົບອົງປະກອບທີ່ໃຫຍ່ທີ່ສຸດໃນ x, ຫຼັງຈາກນັ້ນຈັດເກັບລົງໃນ x ໃນຕຳແໜ່ງທີ່ຖືກຕ້ອງຕາມຕ້ອງການ.
- ສ່ວນພາຍນ for-loop, i ເຮັດວຽກຮອບ ຈາກອົງປະກອບທີ1 ຈົນເຖິງອົງປະກອບຕົວສຸດທ້າຍ.

# 4. ການຈັດລຽງແບບ Insertion

## 4.4. ຜົນການດຳເນີນງານ In-Place ຂອງການຈັດລຽງແບບແຊກເພີ່ມ

ກວດເບິ່ງການເຮັດວຽກຂອງຂັ້ນຕອນການຈັດລຽງ, ໂດຍການວາງອົງປະກອບຂອງ S ໃນຕຳແໜ່ງທີ່ຖືກຕ້ອງໂດຍການຈັດລຽງແຊກເພີ່ມ.

```
1  def insertionsort2(S):
2      n = len(S)
3      for i in range(1, n):
4          print(S)
5          x = S[i]
6          j = i - 1
7          while j >= 0 and S[j] > x:
8              S[j + 1] = S[j]
```

```
9  S = [50, 30, 40, 10, 20]
10 insertionsort2(S)
11 print(S)
```

```
[50, 30, 40, 10, 20]
[30, 50, 40, 10, 20]
[30, 40, 50, 10, 20]
[10, 30, 40, 50, 20]
[10, 20, 30, 40, 50]
```

**Line 4**

- ເພື່ອກວດເບິ່ງການເຮັດວຽກຂອງການຈັດລຽງການແຊກເພີ່ມ, ສະຖານະຂອງ S ແມ່ນຜົນໄດ້ຮັບທຸກໆຄັ້ງທີ່ i-th ສຳລັບ loop ຖືກປະຕິບັດ.

# One More Step

- Time complexity ຂອງ algorithm ການລຽງລຳດັບແມ່ນຫຍັງ?
- Algorithm ການຈັດລຽງແບບ bubble ດຳເນີນການວົນຮອບຊ້ອນກັນສອງຄັ້ງ, ແລະ ຈຳນວນການປຽບທຽບສຳລັບແຕ່ລະຮອບແມ່ນ N – 1, N – 2, N – 3, … , 0.
- ຈຳນວນຂອງການປຽບທຽບໃນ algorithm ການຈັດລຽງແບບ bubble ແມ່ນ $T(N) = (N – 1) + (N - 2) + … + 1 = N(N-1)/2$.
- Time complexity ການຈັດລຽງແບບ bubble ແມ່ນ $O(N^2)$.
- ສ່ວນການຈັດລຽງແບບ selection ປະຕິບັດວົນຮອບຊ້ອນກັນສອງຄັ້ງ ເຊັ່ນດຽວກັນກັບການຈັດລຽງແບບ bubble, Time complexity ແມ່ນ $O(N^2)$ ແບບດຽວກັນ
- ກໍລະນີການຈັດລຽງແບບ Insertion, ຈຳນວນຂອງການປຽບທຽບແບ່ງອອກເປັນສອງສ່ວນ ຄືສ່ວນທີ່ລຽງແລ້ວ ແລະ ສ່ວນທີ່ຍັງບໍ່ໄດ້ລຽງ.
- ໃນກໍລະນີສ່ວນທີ່ຈັດລຽງແລ້ວ, Time complexity ໃນການປະມວນຜົນແມ່ນ O(N) ເນື່ອງຈາກວ່າຕຳແໜ່ງຂອງການແຊກ ສາມາດເປັນໄປໄດ້ພຽງແຕ່ໜຶ່ງບ່ອນໃນແຕ່ລະຄັ້ງ.
- ກໍລະນີທີ່ບໍ່ດີທີ່ສຸດແມ່ນຄ່າໃນລາຍການປົ່ນກັນທັງໝົດ, Time complexity ແມ່ນ $O(N^2)$ ຄືກັນກັບການຈັດລຽງແບບ bubble/selection, ເນື່ອງຈາກວ່າຕຳແໜ່ງແຊກຕ້ອງຖືກປຽບທຽບກັບອົງປະກອບທັງໝົດຂອງລາຍການທີ່ຈັດລຽງສຳເລັດ.

# Pop quiz

## Q1. ການປະມວນຜົນສະຫຼັບຕຳແໜ່ງແມ່ນໄດ້ຖືກປະຕິບັດຈັກຄັ້ງ ໃນຂະບວນການຈັດລຽງແບບ bubble ຂ້າງລຸ່ມນີ້?

```
1  def bubblesort(S):
2      n = len(S)
3      for i in range(n):
4          print(S)
5          for j in range(n - 1):
6              if S[j] > S[j + 1]:
7                  S[j], S[j + 1] = S[j + 1], S[j]
```

```
1  S = [50, 30, 40, 10, 20]
2  bubblesort(S)
3  print(S)
```

```
[50, 30, 40, 10, 20]
[30, 40, 10, 20, 50]
[30, 10, 20, 40, 50]
[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50]
```

## Q2. ການດຳເນີນການປຽບທຽບແມ່ນໄດ້ຖືກປະຕິບັດຈັກຄັ້ງ ໃນຂະບວນການຈັດລຽງແບບ Insertion ຂ້າງລຸ່ມນີ້?

```
1  def insertionsort2(S):
2      n = len(S)
3      for i in range(1, n):
4          print(S)
5          x = S[i]
6          j = i - 1
7          while j >= 0 and S[j] > x:
8              S[j + 1] = S[j]
9              j -= 1
10         S[j + 1] = x
```

```
1  S = [50, 30, 40, 10, 20]
2  insertionsort2(S)
3  print(S)
```

```
[50, 30, 40, 10, 20]
[30, 50, 40, 10, 20]
[30, 40, 50, 10, 20]
[10, 30, 40, 50, 20]
[10, 20, 30, 40, 50]
```

# 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

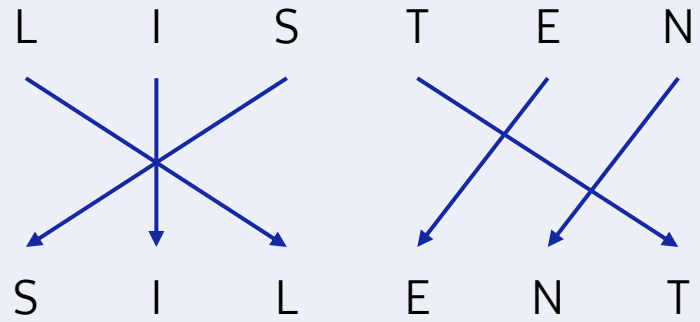
## ປ້ອງກັນການໂກງໃນການເຮັດວຽກຮ່ວມກັນ

ສິ່ງທ້າທາຍສໍາລັບຄູແມ່ນເພື່ອຊອກຫາວິທີທີ່ຈະປະເມີນຜົນໄດ້ຮັບຂອງບຸກຄົນ, ໃນຂະນະທີ່ນໍາໃຊ້ຜົນປະໂຫຍດຂອງການຮ່ວມມື. ຈະຮູ້ໄດ້ແນວໃດວ່ານັກຮຽນຕັ້ງໃຈເຮັດວຽກ ຫຼື ກົງແຮງງານຜູ້ຮ່ວມງານ? 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. ໃນສະພາບແວດລ້ອມການເຮັດວຽກຮ່ວມກັຍ, ນັກຮຽນມີສ່ວນຮ່ວມໃນການສົນທະນາ ປຶກສາຫາລືເຊິ່ງກັນແລະກັນ.

Q1. ກຳໜົດຄຳສັບໃຫ້ 2 ໜວດ, ຂຽນ algorithm ເພື່ອກຳນົດວ່າຄຳສັບເຫຼົ່ານີ້ແມ່ນ anagrams ຫຼືບໍ.
anagrams ແມ່ນຄຳທີ່ສ້າງຂຶ້ນໂດຍການຈັດລຽງຕົວອັກສອນຂອງຄຳທີ່ແຕກຕ່າງກັນ ໂດຍໃຊ້ຕົວອັກສອນຕົ້ນສະບັບ.(ຕົວຢ່າງ. LISTEN” ແລະ “SILENT” ແມ່ນຕົວຫຍໍ້.)



## Q2. ນຳໃຊ້ algorithm ການຈັດລຽງເພື່ອງ່າຍຕໍ່ການກຳໜົດວ່າແມ່ນ anagrams ຫຼືບໍ

- ສ້າງຟັງຊັນປະເມີນອານາແກຣມໂດຍໃຊ້ Python's built-in sorted() .
- ປັບໃຊ້ຟັງຊັນ selection_sort2() ເພື່ອສ້າງໜ້າທີ່ກຳໜົດ anagrams.
- ປັບໃຊ້ຟັງຊັນ insertion_sort2() ເພື່ອຟັງຊັນທີ່ກຳໜົດ anagrams.

```
1 print(is_anagram("listen", "silent"))
2 print(is_anagram("anagram", "nagaram"))
3 print(is_anagram("listen", "silence"))
4 print(is_anagram("anagram", "anagrams"))
5
```

```
True
True
False
False
```