Unit 28.

ภาษาจัดลงรูปแบบ Merge

Learning objectives

- ✓ ເຂົ້າໃຈຂັ້ນຕອນວິທີການຈັດລຽງແບບ Merge ແລະ ແກ້ໄຂບັນຫາໂດຍນຳໃຊ້ຂັ້ນຕອນວິທີການຈັດລຽງແບບ Merge.
- ✓ ເຂົ້າໃຈອະທິບາຍຄວາມແຕກຕ່າງ ລະຫວ່ງການຈັດລຽງແບບ merge ແລະ ການຈັດລຽງແບບ bubble , ການຈັດຮຽງແບບ Selection (selection), ແລະ ການຈັດລຽງແບບ Insertion.
- ✓ ຂ້ົາໃຈ ແລະ ສາມາດອະທິບາຍຄວາມສັບຊ້ອນຂອງເວລາທີ່ໃຊ້ໃນການຈັດລຽງແບບ merge.

Learning overview

- ✓ ສ້າງການຈັດລຽງແບບ merge ແບ່ງລາຍການທີ່ບໍ່ທັນຈັດລຽງ ,ຈັດລຽງແຕ່ລະອັນ, ແລະ ໂຮມ(merge)ລາຍການເຂົ້າກັນ.
- ✓ ສ້າງຟັງຊັນ merge ແປງສອງລາຍການຈັດລຽງໄປເປັນການຈັດລຽງລາຍການດຽວ.
- ✓ ເຂົ້າໃຈວ່າ ການຈັດລຽງແບບ merge ແມ່ນ divide-and-conquer ໂດຍນຳໃຊ້ຟັງຊັນ recursive.

Concepts you will need to know from previous units

- ✓ ນຳໃຊ້ຕົວປະຕິບັດການເພື່ອປຽບທຽບສອງຈຳນວນ.
- ✓ ແບ່ງບັນຫາອອກເປັນສ່ວນຍ່ອຍ ແລະ ນຳໃຊ້ຟັງຊັນເອີ້ນຕົວເອງ recursive.
- ✓ ນຳໃຊ້ສູດຄິດໄລ່ Big O ເພື່ອຄຳນວນ Time Complexity.

Keywords

- Sorting Problem
- Merge Sort
- Merge

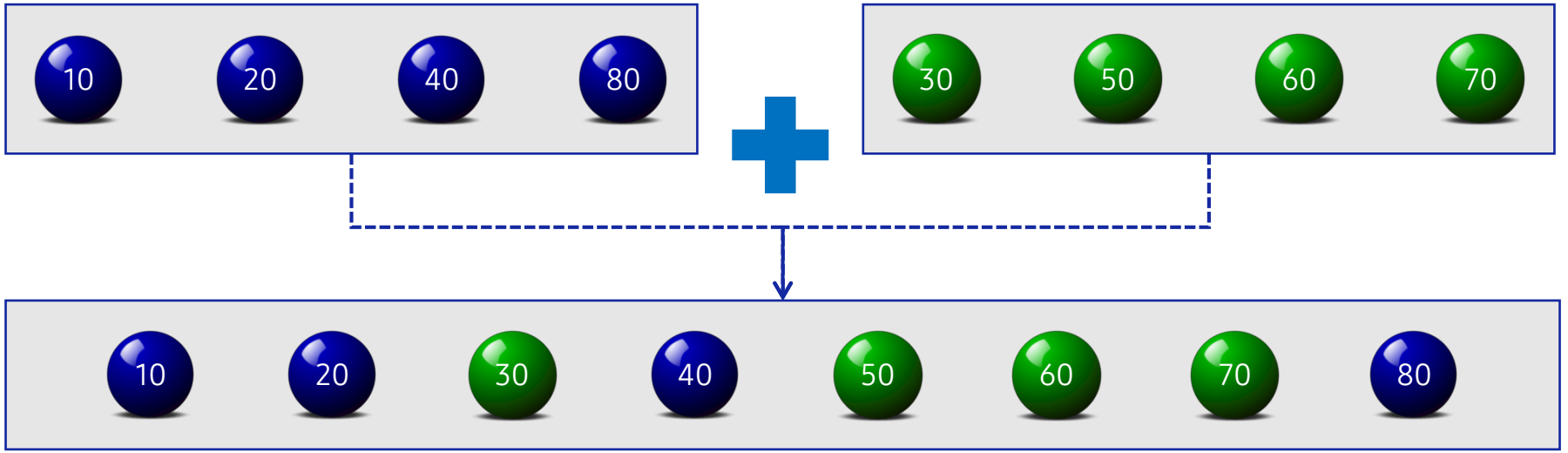
Mission

1. Real world problem

1.1 ບັນຫາຂອງ merge

ໃນບົດຮຽນທີ່ຜ່ານມາ, ເຮົາໄດ້ຮຽນ algorithm ການຈັດລຽງທີ່ມີ time complexity ເປັນ $O(N^2)$. ເຮົາສາມາດສ້າງ algorithm ການຈັດລຽງທີ່ມີປະສິດທິພາບດີກວ່ານີ້ໄດ້ບໍ?

ຕົວຢ່າງ ສົມມຸດເຮົາມີໜ່ວຍເຫຼັກສີຟ້າ 4 ໜ່ວຍ ແລະ ໜ່ວຍເຫຼັກສີຂຽວ 4 ໜ່ວຍ ດັ່ງລຸ່ມນີ້, ເຊິ່ງມີຮູບຮ່າງຄືກັນ ຂະໜາດເທົ່າກັນ, ແຕ່ນ້ຳໜັກຂອງແຕ່ລະໜ່ວຍບໍ່ເທົ່າກັນ. ຖ້າວ່າ, ຫາກຕ້ອງການຈັດລຽງໜ່ວຍເຫຼັກເຫຼົ່ານີ້ ຈາກນ້ຳໜັກໜ້ອຍຫາຫຼາຍຕາມລຳດັບຈົນຄົບ ເຮົາຈະມີວິທີການຈັດລຽງແນວໃດແດ່?

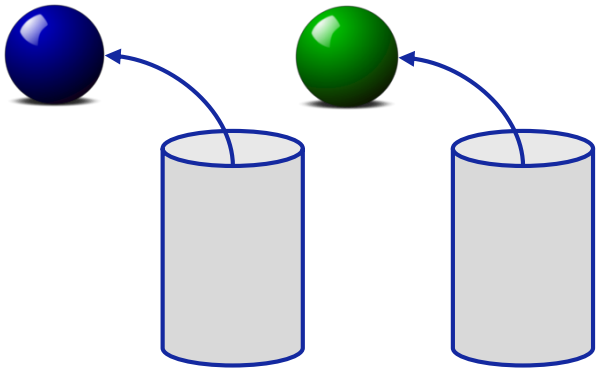


2. Mission

2.1. ລອມສອງລາຍການທີ່ຈັດລຽງ

- ໃຊ້ເຄື່ອງວັດແທກຄວາມສົມດຸນ. ໂດຍເຄື່ອງວັດແທກຄວາມສົມດຸນສາມາດດຳເນີນການດັ່ງຕໍ່ໄປນີ້:
  - ນຳເອົາໜ່ວຍເຫຼັກແຕ່ລະສີຈາກສອງຖັງ ມາປຽບທຽບກັນ ເລີ່ມຈາກໜ່ວຍທີ່ມີນ້ຳໜັກເບົາທີ່ສຸດຂອງແຕ່ລະຖັງ.
  - ໜ່ວຍເຫຼັກທີ່ມີນ້ຳໜັກເບົາກວ່າ ຈະຖືກເອົາມາວາງໄວ້ໃນຖັງລວມ.
- ເມື່ອເຮົາຮູ້ວ່າທັງສອງຖັງລ້ວນແຕ່ແມ່ນໜ່ວຍເຫຼັກທີ່ຈັດລຽງແລ້ວ, ເຮົາສາມາດໃຊ້ຫຼັກການປຽບທຽບໜ່ວຍເຫຼັກທັງສອງທີ່ມີນ້ຳໜັກເບົາ ແລ້ວຈັດລຽງຄືນໃໝ່
- ດ້ວຍຫຼັກການດັ່ງກ່າວ, ຈະຕ້ອງໄດ້ເຮັດການປຽບທຽບຫຼາຍປານໃດໃນການວາງໜ່ວຍເຫຼັກຕາມນ້ຳໜັກເພື່ອຈັດລຽງຄືນໃໝ່?

จัดลงๆ (Sorted)



จัดลงๆ (Sorted)

จัดลงๆ (Sorted)

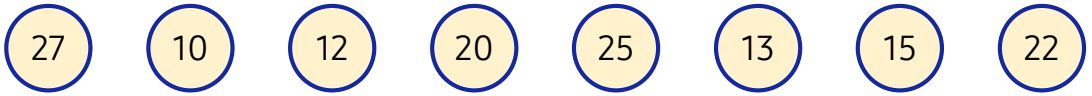
Key concept

1. ການຈັດລຽງແບບ Merge

1.1. ຕົວຢ່າງການຈັດລຽງແບບ merge

ການຈັດລຽງແບບ Merge ແມ່ນວິທີການຈັດລຽງລຳດັບອັນໜຶ່ງ ທີ່ໄດ້ແບ່ງບັນຊີລາຍການທີ່ບໍ່ໄດ້ຈັດລຽງອອກເປັນສອງສ່ວນຍ່ອຍ. ຈັດລຽງແຕ່ລະສ່ວນ ແລະໂຮມ ການຈັດລຽງນັ້ນເປັນລາຍການຈັດລຽງໃໝ່.

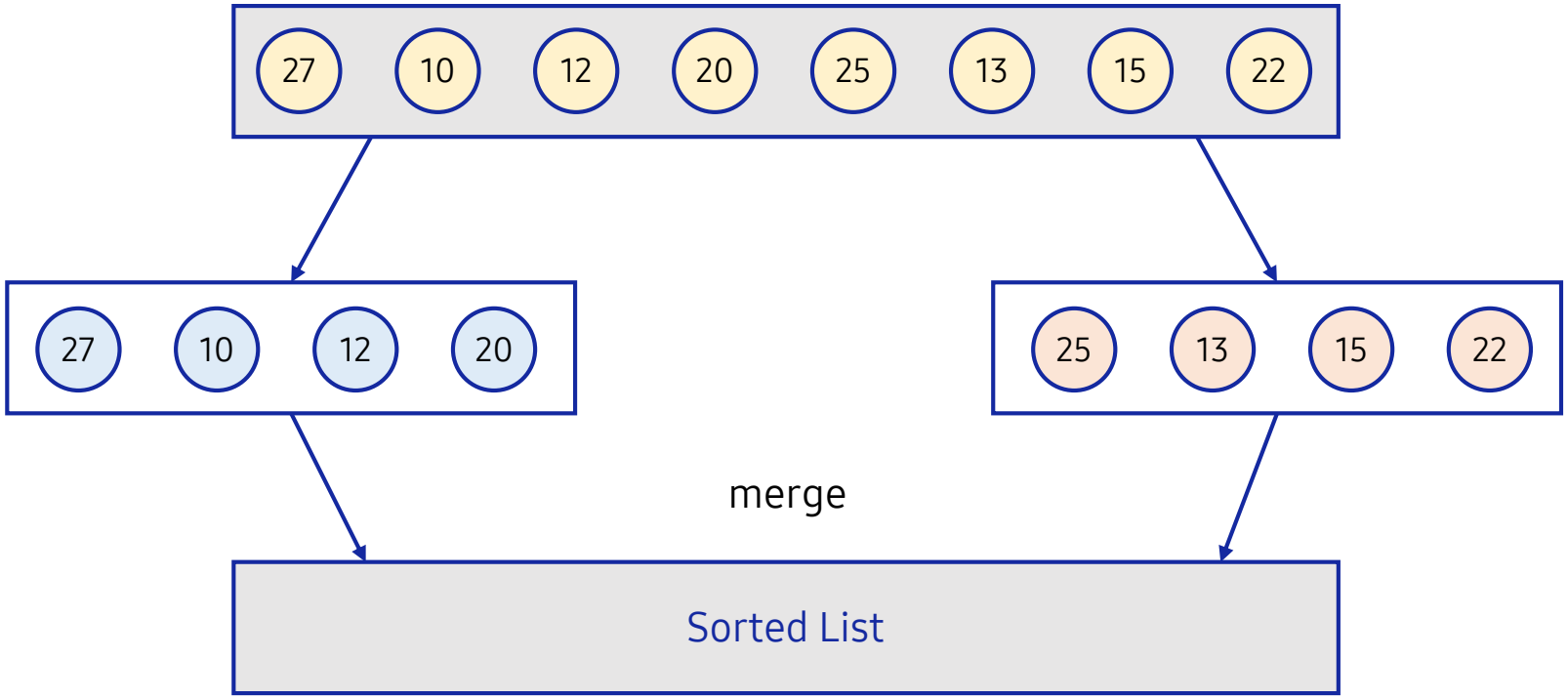
ຕົວຢ່າງ ສົມມຸດວ່າພວກເຮົາຕ້ອງການທີ່ຈະຈັດລຽງລຳດັບຂອງຂໍ້ມູນຊຸດນີ້ [27, 10, 12, 20, 25, 13, 15, 22] ໃຫ້ປະຕິບັດດັ່ງຕໍ່ໄປນີ້.



1. ภาษาจัดลงรูปแบบ Merge

1.1. ตัวอย่างของภาษาจัดลงรูปแบบ merge

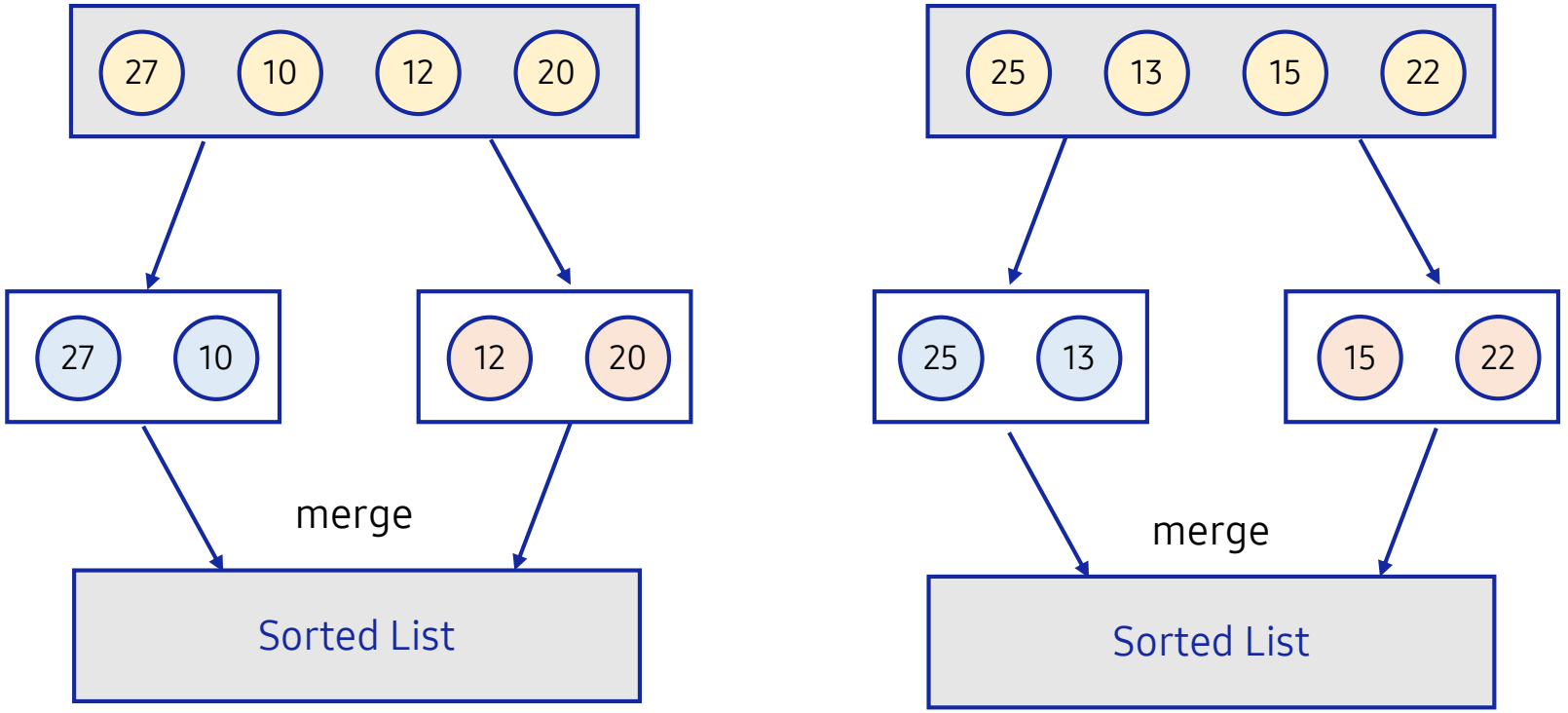
- เริ่มจาก, แบ่งลายการที่มีอิงประกอบ 8 ตัวออกเป็นสองส่วนเท่ากัน ส่วนละ 4 ตัว . ลายการที่แบ่งอยู่ในสถานะการจัดลงรูปแล้ว, เราสามารถลองรวมทั้งสองลายการที่ถูกแบ่งออกในสถานะที่ถูกจัดลงรูปแล้ว เพื่อเอื้นถึงการจัดลงรูปใหม่อีกเที่ยวหนึ่ง.



1. ການຈັດລຽງແບບ Merge

1.1. ຕົວຢ່າງຂອງການຈັດລຽງແບບ merge

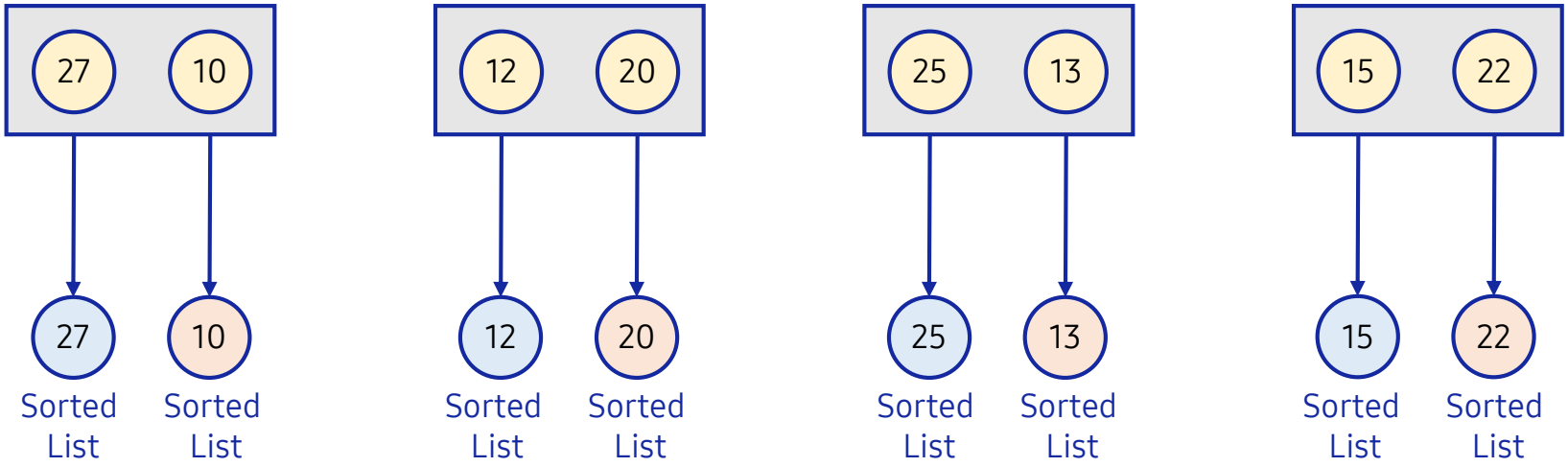
ລາຍການທີ່ບໍ່ໄດ້ຈັດລຽງຖືກແບ່ງອອກເປັນສອງສ່ວນ ແລະ ແຕ່ລະສ່ວນຍັງສາມາດແບ່ງຍ່ອຍອອກເປັນສອງສ່ວນອີກ ແລ້ວຈະເອົາມາລວມເຂົ້າໄປໃນລາຍການທີ່ຖືກຈັດລຽງ. ເນື່ອງຈາກລາຍການທີ່ແບ່ງອອກຍັງບໍ່ໄດ້ຖືກຈັດລຽງ, ດັ່ງນັ້ນ ຈະເອົາຜົນໄດ້ຮັບຂອງແຕ່ລະສ່ວນຍ່ອຍມາໂຮມກັນອີກເທື່ອໜຶ່ງ.



1. ການຈັດລຽງແບບ Merge

1.1. ຕົວຢ່າງຂອງການຈັດລຽງແບບ merge

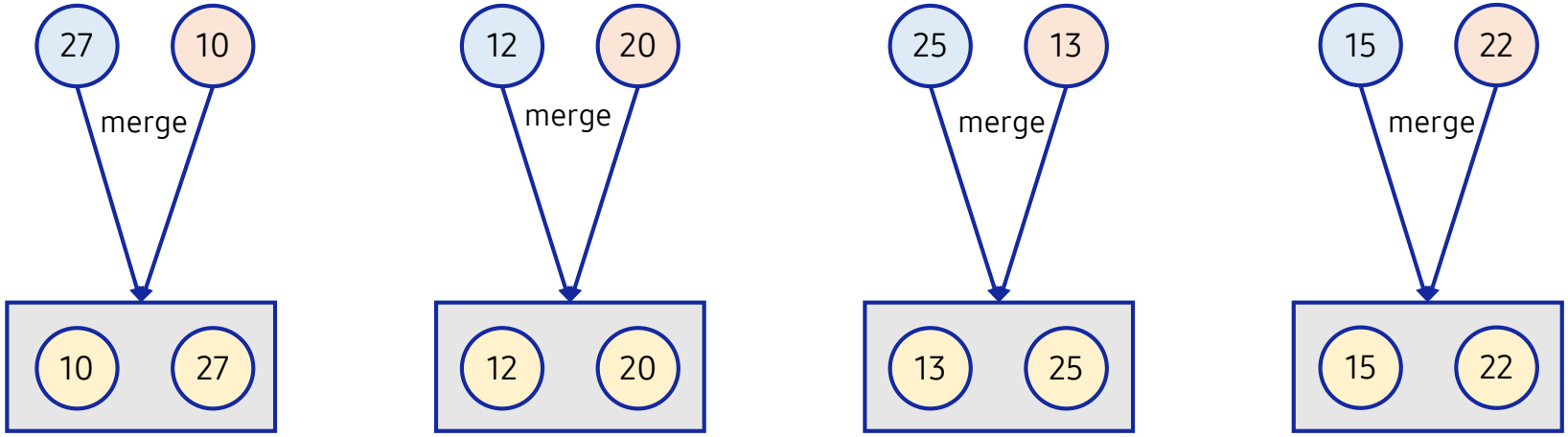
ເມື່ອອົງປະກອບໄດ້ແບ່ງອອກເປັນສອງສ່ວນຍ່ອຍ ຈົນແຕ່ລະສ່ວນຍ່ອຍມີພຽງໜຶ່ງອົງປະກອບ. ໃນກໍລະນີນີ້, ເນື່ອງຈາກລາຍການທີ່ມີອົງປະກອບໜຶ່ງອັນແມ່ນ ລາຍການທີ່ຖືກຈັດລຽງ, ລາຍການດັ່ງກ່າວສາມາດຖືກລວມເຂົ້າກັນອີກຄັ້ງໃນສະຖານະຈັດລຽງສຳເລັດ



1. ການຈັດລຽງແບບ Merge

1.1. ຕົວຢ່າງຂອງການຈັດລຽງແບບ merge

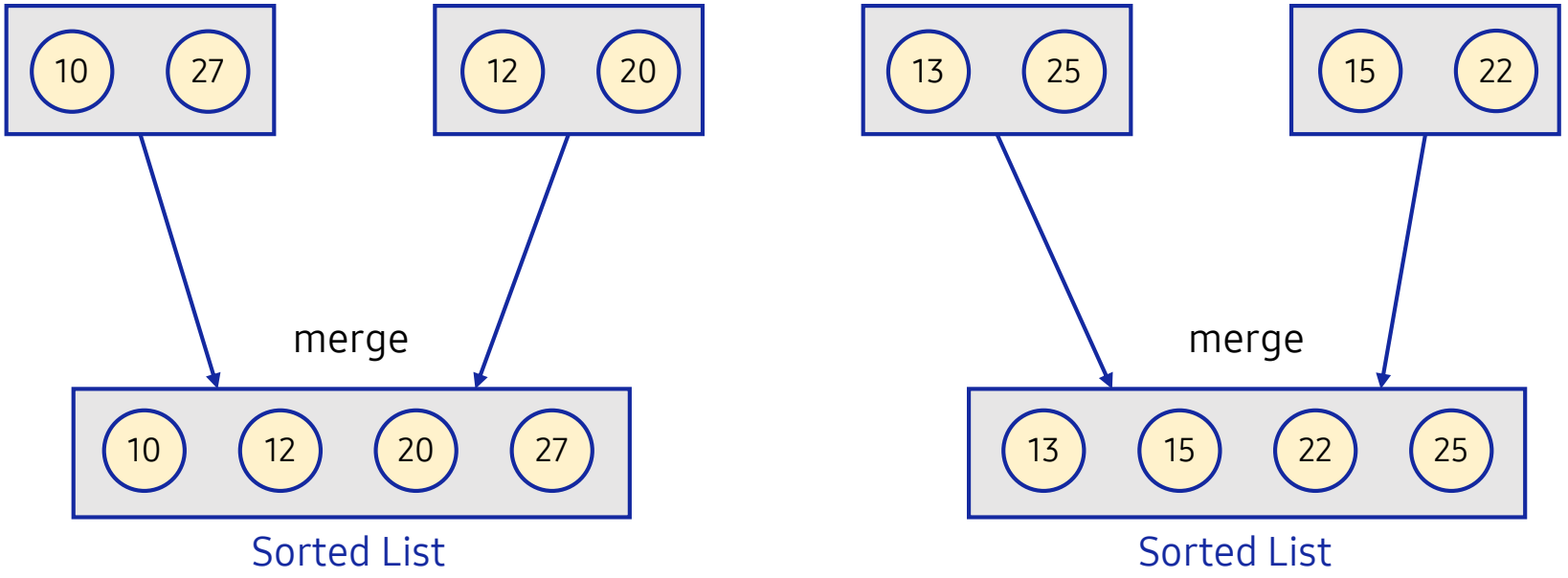
ການລວມລາຍການທີ່ມີ 1 ອົງປະກອບຈະສ້າງເປັນລາຍການຈັດລຽງທີ່ມີ 2 ອົງປະກອບ



1. ภาพจัดลงรูปแบบ Merge

1.1. ตัวอย่างของภาพจัดลงรูปแบบ merge

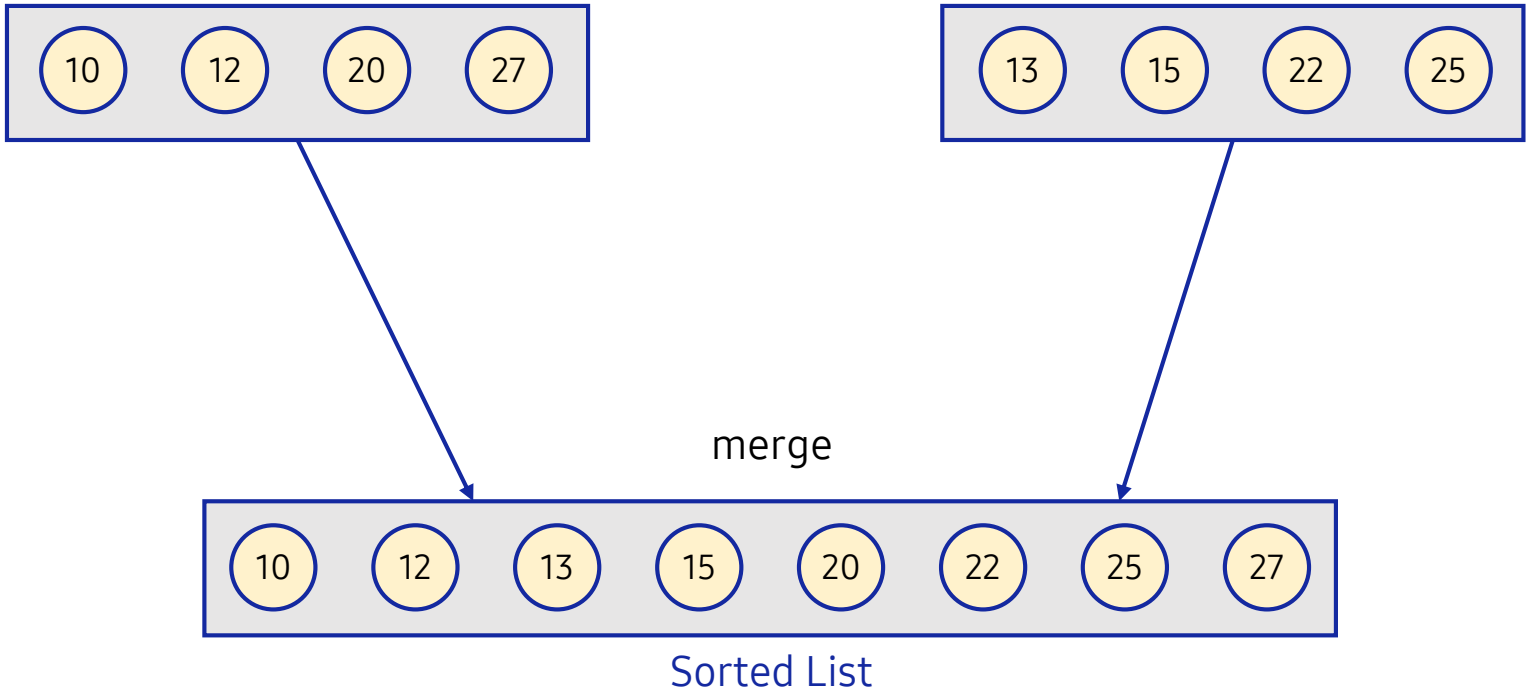
ภาพลอมลายภาพที่มี 2 อิงปะกอบ สำงลายภาพจัดลงรูงที่มี 4 อิงปะกอบ.



1. ການຈັດລຽງແບບ Merge

1.1. ຕົວຢ່າງຂອງການຈັດລຽງແບບ merge

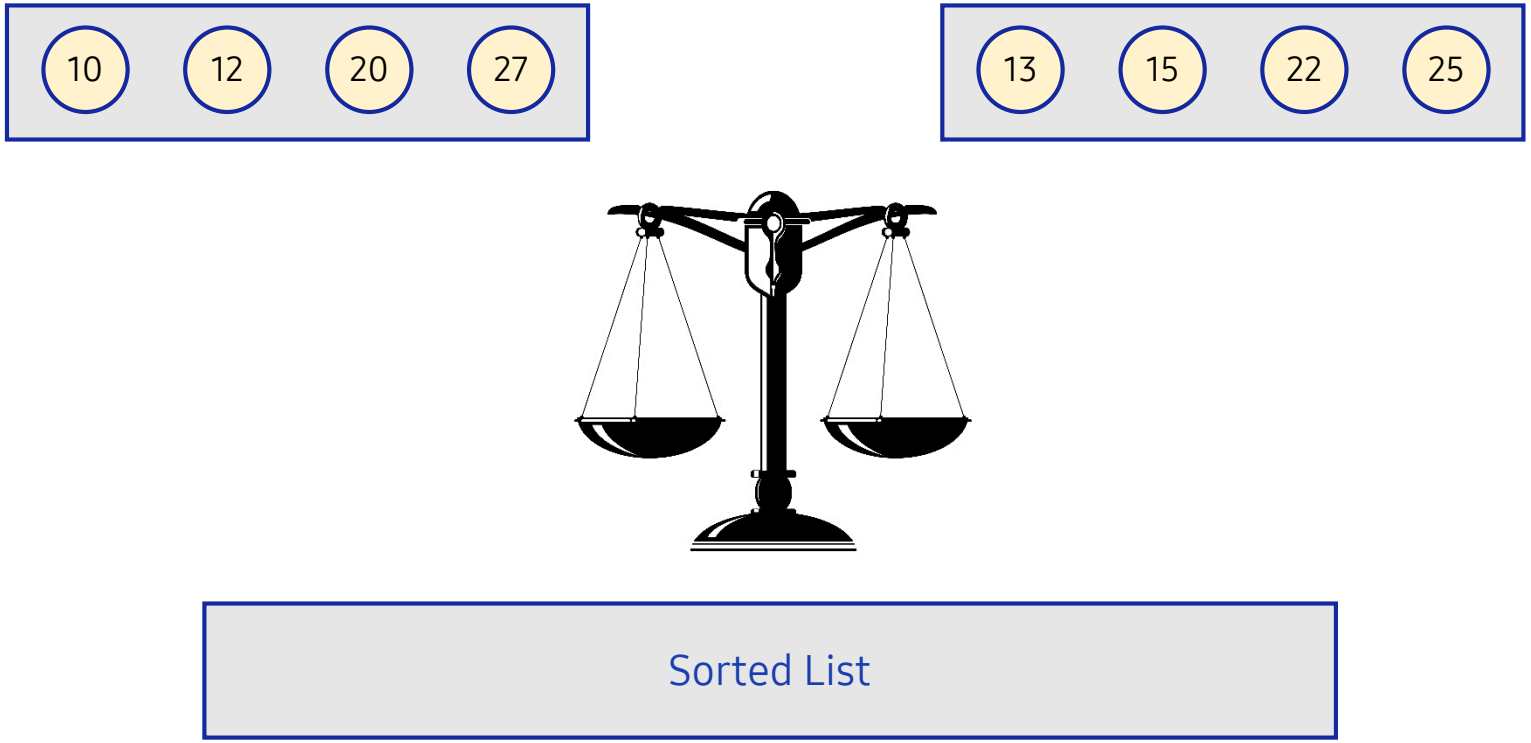
ການລວມລາຍການທີ່ມີ 4 ອົງປະກອບ ສ້າງລາຍການຈັດລຽງທີ່ມີ 8 ອົງປະກອບ.



1. ການຈັດລຽງແບບ Merge

1.2. ຕົວຢ່າງຂອງການຈັດລຽງແບບ merge

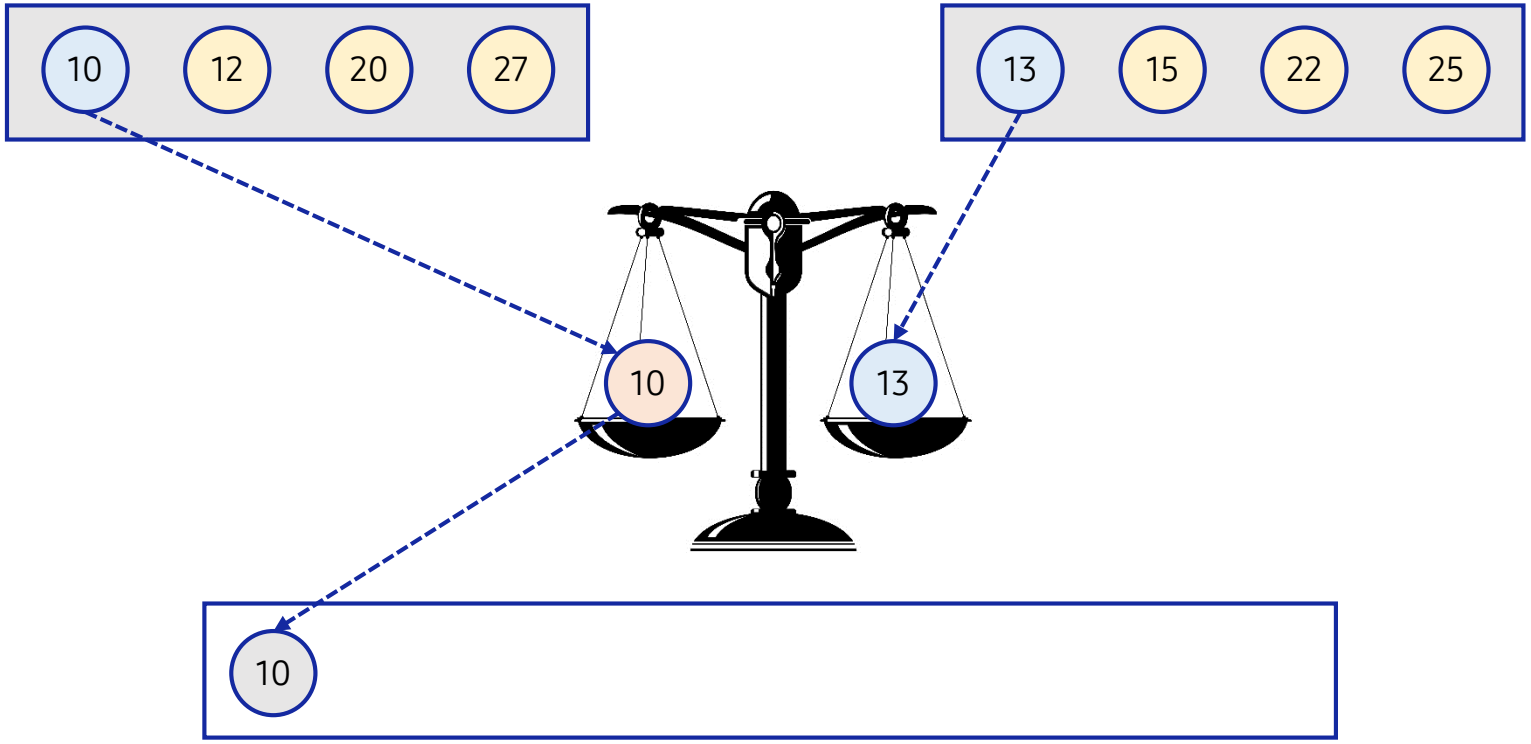
- ຊອກຫາວິທີການ ເພື່ອລວມ 2 ລາຍການຈັດລຽງ ໃຫ້ເປັນລາຍການຈັດລຽງອັນໜຶ່ງ



1. ການຈັດລຽງແບບ Merge

1.2. ຕົວຢ່າງຂອງການຈັດລຽງແບບ merge

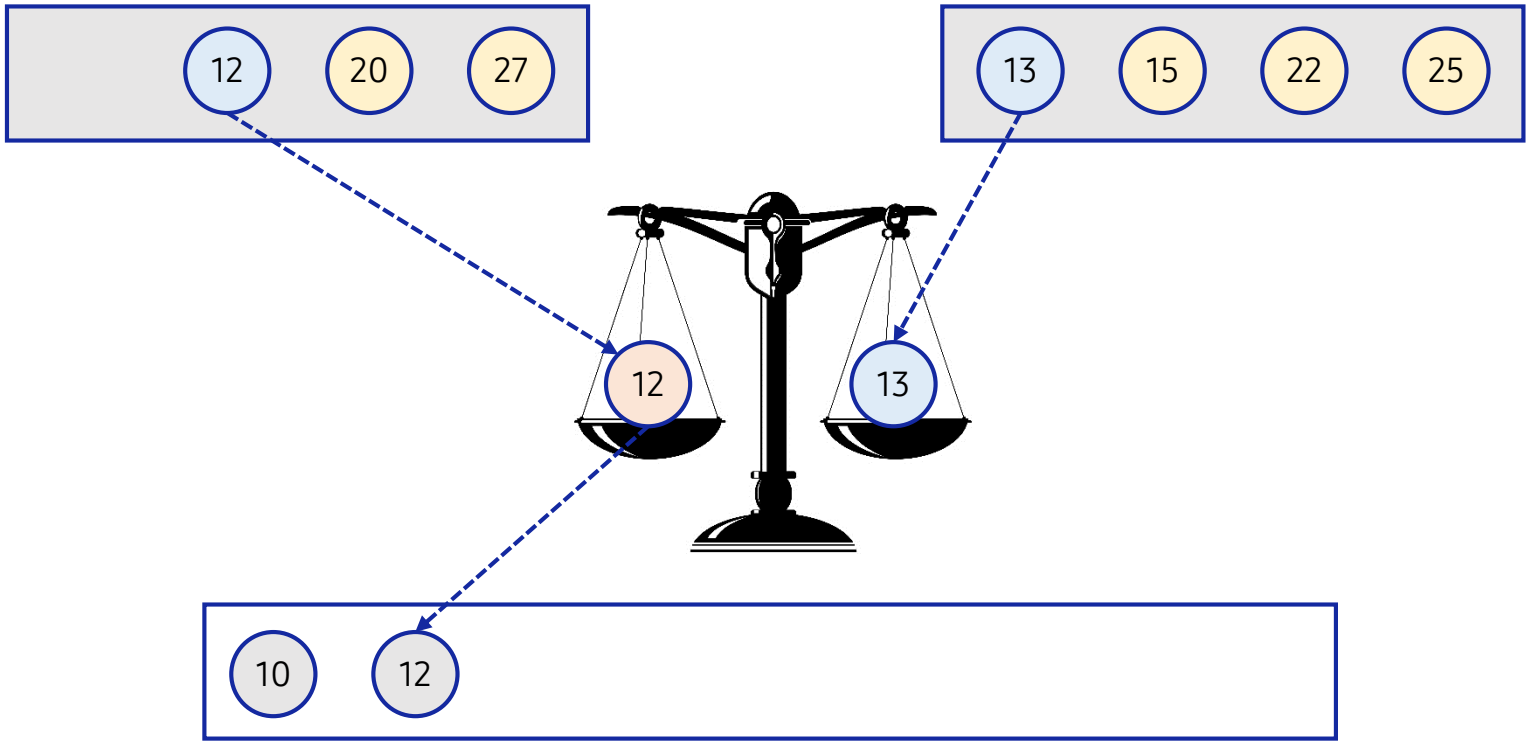
- ເລີ່ມຈາກ, ນຳເອົາອົງປະກອບທີ່ນ້ອຍທີ່ສຸດຈອງແຕ່ລະລາຍການມາປຽບທຽບກັນ.ໃນກໍລະນີນີ້, ເມື່ອພົບວ່າ 10 ນ້ອຍກວ່າ 13, 10 ຈຶ່ງຖືກເອົາມາວາງລົງໃນລາຍການທີ່ຖືກຈັດລຽງແລ້ວ.



1. ການຈັດລຽງແບບ Merge

1.2. ລວມສອງລາຍການທີ່ຈັດລຽງໄວ້ແລ້ວ

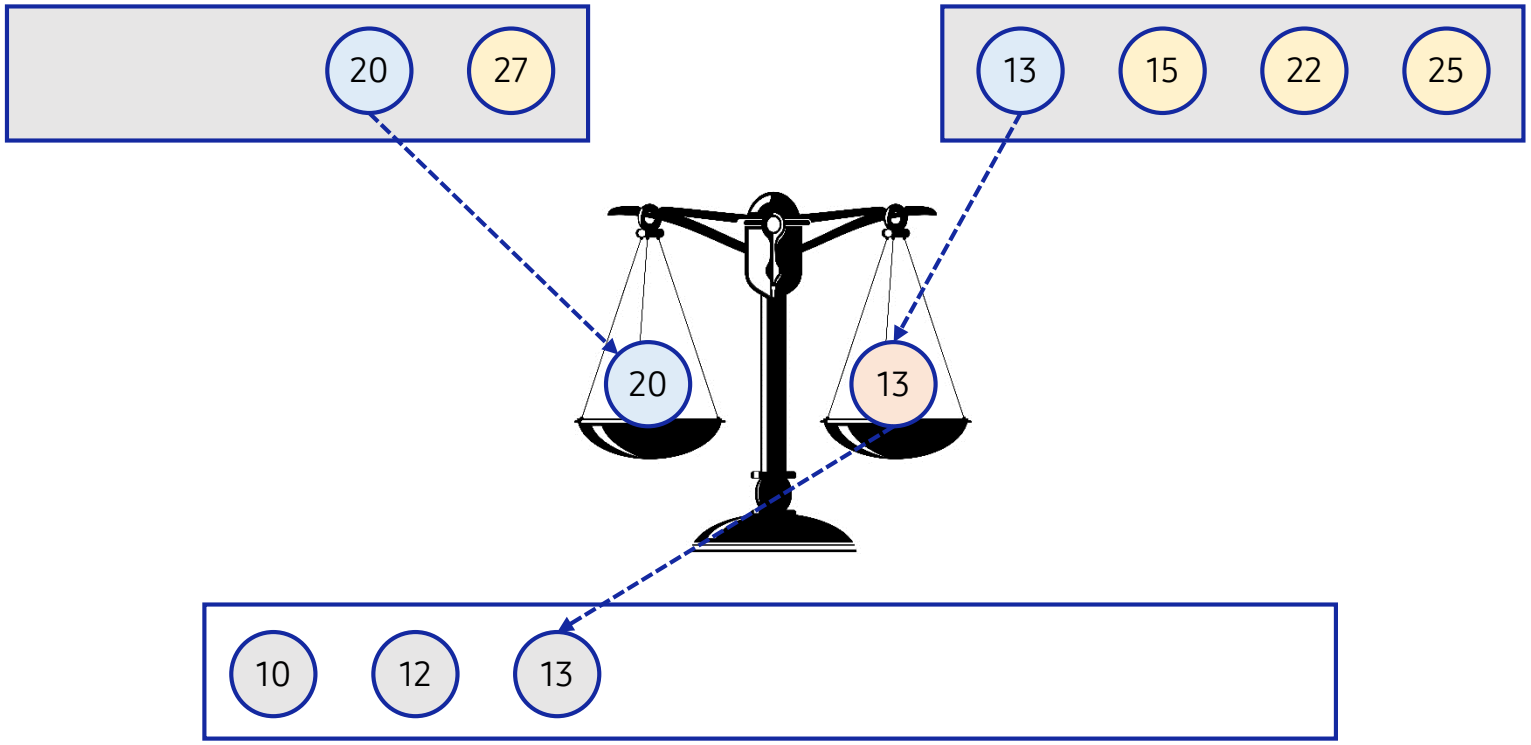
ນຳເອົາອົງປະກອບທີ່ຖືກຈັດລຽງອອກໄປ ແລະ ເລີ່ມຕົ້ນປຽບທຽບກັບອົງປະກອບຕໍ່ໄປ. ເມື່ອ 12 ນ້ອຍກວ່າ 13, 12 ຈຶ່ງຖືກຍ້າຍໄປໄວ້ໃນລາຍການທີ່ຈັດລຽງແລ້ວ.



1. ການຈັດລຽງແບບ Merge

1.2. ລວມສອງລາຍການທີ່ຈັດລຽງໄວ້ແລ້ວ

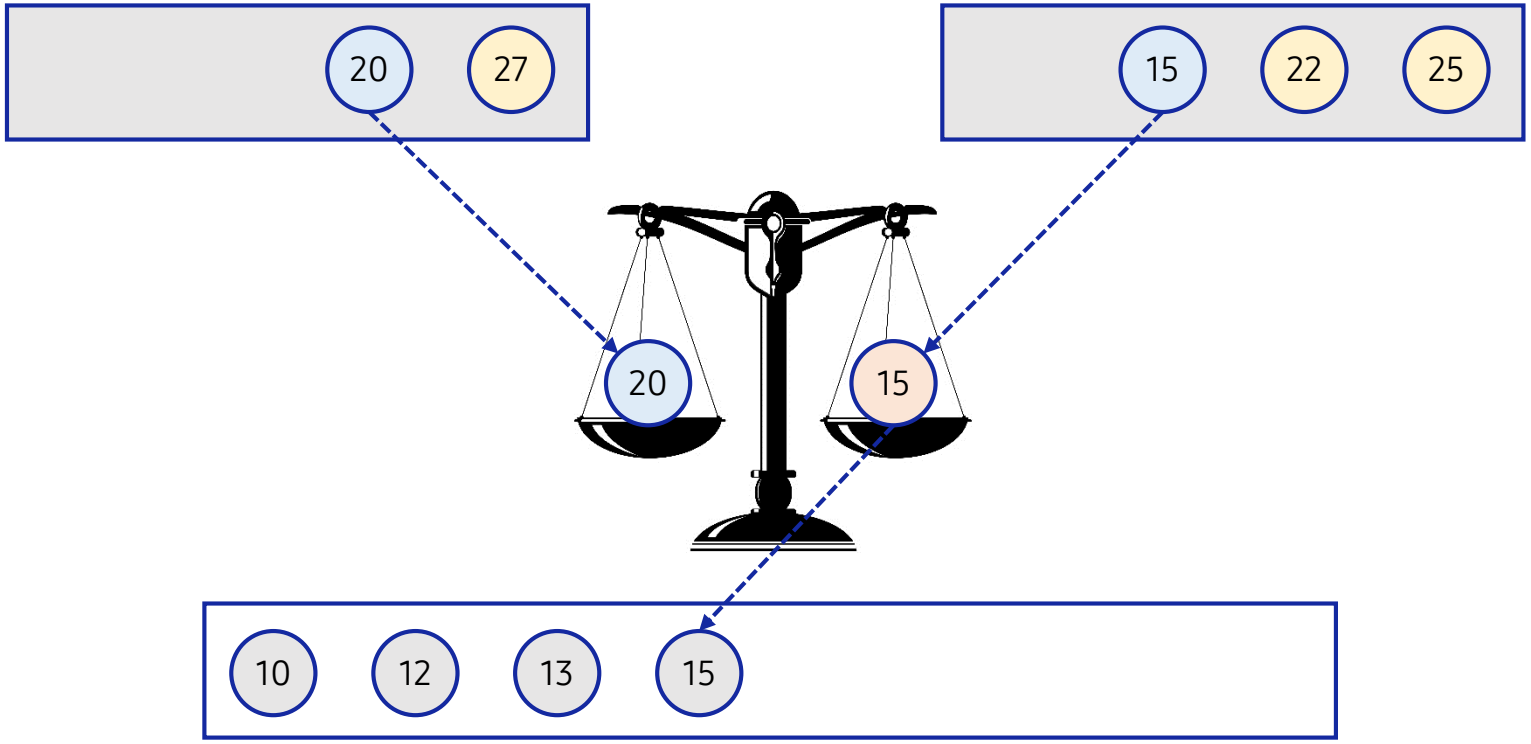
ນຳເອົາອົງປະກອບທີ່ຖືກຈັດລຽງອອກໄປ ແລະ ເລີ່ມຕົ້ນປຽບທຽບກັບອົງປະກອບຕໍ່ໄປ. ເມື່ອ 20 ໃຫຍ່ກວ່າ 13, 13 ຈຶ່ງຖືກຍ້າຍໄປລາຍການທີ່ຈັດລຽງແລ້ວ.



1. ການຈັດລຽງແບບ Merge

1.2. ລວມສອງລາຍການທີ່ຈັດລຽງໄວ້ແລ້ວ

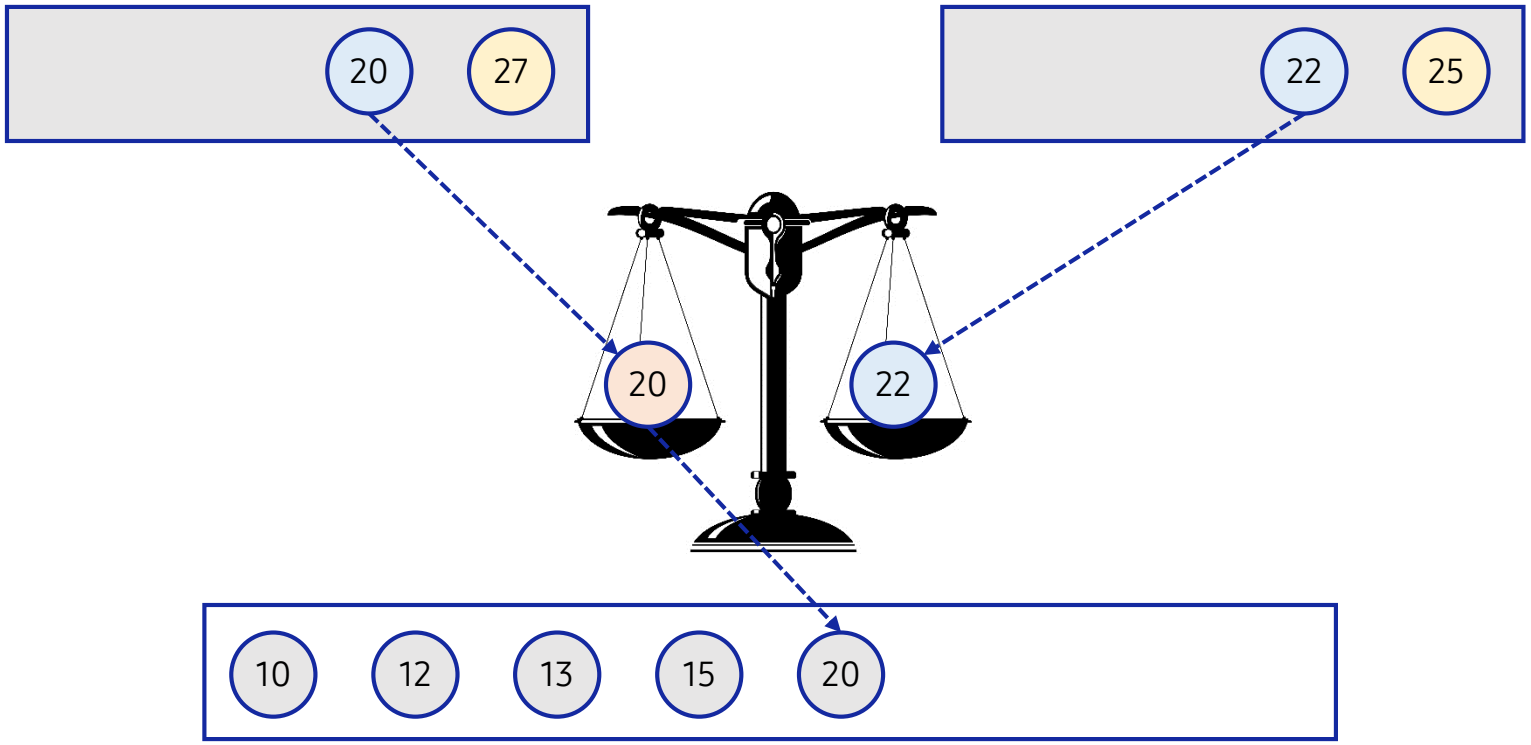
ເອົາອົງປະກອບທີ່ນ້ອຍກວ່າອອກ ແລ້ວສົ່ງໄປຫາລາຍການທີ່ຈັດລຽງ ແລະ ປຽບທຽບອົງປະກອບຕໍ່ໄປ. ເນື່ອງຈາກ 15 ແມ່ນຫນ້ອຍກວ່າ 20, 15 ຖືກເພີ່ມເຂົ້າໃນລາຍການທີ່ຈັດລຽງ.



1. ການຈັດລຽງແບບ Merge

1.2. ລວມສອງລາຍການທີ່ຈັດລຽງໄວ້ແລ້ວ

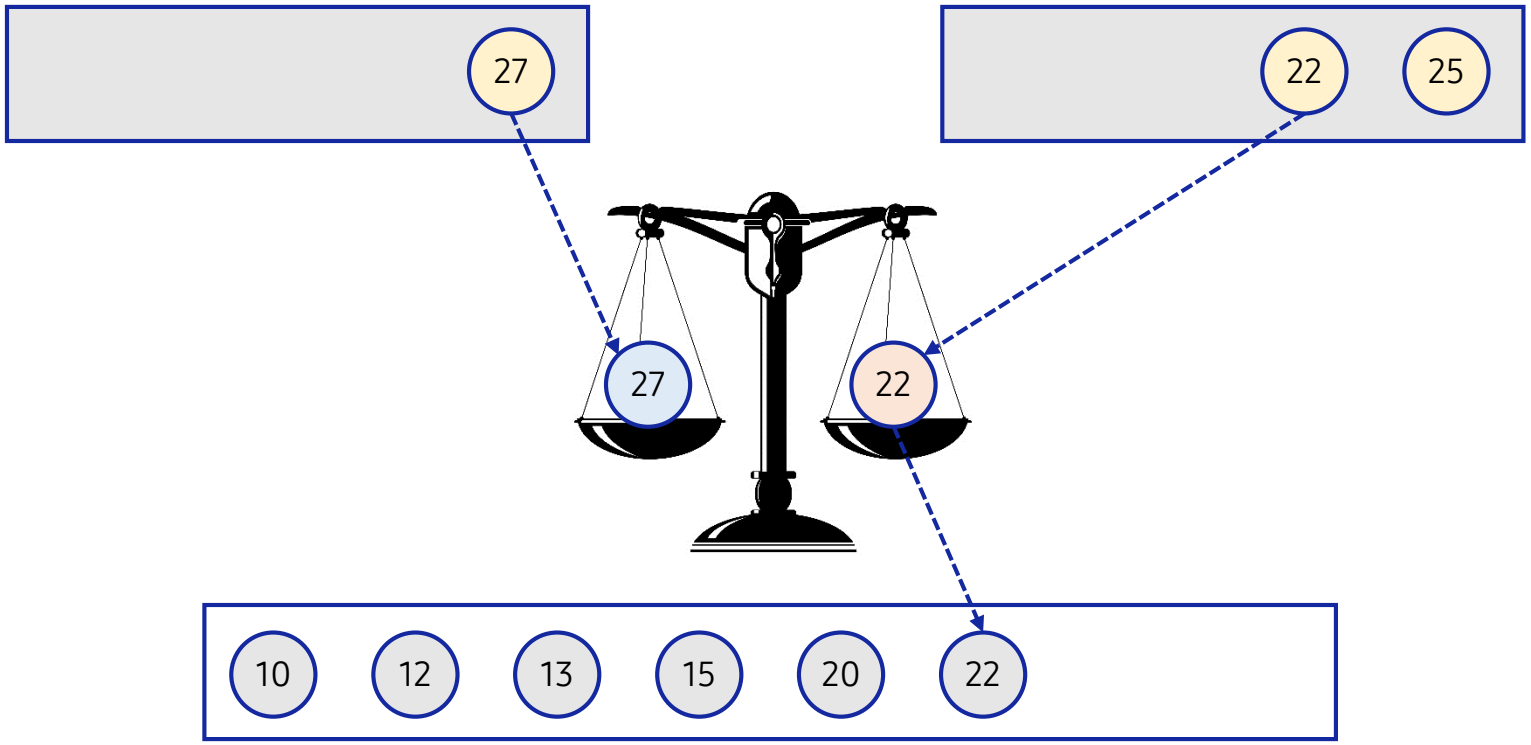
- ເອົາອົງປະກອບທີ່ມີຄ່ນ້ອຍກວ່າໄປໄວ້ໃນລາຍການທີ່ຈັດລຽງ ແລ້ວປຽບທຽບອົງປະກອບຕໍ່ໄປ. ເນື່ອງຈາກ 20 ແມ່ນນ້ອຍກວ່າ 22, 20 ຖືກເພີ່ມເຂົ້າໃນລາຍການທີ່ຈັດລຽງ.



1. ການຈັດລຽງແບບ Merge

1.2. ລວມສອງລາຍການທີ່ຈັດລຽງໄວ້ແລ້ວ

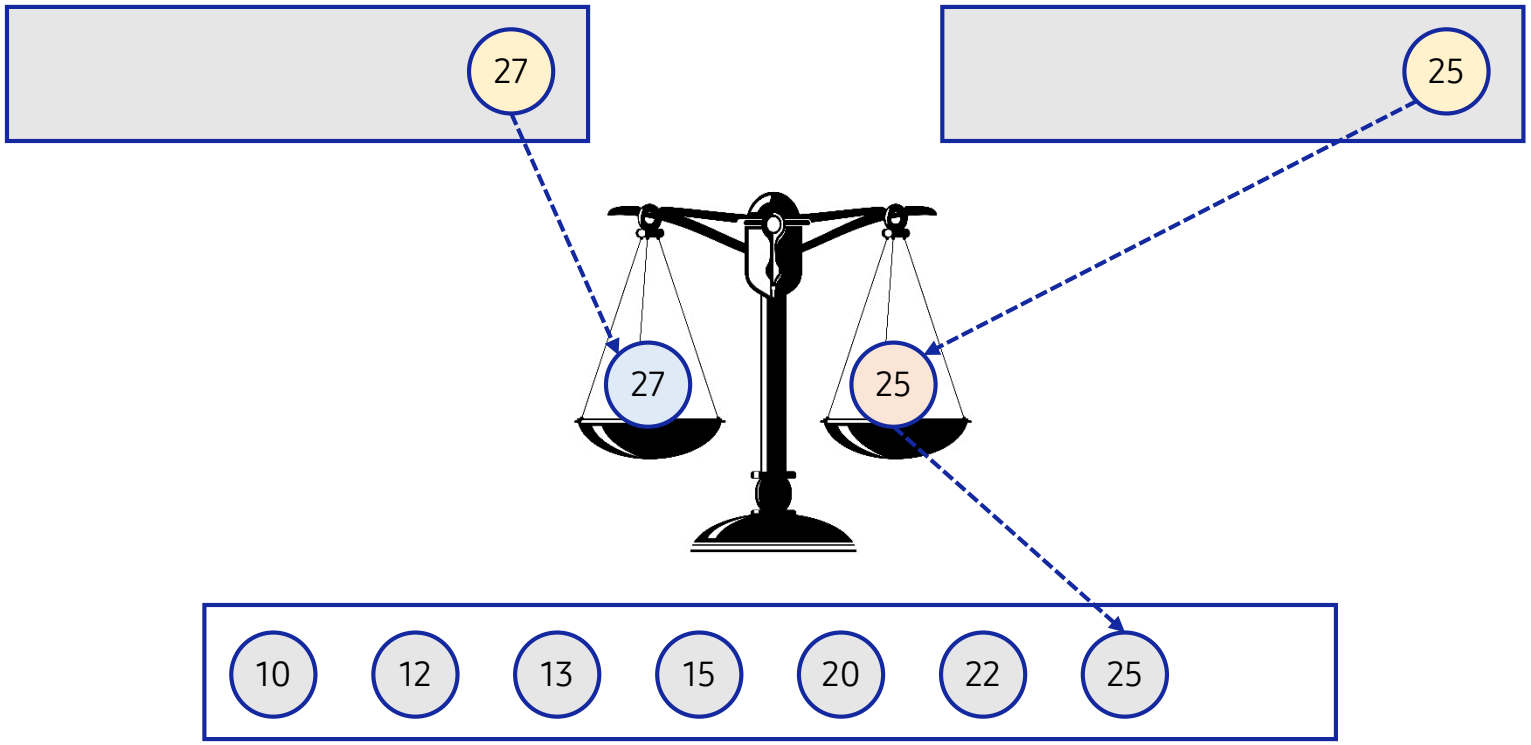
- ເອົາອົງປະກອບທີ່ນ້ອຍກວ່າໄປໄວ້ໃນລາຍການທີ່ຈັດລຽງ ແລະ ປຽບທຽບອົງປະກອບຕໍ່ໄປ. ເນື່ອງຈາກ 22 ແມ່ນຫນ້ອຍກວ່າ 27, 22 ຖືກເພີ່ມເຂົ້າໃນລາຍການທີ່ຈັດລຽງ.



1. ການຈັດລຽງແບບ Merge

1.2. ລວມສອງລາຍການທີ່ຈັດລຽງໄວ້ແລ້ວ

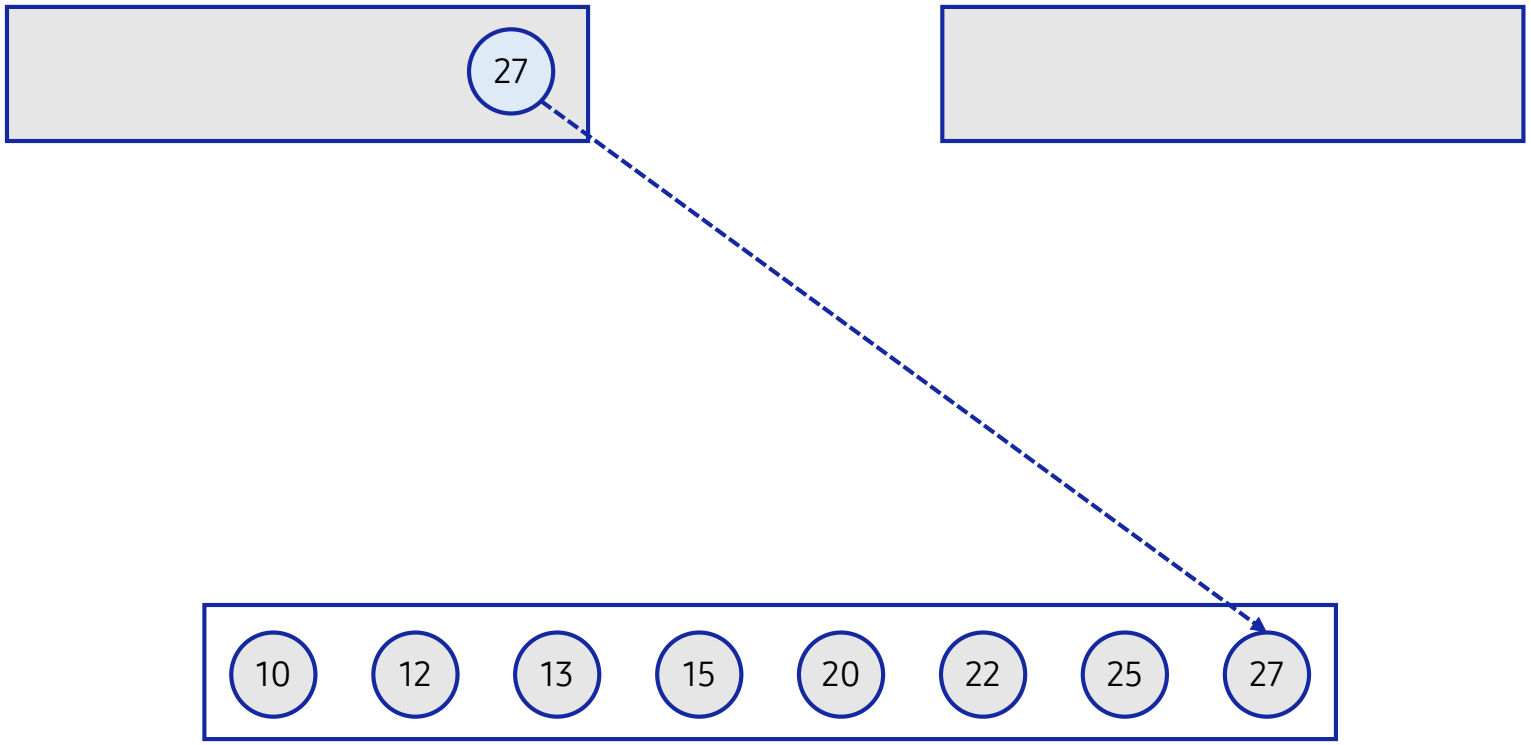
ເອົາອົງປະກອບທີ່ນ້ອຍກວ່າໄປໄວ້ໃນລາຍການທີ່ຈັດລຽງ ແລະ ປຽບທຽບອົງປະກອບຕໍ່ໄປ. ເນື່ອງຈາກ 25 ແມ່ນຫນ້ອຍກວ່າ 27, 25 ຖືກເພີ່ມເຂົ້າໃນລາຍການທີ່ຈັດລຽງ.



1. ການຈັດລຽງແບບ Merge

1.2. ລວມສອງ lists ທີ່ຈັດລຽງໄວ້ແລ້ວ

ຖ້າຫນຶ່ງໃນສອງລາຍການ ຫວ່າງເປົ່າ, ອົງປະກອບທັງໝົດໃນລາຍການທີ່ເຫຼືອຈະຖືກເພີ່ມເຂົ້າໃນລາຍການທີ່ຈັດລຽງ.



Let’s code

1. ส้างกามจัดล฽งแบบ Merge ด้อยกามใช้ห่อยถอามจำเพิ่มเติม

1.1. ส้างกามจัดล฽งแบบ Merge ด้อยอิทิกาม divide-and-conquer

อิทิที่ง่ายในกามส้างกามจัดล฽งแบบ merge เพื่อจัดล฽งข้มูนในลายกามที่ให้มาโดยกามแบ่งออกเป้น สอ฽งลายกาม. อิทิกามออกแบบ algorithm นี้เอิ้นว่า divide-and-conquer ซึ่งจะสึกสาละอ฽งดอิกเท็อ หนึ่งใน Unit 31.

```
1 def mergesort1(S):
2     n = len(S)
3     if n > 1:
4         print(S)
5         mid = n // 2
6         L, R = S[:mid], S[mid:]
7         mergesort1(L)
8         mergesort1(R)
9         merge1(S, L, R)
```

Line 1~3

- กามจัดล฽งแบบ Merge จัดล฽งข้มูนใน S ที่ให้โดยกามแบ่งออกเป้นสอ฽งส่วนไปเลื้อยๆ ถ้าขะ หฒาดของ S หฺายกว่า 1.
- ถ้าขะหมาดของ S น้อยกว่าหฺือเท่ากับ 1, มันยุดกามปะติบัดงาน ซึ่งสะแดงว่าลายกามข้มูนถึกจัด ล฽งธ฽บร้อยแล้อ.

1. ส้างกามจัดล฽งแบบ Merge ด้อยกามใช้ห่อยถอามจำเพิ่มเติม

1.1. ส้างกามจัดล฽งแบบ Merge ด้อยอิทิกาม divide-and-conquer

- ถ้าขะหมาดของ S ใหย่กอ่า 1, จะแบ่งออกเป้นสองลายกาม, แล้อเข้าไปจัดล฽งในสองลายกามนั้น และ ลอมเข้าเป้นลายกามด฽อกัน.

```
1  def mergesort1(S):
2      n = len(S)
3      if n > 1:
4          print(S)
5          mid = n // 2
6          L, R = S[:mid], S[mid:]
7          mergesort1(L)
8          mergesort1(R)
9          merge1(S, L, R)
```

Line 5~8

- ในกามแบ่ง S ให้ออกเป้นสองลายกามนั้น ต้องซอกหาถ่ากางก่อน แล้อจึ่งแบ่งออกเป้นสองลายกาม L และ R.
- สังเกดเห็นอ่าลายกาม S ได้ถืกแยกออกเป้นสองลายกามถื L และ R.
- จัดล฽งสองลายกาม L และ R แยกจากกัน โดยกามเอิ้นใช้ฟังฌันแบบ recursive.

1. สร้างการจัดลำดับแบบ Merge ด้วยการใช้หน่วยความจำเพิ่มเติม

1.1. สร้างการจัดลำดับแบบ Merge ด้วยวิธีการ divide-and-conquer

ถ้า L และ R เป็นลายการที่ถูกจัดลำดับไว้แล้ว, สามารถลองสองลายการนี้เข้าไปไว้ในลายการ S.

```
1  def mergesort1(S):
2      n = len(S)
3      if n > 1:
4          print(S)
5          mid = n // 2
6          L, R = S[:mid], S[mid:]
7          mergesort1(L)
8          mergesort1(R)
9          merge1(S, L, R)
```

Line 9

- ลอม L และ R เพื่อสร้างลายการ S ที่ถูกจัดลำดับ.

1. ส้างกามจัดล฽งแบบ Merge ด้อยกามใช้ฑ่อยถอามจำเพิ่มเติม

1.2. ลอมสองลายกามที่ถืกล฽ง เป้นลายกามที่ถืกล฽งอันฒึ่ง

ฟังฌัน merge1() ส้างลายกาม S ที่ถืกจัดล฽งโดยกามลอม L และ R ที่ถืกจัดล฽งแล้อเข้ำด้อยกัน.

```
1  def merge1(S, L, R):
2      k = 0
3      while len(L) > 0 and len(R) > 0:
4          if L[0] <= R[0]:
5              S[k] = L.pop(0)
6          else:
7              S[k] = R.pop(0)
8          k += 1
9      while len(L) != 0:
10         S[k] = L.pop(0)
11         k += 1
12     while len(R) != 0:
13         S[k] = R.pop(0)
14         k += 1
```

Line 2~8

- กามเพิ่มดัดฉะมิ k, อิงปะกอบที่มิถ่าน้อยที่สุดต๊อทำอิดของ L หื R จะถืกเอิาไปเก้บไว้เป้นอิง ปะกอบที k ของ S.

1. ส้างกามจัดล฽งแบบ Merge ด้อยกามใช้ห่อยถอามจำเพิ่มเติม

1.2. ลอมสองลายกามที่ถืกล฽ง เป้นลายกามที่ถืกล฽งอันฒึ่ง

ฟังฺชัน merge1() ส้างลายกาม S ที่ถืกจัดล฽งโดยกามลอม L แล๊ะ R ที่ถืกจัดล฽งแล้วเข๊้าด้อยกัน.

```
1  def merge1(S, L, R):
2      k = 0
3      while len(L) > 0 and len(R) > 0:
4          if L[0] <= R[0]:
5              S[k] = L.pop(0)
6          else:
7              S[k] = R.pop(0)
8          k += 1
9      while len(L) != 0:
10         S[k] = L.pop(0)
11         k += 1
12     while len(R) != 0:
13         S[k] = R.pop(0)
14         k += 1
```

Line 9~11

- เมื่อออกจาก while-loop ขอ๋ Line 3-8, หนึ่๋งในสอ๋งลายกาม L แล๊ะ R ได้ปะมอมผืมอึ๋งปะกอบทั๋งหมืด.
- ถ้าอึ๋งปะกอบขอ๋ง L เห฼ือ, อึ๋งปะกอบที่ยั๋งเห฼ือทั๋งหมืดขอ๋งมันจะถืกเอีาไปเพิ่มใส่ S.

1. ส้างกามจัดล฽งแบบ Merge ด้อยหม่อยถอามจำเพิ่มเติม

1.2. ลอมสองลายกามที่ถืกล฽ง เป้นลายกามที่ถืกล฽งอันหนึ่ง

ฟังฺชัน merge1() ส้างลายกาม S ที่ถืกจัดล฽งโดยกามลอม L แลฯ R ที่ถืกจัดล฽งแล้วเข้ำด้อยกัน.

```
1  def merge1(S, L, R):
2      k = 0
3      while len(L) > 0 and len(R) > 0:
4          if L[0] <= R[0]:
5              S[k] = L.pop(0)
6          else:
7              S[k] = R.pop(0)
8          k += 1
9      while len(L) != 0:
10         S[k] = L.pop(0)
11         k += 1
12     while len(R) != 0:
13         S[k] = R.pop(0)
14         k += 1
```

Line 12~14

- ถ้ำอิงปะกอบ R ยังเห฼ือ, สะแดงอ่ำอิงปะกอบของ L บ่ เห฼ือแล้อ ยังมิพ฽งแต่อิงปะกอบของ R เท่ำนั้น.
- อิงปะกอบที่ยังเห฼ือทังหมิดของ R จะถืกเอำไปเพิ่มเข้ำใน S.

1. ส้างกามจัดล฽งแบบ Merge ด้อยกามใช้ฑ่อยถอามจำเพิ่มเติม

1.3. Run โปธแกธมกามจัดล฽งแบบ merge

เมื่อฟังชัน merge1() ถืกนำไปใช้กับฟังชัน mergesort1(), จะมิผิมได้ธับดั่งต่ำไปนี้.

```
1 S = [27, 10, 12, 20, 25, 13, 15, 22]
2 mergesort1(S)
3 print(S)
```

```
[27, 10, 12, 20, 25, 13, 15, 22]
[27, 10, 12, 20]
[27, 10]
[12, 20]
[25, 13, 15, 22]
[25, 13]
[15, 22]
[10, 12, 13, 15, 20, 22, 25, 27]
```

Line 1~3

- S ได้ธับโดย print(S) แม่นพิมยู่ใน Line 4 ของฟังชัน mergesort1().
- ผิมได้ธับ S จาก Line 3 เป้นผิมได้ธับกามจัดล฽ง ฬังจากฟังชัน mergesort1() ปะมอมผิมแล้อ.

One More Step

- ฟังชัน mergesort1() ไดใชหนอยถอามจำเพิ่มเติมหลายปานใด?
- ฟังชัน mergesort1() สางลายการใหม่ L และ R ในแต่ละถังที่ฟังชันถูกเอิ้นใช้. ถาขะหนาดของสองลายการนิลอมเข้ากัน, มันจะเท่ากับขะหนาดของ S ที่เป็นต้นสะบับ.
- นับตั้งแต่การเอิ้นใช้ recursive ลายการ L และ R ตามลำดับ, ขะหนาดของสองลายการในแต่ละการเอิ้นใช้ recursive แม่นปะมานเถิงหนึ่งของขะหนาดของการเอิ้นใช้ที่ผ่านมา. ดั่งนั้น, ขะหนาดของลายการที่นำใช้เพิ่มเป็นดั่งต่ำไปนี้.

$$N + \frac{N}{2} + \frac{N}{2^2} + \cdots + \frac{N}{2^k} = 2N$$

  - ฟังชัน mergesort1() จะต้องใช้หนอยถอามจำเพิ่มด้อยขะหนาด 2N เซิ่งเป็นสองเท่าของขะหนาด N ของ S ที่ให้มา.
  - สามาดปับปุงขั้นตอนอิที่นี้เพื่อให้ใช้หนอยถอามจำเพิ่มพรงแต่ N ได้.

2. ການສ້າງການຈັດລຽງແບບ Merge ທີ່ມີປະສິດທິພາບ

2.1. ການຈັດລຽງແບບ Merge ທີ່ໄດ້ປັບປຸງ

ฟังชัน mergesort2() จัดลຽง S ໂດຍກຳໜົດຄ່າ low ແລະ high ແທນທີ່ຈະແບ່ງ S ເປັນ L ແລະ R.

```
1 def mergesort2(S, low, high):
2     if low < high:
3         print(S)
4         mid = (low + high) // 2
5         mergesort2(S, low, mid)
6         mergesort2(S, mid + 1, high)
7         merge2(S, low, mid, high)
```

Line 1~3

- ສຳລັບ S ທີ່ໃຫ້ມາ, ມີຕົວດັດຊະນີ low ແລະ high ເປັນພາຣາມິເຕີ ຂໍ້ມູນປ້ອນເຂົ້າ.
- ການເອີ້ນໃຊ້ recursive ແມ່ນເຮັດໄດ້ໃນກໍລະນີຄ່າ low ນ້ອຍກວ່າຄ່າ high, ຖ້າບໍ່ດັ່ງນັ້ນ, ຈະຢຸດການຈັດລຽງ.

2. ການສ້າງການຈັດລຽງແບບ Merge ທີ່ມີປະສິດທິພາບ

2.1. ການຈັດລຽງແບບ Merge ທີ່ໄດ້ປັບປຸງ

ฟังชัน mergesort2() จัดลรง S โดยกำหนดค่า low และ high แทนที่จะแบ่ง S เป็น L และ R.

```
1 def mergesort2(S, low, high):
2     if low < high:
3         print(S)
4         mid = (low + high) // 2
5         mergesort2(S, low, mid)
6         mergesort2(S, mid + 1, high)
7         merge2(S, low, mid, high)
```

Line 4~7

- ເພື່ອທີ່ຈະແບ່ງແລະຈັດລຽງອົງປະກອບໃນຊ່ວງ low ແລະ high, ໄດ້ເອີ້ນໃຊ້ recursive ฟังชันปะติบัดการดั่งก่าอ โดยอิงใส่ค่า mid.
- ฟังชัน merge2() ลอมสองอิงปะกอบที่จัดลรงแล้วเข้าด้วยกัน โดยอิงใส่ค่า mid.

2. ການສ້າງການຈັດລຽງແບບ Merge ທີ່ມີປະສິດທິພາບ

2.2. ລວມສອງລາຍການທີ່ຈັດລຽງແລ້ວເພື່ອປັບປຸງການຈັດລຽງແບບ merge

ฟังชัน merge2() จัดลงอิงปะกอบจากซ้ายหาขวาอยู่ในช่วง low และ high โดยอิงใส่ค่า mid.

```
1  def merge2(S, low, mid, high):
2      R = []
3      i, j = low, mid + 1
4      while i <= mid and j <= high:
5          if S[i] < S[j]:
6              R.append(S[i]); i += 1
7          else:
8              R.append(S[j]); j += 1
9      if i > mid:
10         for k in range(j, high + 1):
11             R.append(S[j])
12     else:
13         for k in range(i, mid + 1):
14             R.append(S[i])
15     for k in range(len(R)):
16         S[low + k] = R[k]
```

Line 2~8

- เพิ่มอิงปะกอบที่จัดลงแล้วใส่ลายการใหม่ R, เลิ่มต้นที่ตำแหน่ง i และ j มีค่าเท่า low และ mid+1, ตามลำดับ.
- ให้จำไอ้อ่า อิงปะกอบก่อน mid และ หลังจาก mid + 1 แม่นอิงปะกอบที่จัดลงแล้ว ตามลำดับ.

2. ການສ້າງການຈັດລຽງແບບ Merge ທີ່ມີປະສິດທິພາບ

2.2. ລວມສອງລາຍການທີ່ຈັດລຽງແລ້ວເພື່ອປັບປຸງການຈັດລຽງແບບ merge

ฟังชัน merge2() จัดลงอิงปะกอบจากซ้ายหาขวาอยู่ในช่วง low และ high โดยอิงใส่ค่า mid.

```
1  def merge2(S, low, mid, high):
2      R = []
3      i, j = low, mid + 1
4      while i <= mid and j <= high:
5          if S[i] < S[j]:
6              R.append(S[i]); i += 1
7          else:
8              R.append(S[j]); j += 1
9      if i > mid:
10         for k in range(j, high + 1):
11             R.append(S[j])
12     else:
13         for k in range(i, mid + 1):
14             R.append(S[i])
15     for k in range(len(R)):
16         S[low + k] = R[k]
```

Line 9~11

- เมื่อออกจาก while-loop ของ Line 3-8, i จะใหย่กว่า mid หรือ j จะใหย่กว่า high, ตามเงื่อนไขของ Line 4.
- ถ้า i ใหย่กว่า mid, อิงปะกอบที่ยังเหลือต้องถูกเพิ่มใส่ R จนก่อาดัดชะนิของ j เป็น high.

2. ການສ້າງການຈັດລຽງແບບ Merge ທີ່ມີປະສິດທິພາບ

2.2. ລວມສອງລາຍການທີ່ຈັດລຽງແລ້ວເພື່ອປັບປຸງ merge sort

ฟังชัน merge2() จัดลงิงอิงปะกอบจากซ้ายหาขวาอยู่ในช่วง low และ high โดยอิงใส่ค่า mid.

```
1  def merge2(S, low, mid, high):
2      R = []
3      i, j = low, mid + 1
4      while i <= mid and j <= high:
5          if S[i] < S[j]:
6              R.append(S[i]); i += 1
7          else:
8              R.append(S[j]); j += 1
9      if i > mid:
10         for k in range(j, high + 1):
11             R.append(S[j])
12     else:
13         for k in range(i, mid + 1):
14             R.append(S[i])
15     for k in range(len(R)):
16         S[low + k] = R[k]
```

Line 12~14

- เมื่อออกจาก while-loop ของ Line 3-8, i จะใหย่กว่า mid หรือ j จะใหย่กว่า high, ขึ้นกับเงื่อนไขของ Line 4.
- ถ้า i บ่ใหย่กว่า mid, อิงปะกอบที่ยังเหลือต้องถูกเพิ่มใส่ R จิมก่อาดัดชะนิของ i เป็น mid.

2. ການສ້າງການຈັດລຽງແບບ Merge ທີ່ມີປະສິດທິພາບ

2.2. ລວມສອງລາຍການທີ່ຈັດລຽງແລ້ວເພື່ອປັບປຸງການຈັດລຽງແບບ merge

ฟังชัน merge2() จัดลงอิงปะกอบจากซ้ายหาขวาอยู่ในช่วง low และ high โดยอิงใส่ค่า mid.

```
1  def merge2(S, low, mid, high):
2      R = []
3      i, j = low, mid + 1
4      while i <= mid and j <= high:
5          if S[i] < S[j]:
6              R.append(S[i]); i += 1
7          else:
8              R.append(S[j]); j += 1
9      if i > mid:
10         for k in range(j, high + 1):
11             R.append(S[j])
12     else:
13         for k in range(i, mid + 1):
14             R.append(S[i])
15     for k in range(len(R)):
16         S[low + k] = R[k]
```

Line 15~16

- ใน Line 15, เป็นการ Loop เอาค่าที่เก็บในลายการ R ไปเก็บไว้ในลายการ S คือเก่า.
- อิงปะกอบทั้งหมดจาก low หา high ของ R ถูกสำเนาไปใส่ไว้ในลายการ S.

2. ການສ້າງການຈັດລຽງແບບ Merge ທີ່ມີປະສິດທິພາບ

2.3. Run ໂປຣແກຣມການເພີ່ມປະສິດທິພາບການຈັດລຽງແບບ merge

ເມື່ອຕົວຢ່າງທີ່ໃຊ້ກ່ອນຫນ້ານັ້ນຖືກນໍາໃຊ້ກັບຟັງຊັນ mergesort2(), ຈະມີຜົນໄດ້ຮັບດັ່ງຕໍ່ໄປນີ້.

```
1 S = [27, 10, 12, 20, 25, 13, 15, 22]
2 mergesort2(S, 0, len(S) - 1)
3 print(S)
```

```
[27, 10, 12, 20, 25, 13, 15, 22]
[27, 10, 12, 20, 25, 13, 15, 22]
[27, 10, 12, 20, 25, 13, 15, 22]
[10, 27, 12, 20, 25, 13, 15, 22]
[10, 12, 20, 27, 25, 13, 15, 22]
[10, 12, 20, 27, 25, 13, 15, 22]
[10, 12, 20, 27, 13, 25, 15, 22]
[10, 12, 13, 15, 20, 22, 25, 27]
```

Line 1~3

- S ທີ່ໄດ້ຮັບໂດຍ print(S) ແມ່ນພິມຢູ່ໃນ Line 3 ຂອງຟັງຊັນ mergesort2().
- ຜົນໄດ້ຮັບ S ຈາກ Line 3 ເປັນຜົນໄດ້ຮັບທີ່ມີການຈັດລຽງແລ້ວ ຫຼັງຈາກຟັງຊັນ mergesort2() ໄດ້ຖືກປະຕິບັດ.
- ສັງເກດວ່າຂະຫນາດຂອງ S ບໍ່ປ່ຽນແປງ, ບໍ່ເໝືອນກັບຜົນໄດ້ຮັບຂອງຟັງຊັນ mergesort1().

Pop quiz

Q1. ฟังຊັນ merge2() ປะติบัดภาบในภาบจัดลฯงข้มูบลุ่มมิ้จักถั้ง?

```
S = [6, 2, 11, 7, 5, 4, 8, 16, 10, 3]
mergesort2(S, 0, len(S) - 1)
print(S)
```

```
S = [6, 2, 11, 7, 5, 4, 8, 16, 10, 3, 1, 12, 9]
mergesort2(S, 0, len(S) - 1)
print(S)
```

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. ໃນສະພາບແວດລ້ອມການເຮັດວຽກຮ່ວມກັນ, ນັກຮຽນມີສ່ວນຮ່ວມໃນການສົນທະນາ ປຶກສາຫາລືເຊິ່ງກັນແລະກັນ.

Q1. ให้ N ลายการที่ถูกจัดลงงแล้ว เป็นข้อมูลที่ใช้จัดลงง, จงงเขียนโปรแกรมที่ลอมพวกมันเข้าไปในลายการจัดลงงอันหนึ่ง.

```
N = int(input("Input the number of list: "))
list_of_nums = []
for i in range(N):
    nums = list(map(int, input("Input a list of numbers: ").split()))
    print(nums)
    list_of_nums.append(nums)
sorted = multiway_merge(list_of_nums)
print("Merged into : ", sorted)
```

```
Input the number of list: 3
Input a list of numbers: 1 5
[1, 5]
Input a list of numbers: 2 6
[2, 6]
Input a list of numbers: 3 4 7
[3, 4, 7]
Merged into :  [1, 2, 3, 4, 5, 6, 7]
```