ບົດລາຍງານ ກຸ່ມ 2 : ບົດທີ 4 Stack ແລະ Queue

ສອນໂດຍ: ອຈ ຈິດນາວັນ ຫ້ອງ 2Cw1

ສະມາຊິກກຸ່ມມີ: 9 ສະຫາຍ

1 ນາງ ນ່ອນທະລາ ສີພັນທອງ

2 ນາງ ດຽວຄຳໃຍວິດວຽງ

3 ນາງ ບົ່ວງາງ

4 ທ້າວ ຕູ້ຢ່າງ

5 ທ້າວ ສຸວັນ ດວງຄຳດີ

6 ພຣະ ມີໄຊ

7 ທ້າວ ນ້ອຍຈັນ

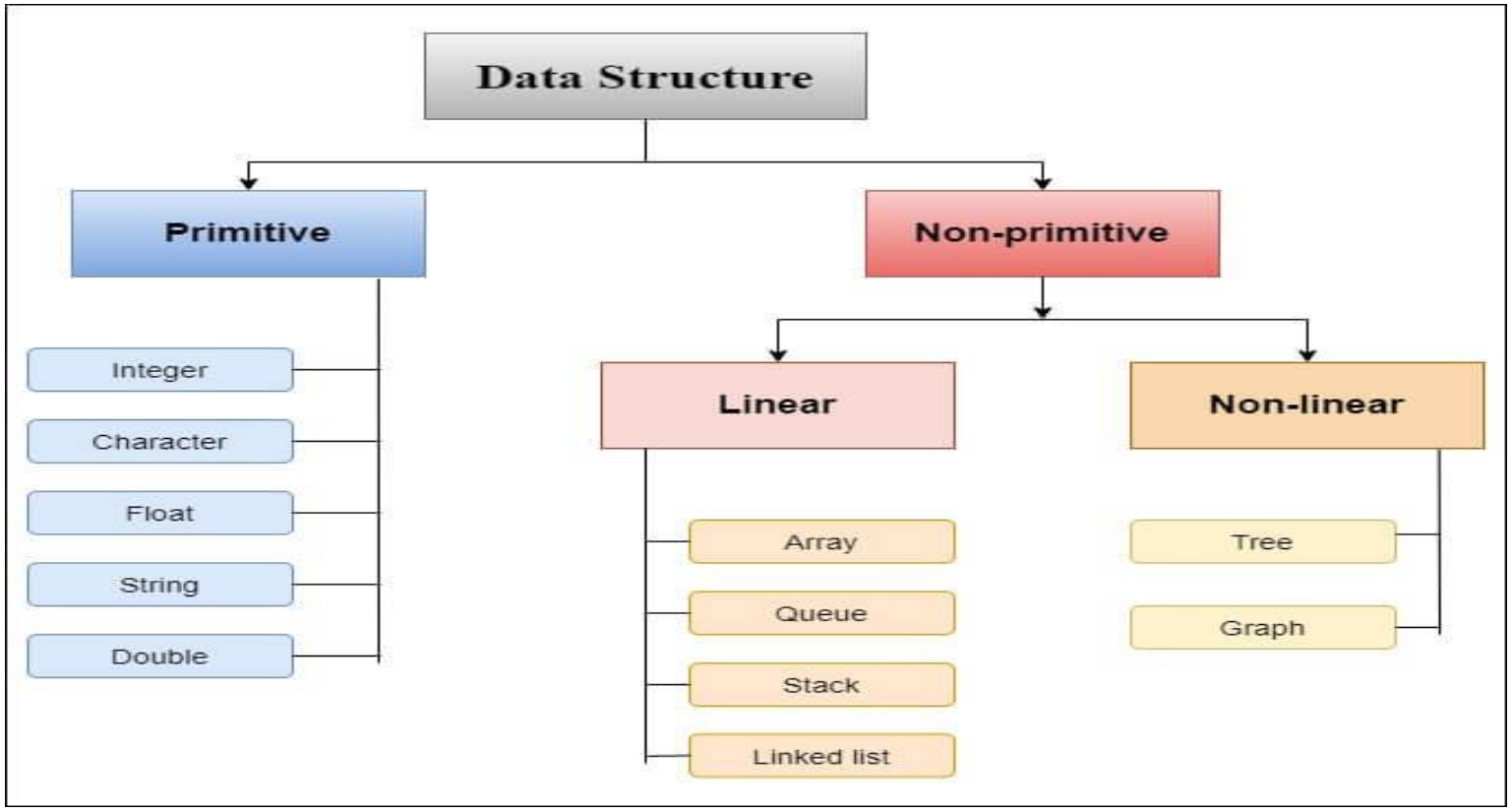
8 ທ້າວ ນູ່ຊົວ ເຮີ

9 ພຣະ ບຸນຄ້ຳ ພົມມະສຸລິນ

ขิดที่ 4 Stack ແລະ Queue

1.ທິດສະດີໂດຍ ຫຍໍ້

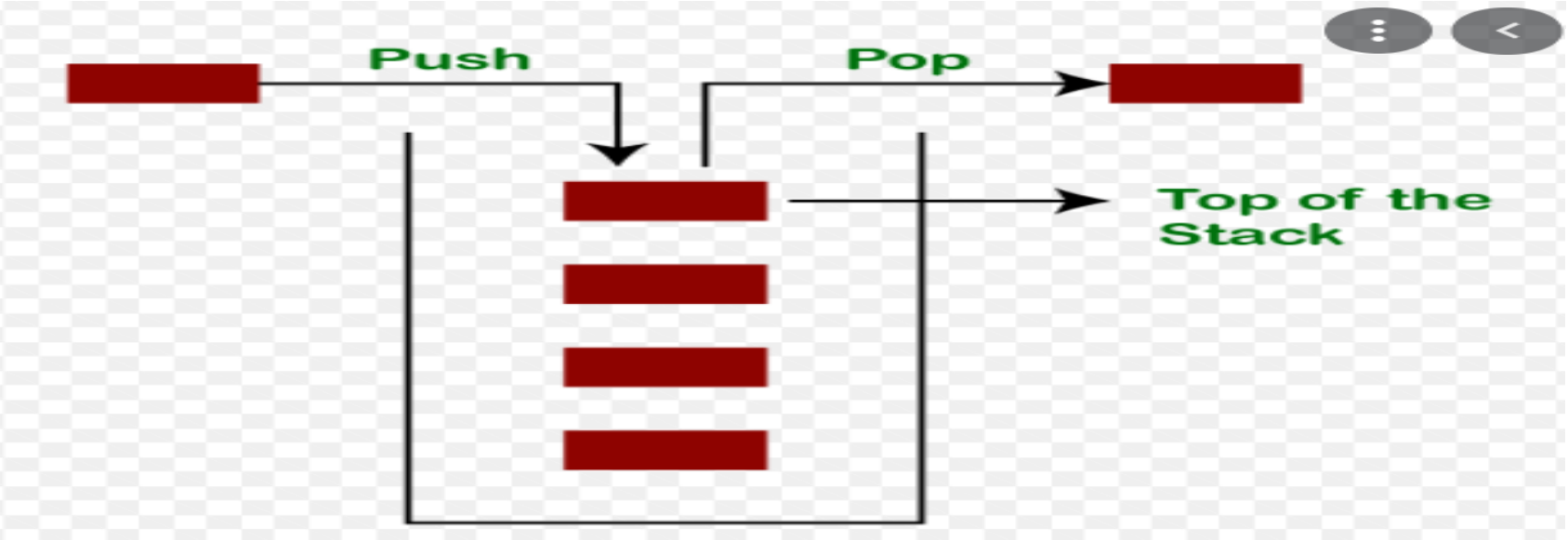
- Data Structure ຄືວ່າເປັນເຄື່ອງມືທີ່ໃຊ້ໃນການຈັດເກັບ Data store ທີ່ມີໂຄງສ້າງໃນຄອມພິວເຕີ້ ເພື່ອນຳໄປໃຊ້ງານ ໄດ້ຢ່າງມີປະສິດທິພາບ ແລະ ມີບົດບາດສຳຄັນໃນການອອກແບບ Algorithm ທີ່ດີ

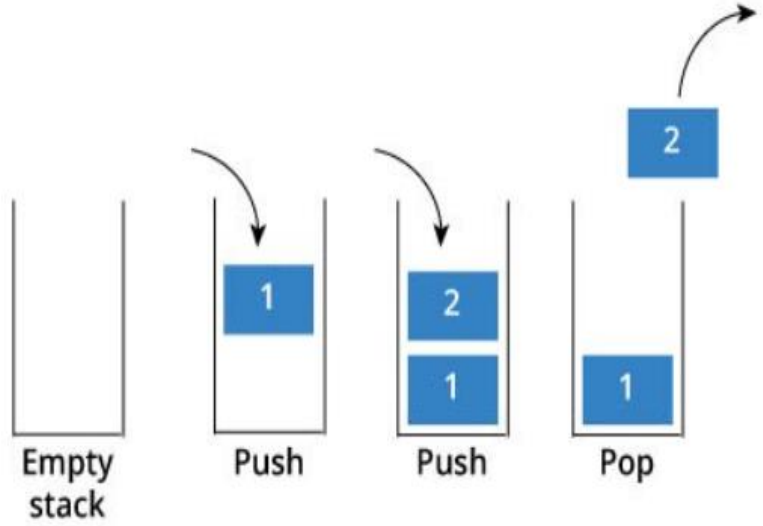


1 Stack ແມ່ນຫຍັງ?

stack ແມ່ນປະເພດຂໍ້ມູນທີ່ບໍ່ມີຕົວຕົນເຊິ່ງເຮັດ ໜ້າ ທີ່ເກັບ ກຳ ຂໍ້ມູນຂອງອົງປະກອບ, ໂດຍມີສອງປະຕິບັດ ການຫລັກຄື: Push, ເຊິ່ງເພີ່ມອົງປະກອບເຂົ້າໃນການເກັບ ກຳ ຂໍ້ມູນ, ແລະ. Pop, ເຊິ່ງ ກຳ ຈັດອົງປະກອບທີ່ ເພີ່ມເຂົ້າມາ ໃໝ່ ທີ່ສຸດທີ່ຍັງບໍ່ໄດ້ເອົາອອກເທື່ອ.

ຮູບພາບຕົວຢ່າງ





2. INFIX ແລະ POST FIX ແມ່ນຫຍັງ?

ແມ່ນລຳດັບການເຮັດວຽກຂອງຕົວດຳເນີນການທາງຄະນິດສາດ ຈາກສູງໄປຫາຕ່ຳ
ໃນນີ້ການ ແປງ infix ເປັນ postfix ເຮົາຕ້ອງຮູ້ລຳດຳຕົວດຳເນີນການມາກ່ອນ

1 ເຄື່ອງໝາຍ () (ບໍ່ແມ່ນຕົວດຳເນີນການ)

2 ຕົວດາເນີນການຍົກກຳລັງ (^)

3 ຕົວດຳເນີນການຄູນ (*) ແລະ ຫານ (/)

4 ຕົວດຳເນີນການບວກ (+) ແລະ ລົບ (-)

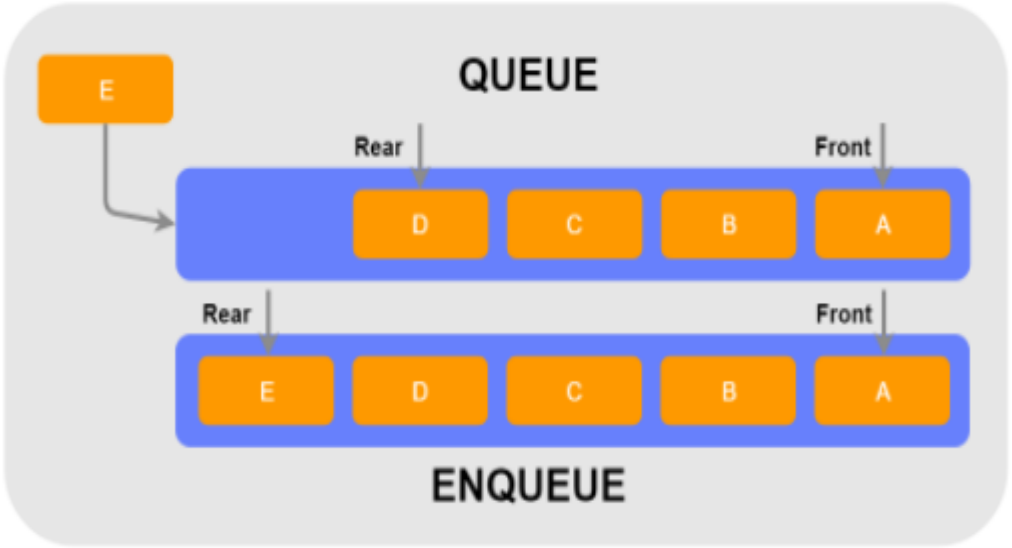
ตัวอย่าง

A+B+C หมายถึง (A+B)+C

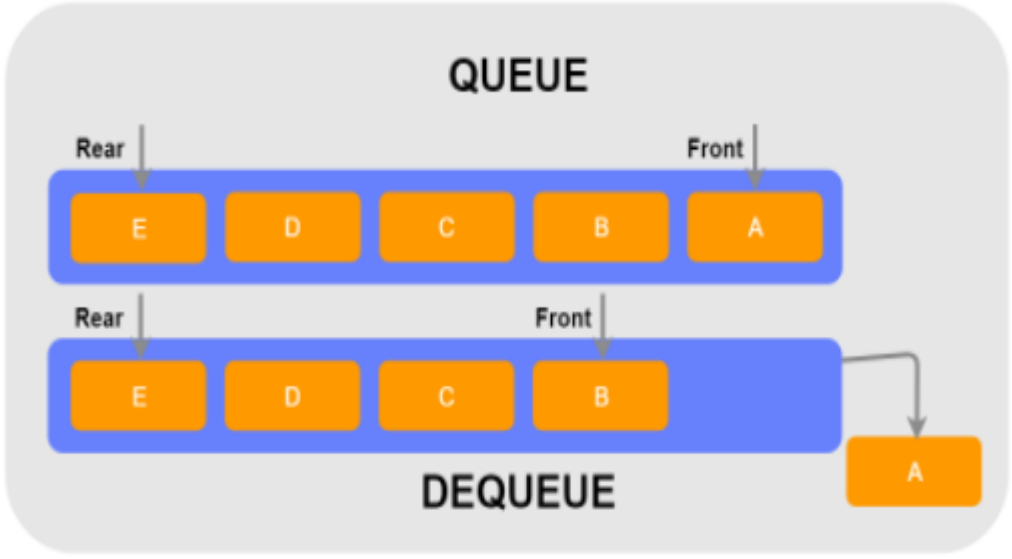
A^B^C หมายถึง A^(B^C)

3. ຄິວ QUEUE ແມ່ນຫຍັງ?

Queue ເປັນ Linear Data structure ເຊິ່ງຈະຄ້າຍຄືກັບ Stack ແຕ່ການ ບໍ່ຄືຫມົດດອກ ຄິວ ຈະເປີດ ປາຍທາງຫົວ ແລະ ທ້າຍ ດ້ານຫນຶ່ງຈະໃຊ້ໃນການນຳເຂົ້າຂໍ້ມູນ (Enqueue) ແລະ ດ້ານ ຫນຶ່ງແມ່ນມາຂໍ້ ມູນອອກ (Dequeue)



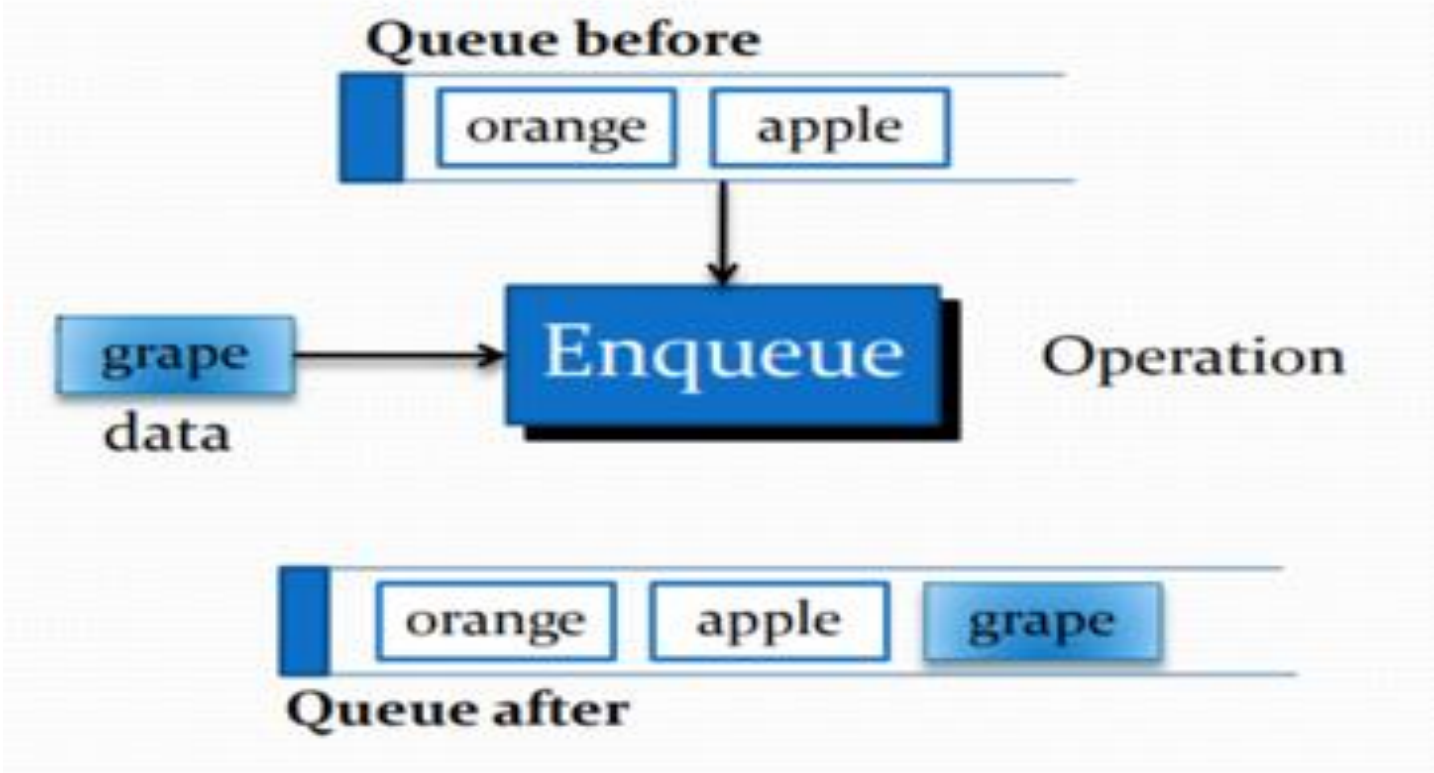
ENQUEUE



DEQUEUE

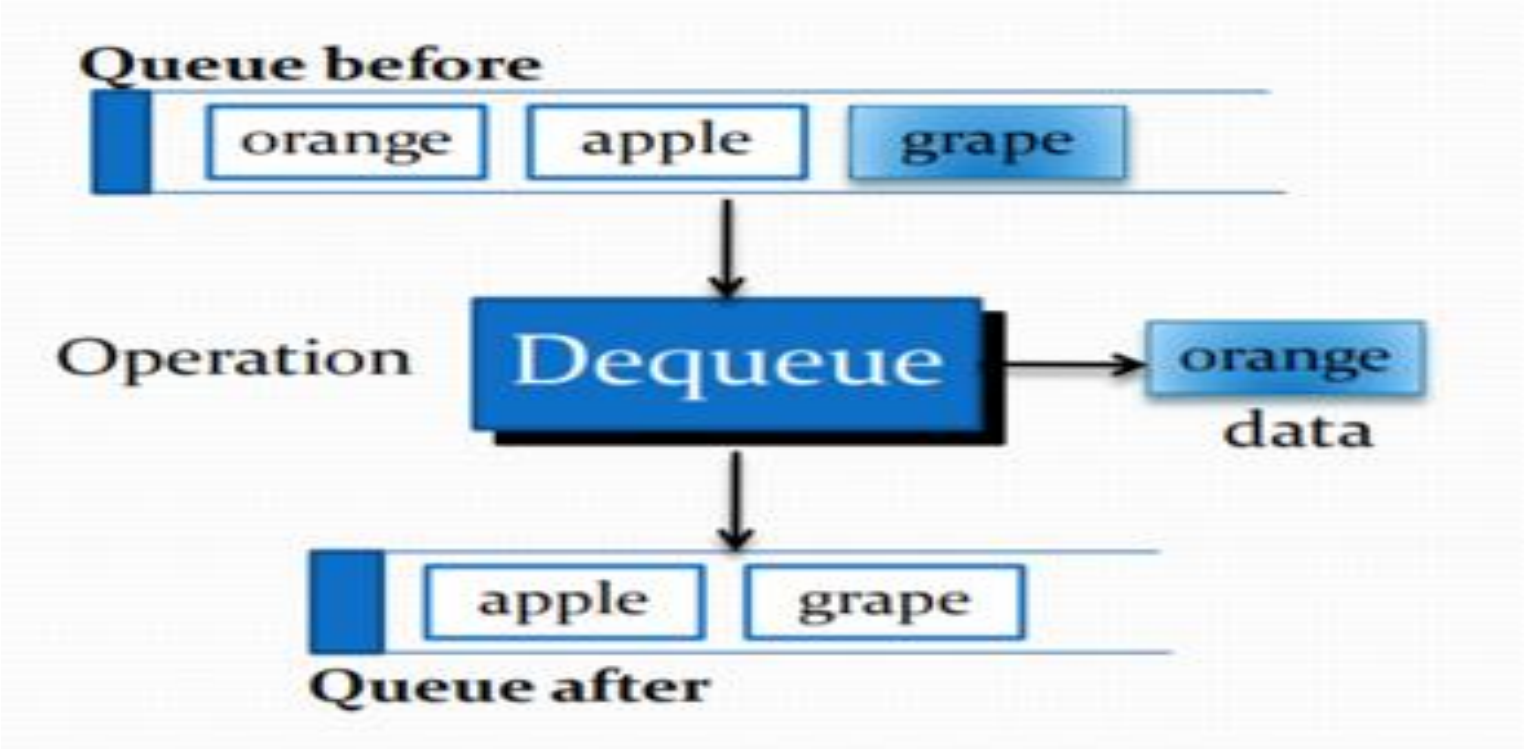
4.ฟังຊັ່ນ ENQUEUE

ເປັນຟັ່ງຊັ່ນພື້ນຖານຂອງໂຄງສ້າງຂໍ້ມູນແບບຄິວ ເຫມືອນກັບການທີ່ເຮົາໄປເຂົ້າແຖວຕໍ່ຊື້ເຄື່ອງ ໃຜຊື້ກ່ອນກະອອກຮ້ານໄປກ່ອນຕໍ່ຕ່ານກັນໄປຈົນຈົບ



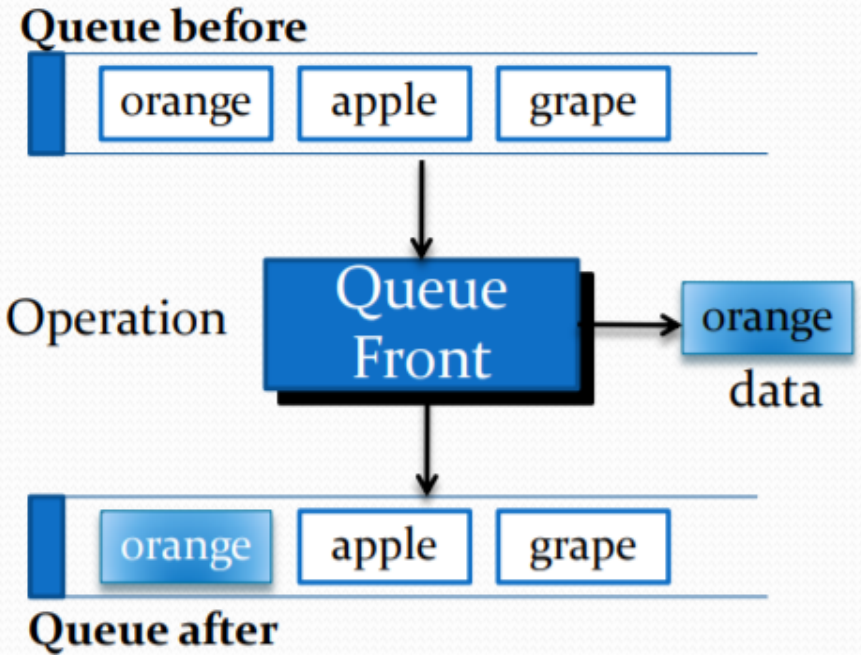
5.ฟังຊັ່ນ DEQUEUE

ເປັນຟັ່ງຊັ່ນການນຳຂໍ້ມູນອອກຈາກຄິວ ເຊິ່ງມີການເຮັດວຽກກົງກັນຂ້າມກັບ Enqueue ເມື່ອງ ຈາກຟັ່ງຊັ່ນ Dequeue ຈະນຳຂໍ້ມູນອອກທາງດ້ານ front



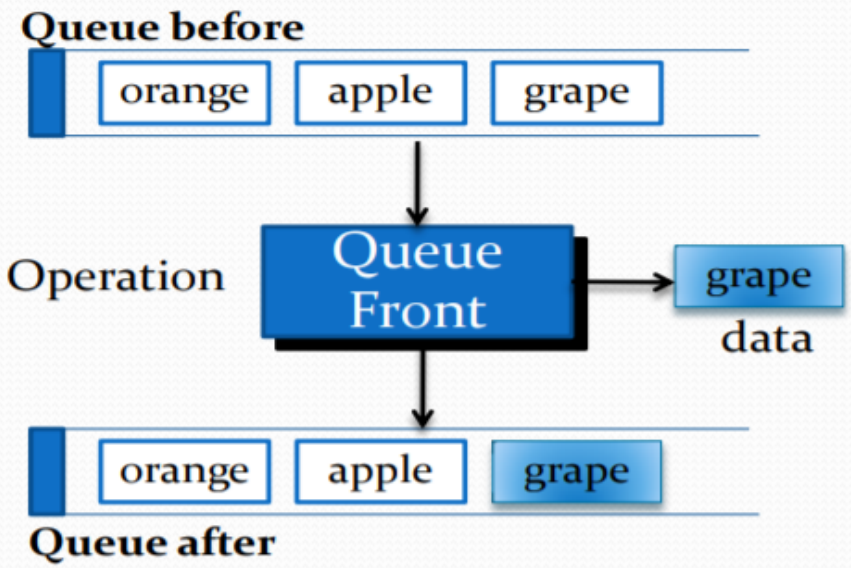
6.ฟังชัน QUEUE FRONT

Queue Front ຈະທຳການຕົ້ງຂໍ້ມູນສະມາຊິກໂດຍກົງຍອກມາໃຊ້ງານຫາກກາຍໃນຄິວບໍ່ມີຂໍ້ມູນ ກໍຈະທຳໃຫ້ເກີດສະຖານະ Underflow



7.ฟังຊັ່ນ QUEUE REAR

Queue Rear ຈະທຳການຕົ້ງຂໍ້ມູນສະນາຊິກໂດຍກົງຈອງການໃຊ້ງານ ຫາກການໃນຄິວບໍ່ມີຂໍ້ມູນກໍຈະທຳໃຫ້ເກີດສະຖານະ Underflow



8. ABSTR DATA TYPE ຂອງ QUEUE (QUEUE ADT)

ປະເພດຂໍ້ມູນແບບຫຍໍ້ແມ່ນ ຄໍາ ນິຍາມຂອງປະເພດ ໃໝ່ ທີ່ອະທິບາຍກ່ຽວກັບ ຄຸນສົມບັດແລະການ ດໍາ ເນີນງານຂອງມັນ. ໂຄງສ້າງຂໍ້ມູນແມ່ນການຈັດຕັ້ງປະຕິບັດ ADT, ຫຼາຍ ADT ສາມາດ ນໍາ ໃຊ້ກັບໂຄງສ້າງຂໍ້ມູນດຽວກັນ.ຖ້າຂ້ອຍຄິດວ່າມັນ ຖືກຕ້ອງ, ຂບວນແມ່ນ ADT, ໝາຍ ຄວາມວ່າເປັນການລວບລວມຍົກປະກອບ, ແລະມັນແມ່ນໂຄງສ້າງຂໍ້ມູນໃນການເກັບຮັກສາໃນຄວາມຊົງ ຈໍາ. ແຕ່ພວກເຮົາ ສາມາດເວົ້າກ່ຽວກັບໂຄງສ້າງຂອງຂໍ້ມູນ stack ຖ້າຂ້ອຍຫມາຍຄວາມວ່າບໍ? ແລະ ເປັນຫຍັງ heap ຈຶ່ງບໍ່ແມ່ນ ADT ມັນສາມາດໃຊ້ເປັນຕົ້ນໄມ້ຫລືຂບວນ.

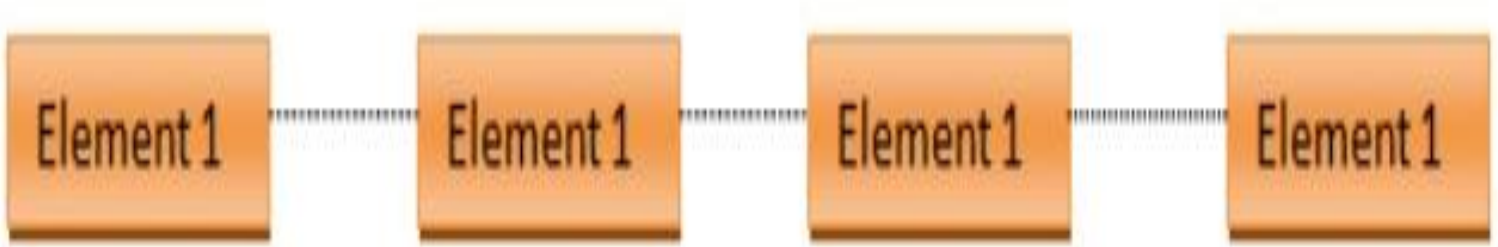
9.ການອອກແບບຄິວດ້ວຍ ARRAY (QUEUE ARRAY DESIGN)

ສໍາ ລັບໂຄງສ້າງຂໍ້ມູນແຕ່ລະແຖວ, ພວກເຮົາ ຈໍາ ເປັນຕ້ອງເກັບຄ່າຄຸນຄ່າຕໍ່ໄປນີ້:
1.Queue [] ອາເລທີ່ເກັບສະມາຊິກແຕ່ລະຄົນໃນແຖວ.
2.ດ້ານໜ້າ ຕໍາ ແໜ່ງ ຂອງຫົວທ້າຍຂອງແຖວ.
3.ຕໍາ ແໜ່ງ ຂອງທ້າຍແຖວ.
4.size ຈໍາ ນວນສະມາຊິກທັງ ໝົດ ທີ່ຢູ່ໃນແຖວ.

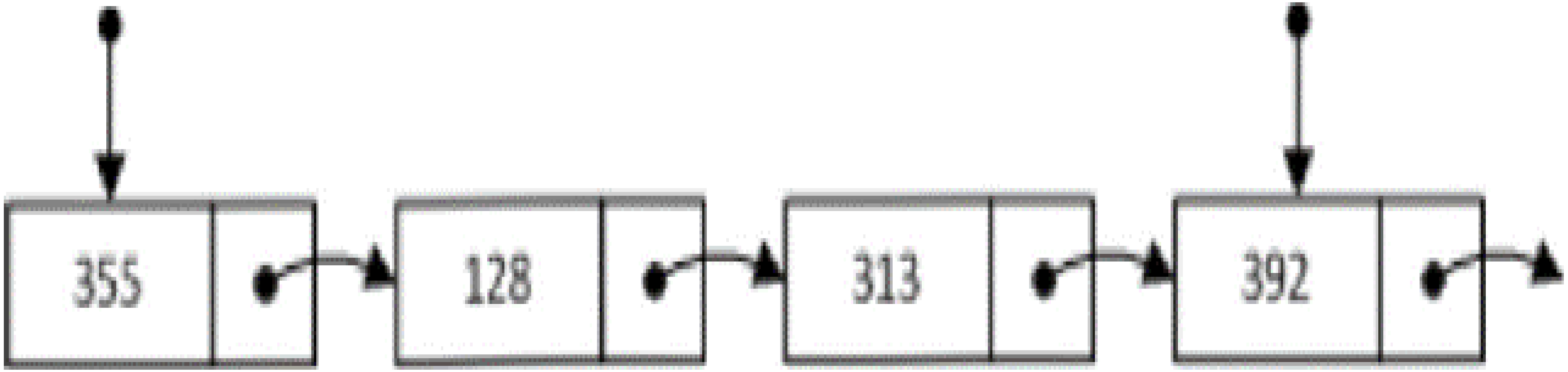
10. ແນວຄິດພື້ນຖານກ່ຽວກັບລາຍການແບບເສັ້ນຊື່ (LINEAR LIST CONCEPTS)

ການຈັດຮຽງ ລຳ ດັບຂອງຂໍ້ມູນພາຍໃນບັນຊີທີ່ເປັນ ລຳ ດັບ. ສາມາດອະທິບາຍໄດ້ສະມາຊິກແຕ່ລະຄົນຫຼືອົງປະກອບໃດ ໜຶ່ງ ເຊື່ອມໂຍງກັບອົງປະກອບຕໍ່ໄປເປັນລາຍຊື່ຕໍ່ເນື່ອງ.

ຕົວຢ່າງ



11.ລາຍການເຊື່ອມຕໍ່ແບບເສັ້ນຊື່ດ່ຽວ (SINGLE LINKED LIST)



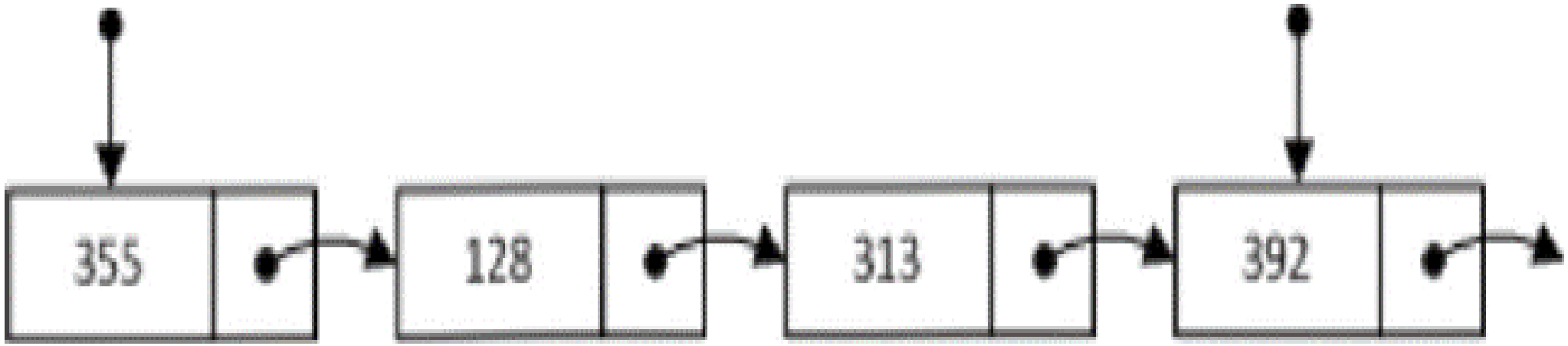
ການດໍາເນີນງານພື້ນຖານຂອງລີສ (Basic Operations)

ລາຍຊື່ການປະຕິບັດງານຂັ້ນພື້ນຖານ ສັບປະກອບດ້ວຍການແຊກ (ການແຊກ), ການລຶບ (ລຶບ). ການອ່ານ (ດຶງຂໍ້ມູນ) ແລະຜ່ານລາຍຊື່ (Traversal) ໂດຍການໃສ່ແຜ່ນການເພີ່ມສະມາຊິກ.

12. ລາຍການແບບເສັ້ນຊື່ LINKED LIST
ລິ້ງລິດທາງດຽວ (SINGLY LINKED LIST)

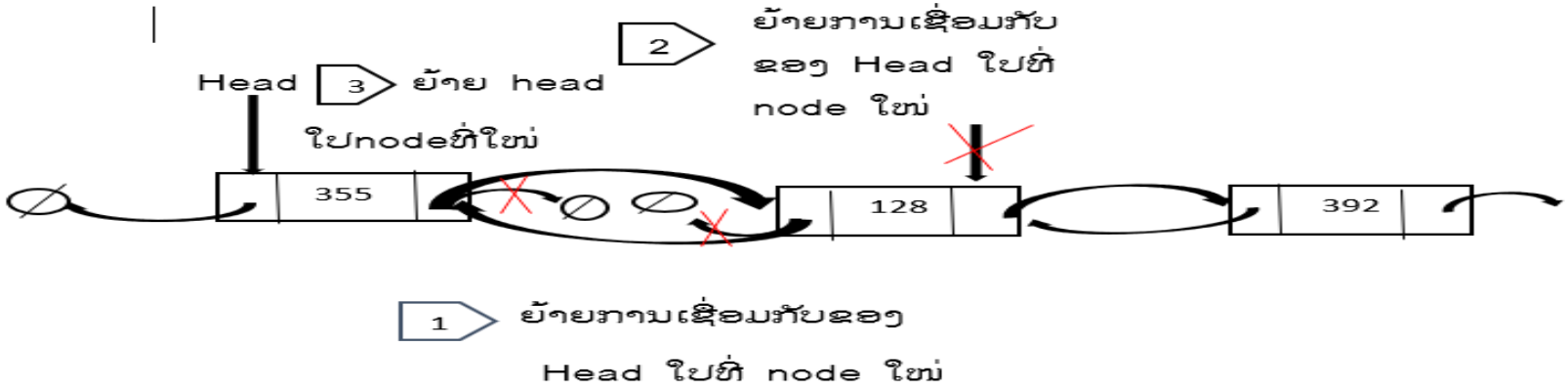
ເປັນໂຄງສ້າງທີ່ໃຊ້ node, ໃຫ້ເຊື່ອມຕໍ່ເຊິ່ງກັນແລະກັນເປັນຂັນຊີລາຍຊື່, ເຊິ່ງແຕ່ລະ node ມີສ່ວນແມ່ນຂໍ້ມູນ (ຂໍ້ມູນ) ແລະສ່ວນ ໜຶ່ງ ທີ່ເຊື່ອມຕໍ່ກັບ node ຕໍ່ໄປ (link)

ຕົວຢ່າງ 1:



13.ການເພີ່ມເຂົ້າ

- ການນຳເຂົ້າສູ່ Doubly linked list ທັງສອງທາງເຮັດໃນລັກສະນະດຽວກັນຄືສ້າງໂນດໃຫມ່ ສຳຫລັບຂໍ້ມູນທີ່ຕ້ອງການ ນຳເຂົ້າຍ້າຍຕົວເຊື່ອມຕ່າງໆ ໄປຍັງຕຳແຫນ່ງທີ່ເຫມາະສົມການນຳເຂົ້າໃນກໍລະນີບໍ່ມີຂໍ້ມູນຢູ່ໃນລິດສນັ້ນ ສິ່ງທີ່ເຮົາຕ້ອງເຮັດຄືຍ້າຍ ຫາງ ມາໃສ່ ໂນດສ ໃຫມ່ແຕ່ຖ້າມີຂໍ້ມູນຢູ່ເຮົາຕ້ອງຍ້າຍຕົວເຊື່ອມກັບຂອງ ສ່ວນຫົວມາທີ່ ໂນດ ໃຫມ່ນີ້ຂັ້ນຕອນທີ່ເຫລືອຂອງທັງສອງ ກໍລະນີຄືການເຊື່ອມຕົວຊ້ຶ້ຂອງໂນດ ໃຫມ່ໄປທີ່ຫົວ ພ້ອມກັບການຍ້າຍຫົວ ໄປທີ່ໂນດ

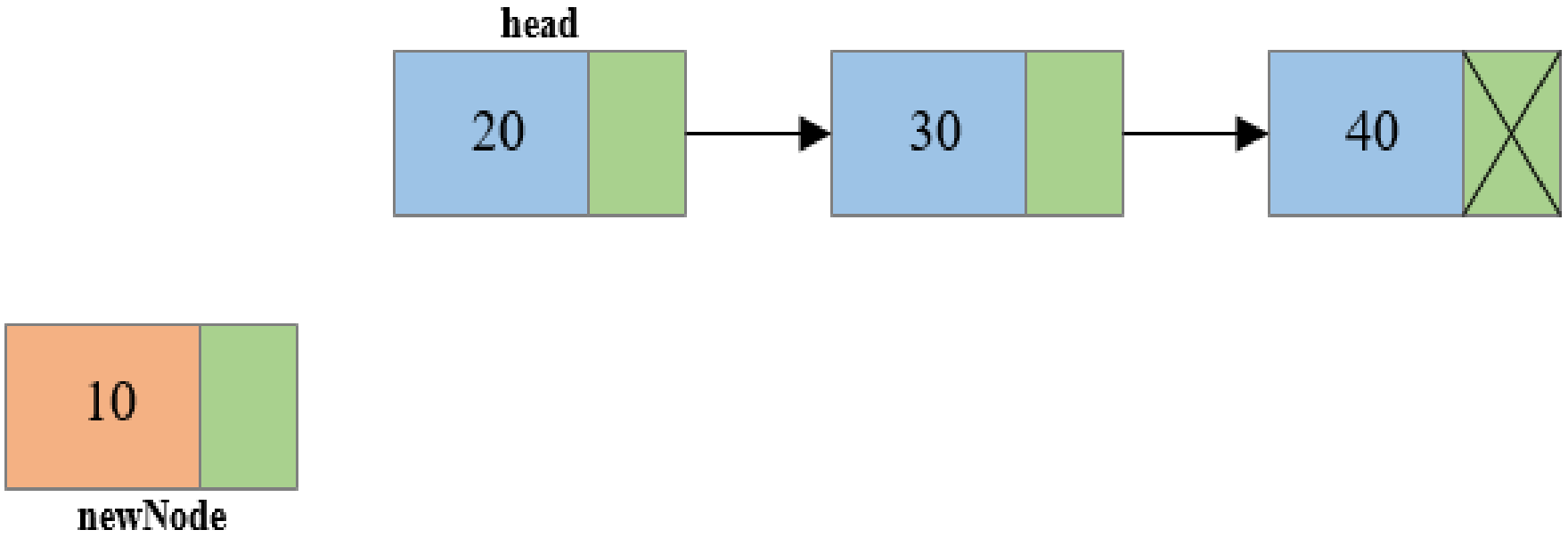


14.ตัวอย่าง :

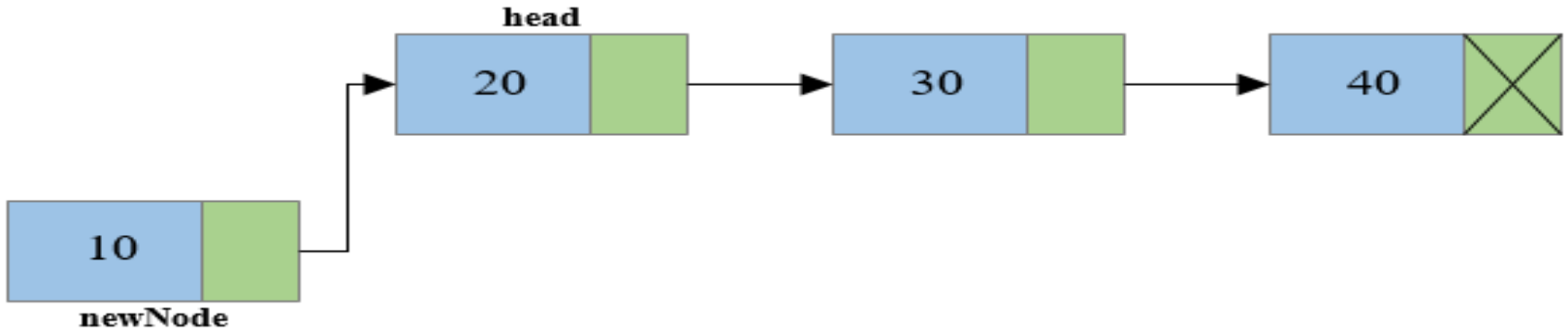
```
01: ALGORITHM INSERTATHEAD(VALUE)
02: BEGIN
03: NEWNODE ⇓NEW NODE(VALUE)
04: IF HEAD = NULL
05: TAIL ⇓ NEWNODE
06: ELSE
07: HEAD.PREVIOUS ⇓ NEWNODE
08: END IF
09: NEWNODE.NEXT ⇓ HEAD
10: HEAD ⇓ NEWNODE
11: END INSERTAFTER
```

15.ການເພີ່ມເຂົ້າໂນດທີ່ຕຳແໜ່ງຕ່າງ(Insert at Beginning)

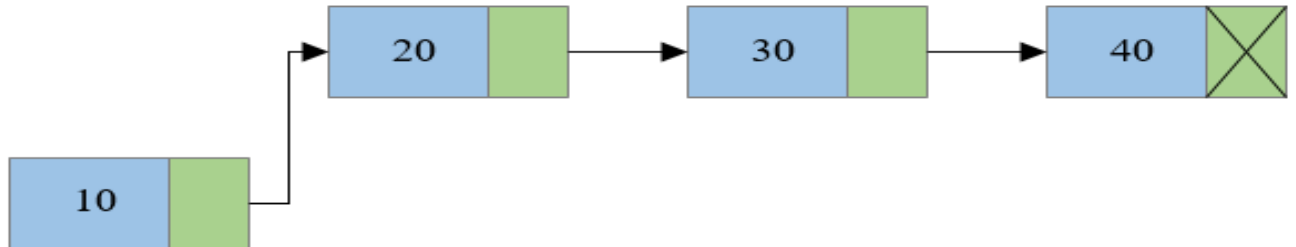
ຂັ້ນຕອນໃນການແທຣກໂທນດໃຫມ່ທີ່ຈຸດເລີ່ມຕົ້ນຂອງລາຍການທີ່ເຊື່ອມໂຍງ.
ສ້າງໂທນດໃຫມ່ຊີ້ໄປທີ່ໂທນດສ້າງຂຶ້ນໃຫມ່.



16. ເຊື່ອມໂຍງໂຫນດທີ່ສ້າງຂຶ້ນໃຫມ່ກັບໂຫນດຫລັກເຊັ່ນ: ໂຫນດໃຫມ່ຈະຊີ້ໄປທີ່ຫົວໂຫນດ.

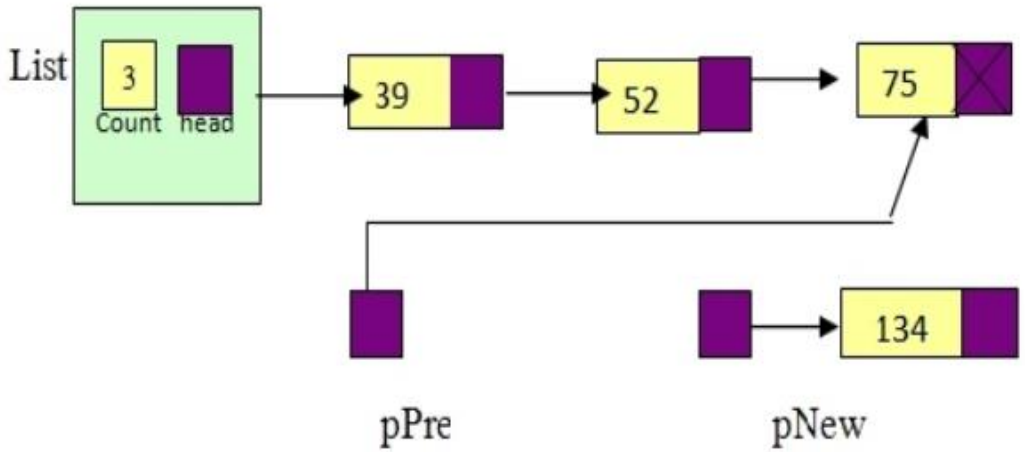


ສ້າງໂຫນດໃຫມ່ເປັນໂຫນດຫຼັກເຊັ່ນ: ຫົວໂຫນດຈະຊີ້ໄປໂຫນດໃຫມ່.



17 ການເພີ່ມເຂົ້າໂນດທາງທ້າຍ(Insert at End)

ເມື່ອ node ຖືກເພີ່ມໃສ່ໃນຕອນທ້າຍຂອງບັນຊີ ພວກເຮົາພຽງແຕ່ຕ້ອງການຕົວຊີ້ບອກກ່ອນ.
ເພື່ອຊີ້ໄປທີ່ຂໍ້ ໃໝ່ ໃນນັ້ນບໍ່ມີ node Successor ເພາະວ່ານັ້ນຖືກໃສ່ໃນຕອນທ້າຍບັນຊີລາຍຊື່, ດັ່ງນັ້ນພາກສະຫນາມເຊື່ອມຕໍ່ຂອງໂຫນດ ໃໝ່ ຖືກ ກໍາ ນົດໃຫ້ບໍ່ມີ ປະໂຫຍດ
ເຖິງຢ່າງໃດກໍ່ຕາມ, ນັ້ນມີອີກປະເພດ ໜຶ່ງ ຂອງເຫດຜົນພິເສດເພື່ອ ນໍາ ໃຊ້ກັບລະບົບການແຊກຂອງຂໍ້ມູນໃນຕອນທ້າຍຂອງບັນຊີ, ເຊິ່ງແມ່ນ ໜຶ່ງ ໃນຂໍ້ດີຂອງໂຄງສ້າງ ບັນຊີລາຍຊື່ການເຊື່ອມໂຍງ.



18.ການລຶບໂນດ (Delete Node)

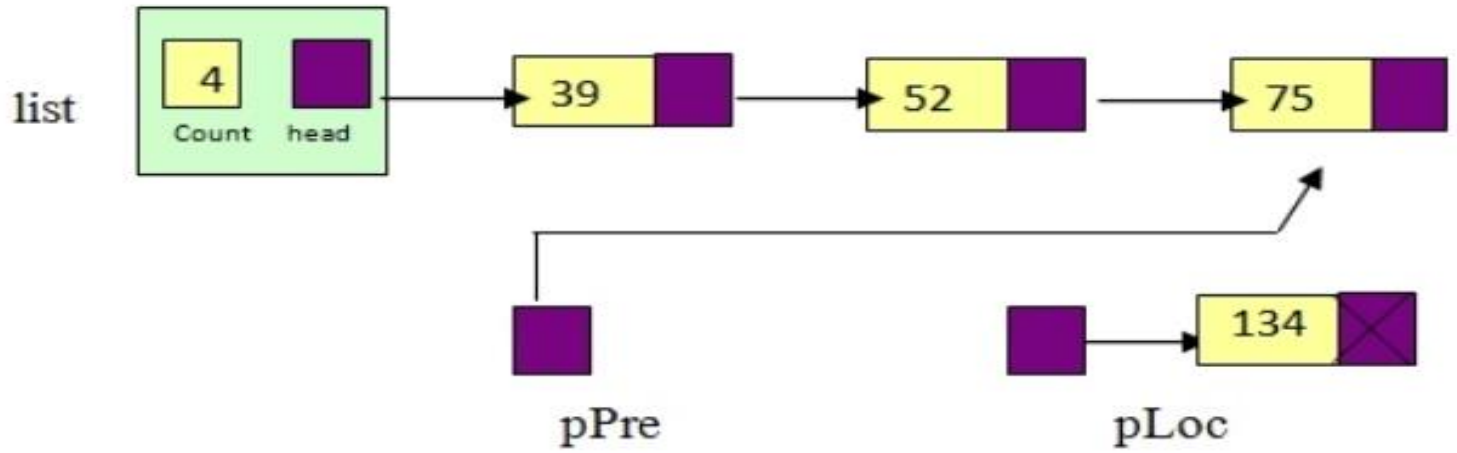
ການລຶບຂໍ້ມູນອອກຈາກ LIST ທາງດ້ານຫນ້າເຮົາຕ້ອງຄຳນຶງເຖິງຫນຶ່ງກໍລະນີ LIST ບໍ່ຂໍ້ມູນສອງ ກໍລະນີ LIST ມີຂໍ້ມູນພຽງຫນຶ່ງໂຕແລະ 3 ກໍລະນີ LIST ມີຂໍ້ມູນຫລາຍກວ່າຫນຶ່ງໂຕ ກໍລະນີທີ່ຫນຶ່ງ ແລະ ທີ່ສອງ ນັນງ່າຍກວ່າເຮົາພຽງແຕ່ກຳນົດໃຫ້ HEAD ແລະ TAIL ມີຄ່າເປັນ NULL ກໍລະນີທີ່ສາມ ເຮົາຍ້າຍຕົວເຊື່ອມກັບຂອງ NODE ທີ່ຢູ່ຕັດຈາກ HEAD (ດ້ານ) ໄປທີ່ NULL ແລະ ຍ້າຍ HEAD ໄປທີ່ NODE ນີ້ຫລັງຈາກນັ້ນດັ່ງ ຮູບລຸ່ມນີ້

ການລຶບໂນດທີ່ຕຳແໜ່ງທຳອິດ (Delete first Node)

ການລຶບຂໍ້ມູນອອກຈາກ linked list ທາງດ້ານຫນ້ານັ້ນເຮັດໄດ້ງ່າຍກວ່າທາງດ້ານຫລັງເຮົາພຽງແຕ່ ຍ້າຍ head ໃຫ້ຊີ້ໄປທີ່ node ຕໍ່ໄປກໍແລ້ວແຕ່ການລຶບອອກຈາກທາງດ້ານຫລັງ ນັ້ນເຮົາຕ້ອງຊອກຫາ node ທີ່ຢູ່ກ່ອນ tail ໃຫ້ເບິ່ງເພື່ອທີ່ຈະຍ້າຍຕົວເຊື່ອມຈາກ node ນີ້ໄປທີ່ null ເຊິ່ງກໍ່ຫມາຍເຖິງການນຳເຂົ້າຫາທຸກ node ທີ່ຢູ່ໃນ list ໃຊ້ເວລາຫລາຍຖ້າຈຳນວນ node ມີຫລາຍ

19.ການລຶບໂນດທີ່ຕຳແໜ່ງສຸດທ້າຍ (Delete at end)

ການລຶບຂໍ້ຈອງຈາກບັນຊີລາຍຊື່ໃນກໍລະນີທົ່ວໄປ ນີ້ປະກອບດ້ວຍໃນການຖອດຂໍ້ທີ່ຈອງເປັນຈຸດສູນກາງ. ຢູ່ໃນລາຍຊື່ແລະລຶບຂໍ້ສຸດທ້າຍຂອງບັນຊີ ໃນທັງສອງກໍລະນີ, ເຫດຜົນດຽວກັນສາມາດຖືກ ນຳ ໃຊ້ເພື່ອຈັດຕັ້ງປະຕິບັດ ລຶບ NODE ກຳ ຈັດຂໍ້ມູນຈອງຈາກບັນຊີ. NODE ແມ່ນຢູ່ໃນບັນຊີກາງຫຼືຢູ່ໃນຕອນທ້າຍຂອງບັນຊີ.



20. ລາຍການເຊື່ອມຕໍ່ແບບເສັ້ນຊື່ ຊະນິດອື່ນໆ(Others Linked Lists)

ຄວາມຮູ້ກ່ຽວກັບລາຍຊື່ LINK ທີ່ກ່າວມາຂ້າງເທິງແມ່ນບັນຊີລາຍຊື່ LINK ດຽວ. (ບັນຊີລາຍຊື່ທີ່ເຊື່ອມ
ໂຍງດຽວ) ເພາະວ່າບັນຈຸມີພຽງແຕ່ LINK ດຽວທີ່ຊີ້ໄປທີ່ NODE ຕໍ່ໄປ.

ລາຍການເຊື່ອມຕໍ່ແບບວົງມົນ (Circular-linked List)

ປະເພດຂອງບັນຊີລາຍຊື່ການເຊື່ອມຕໍ່ນີ້ ເກີດມາຈາກການປັບປຸງມູນຄ່າບັນຊີການເຊື່ອມຕໍ່ ສໍາລັບການປຸງແຕ່ງທີ່ດີກ
ວ່າ ໂດຍການທົດແທນການເຊື່ອມຕໍ່ທີ່ເປັນNULL ຂອງ node ສຸດທ້າຍຂອງບັນຊີລາຍຊື່ການເຊື່ອມຕໍ່ກັບທີ່ຢູ່ຂອງ node
ທໍາ ອິດ.

ການເຊື່ອມຕໍ່ບັນຊີລາຍຊື່ວົງ ມີປະໂຫຍດຫຼາຍກ່ວາບັນຊີລາຍຊື່ການເຊື່ອມໂຍງທີ່ງ່າຍດາຍ.

1. ເພື່ອເຂົ້າເຖິງຂໍ້ມູນ ຂອງຂໍ້ມູນທັງ ໝົດ nodes circle link nodes ສາມາດເຂົ້າເບິ່ງໄດ້ຈາກ node ໃດກໍ່ໄດ້ທີ່
ໃຫ້ຜ່ານລະບົບຕ່ອງໂສ້ (link) ຂອງລາຍການ
2. ເພື່ອລຶບ node, ເພື່ອຄົ້ນຫາ node ທີ່ຢູ່ເບື້ອງຕົ້ນຂອງ node ໃດ ໜຶ່ງ, ການຄົ້ນຫາສາມາດລິເລີ່ມໃນ node ນັ້ນໄດ້.

21.ການເພີ່ມເຂົ້າ node ທາງດ້ານໜ້າ

ເພື່ອ ນຳ ເອົາຂໍ້ມູນໄປທາງ ໜ້າ ຂອງ Linked-List, ພວກເຮົາ ຈຳ ເປັນຕ້ອງຊອກຫາຂໍ້ມູນ.ເພື່ອເຮັດໃຫ້ມັນຮ້າຍແຮງກວ່າເກົ່າ, ພວກເຮົາສາມາດເຮັດສິ່ງນີ້ໄດ້ໂດຍການໃຊ້ node ປັດຈຸບັນເພື່ອເຮັດ ໜ້າ ທີ່ເປັນ deni.

ການເພີ່ມເຂົ້າ Rear node

ວິທີການສະກັດເອົາຂໍ້ມູນຈາກ NODEພື່ອ ນຳ ເອົາການ ນຳ ໃຊ້ນັ້ນຈອກມາ ຈະເລີ່ມຕົ້ນໂດຍການຊອກຫາຂໍ້ມູນຈາກສະຖານທີ່ຂໍ້ມູນພາຍໃນບັນຊີ, ຖ້າພົບ.ຂໍ້ມູນທີ່ຕ້ອງການ ມັນຈະຍ້າຍຂໍ້ມູນໄປສູ່ພື້ນທີ່ຜົນຜະລິດຂອງໂມດູນການປະຕິບັດ.ແລະຈະສົ່ງຄ່າເຫດຜົນກັບຄືນສູ່ຄວາມຈິງ ແຕ່ຖ້າບໍ່ພົບມັນຈະສົ່ງມູນຄ່າເຫດຜົນທີ່ບໍ່ຖືກຕ້ອງໃຫ້ PSEUDOCODE ເພື່ອດຶງຂໍ້ມູນຈາກ NODES ພາຍໃນບັນຊີ.

22.ລາຍການເຊື່ອມຕໍ່ແບບເສັ້ນຊື່ຄູ່ (Doble Linked List)

ໃນບັນຊີລາຍຊື່ການເຊື່ອມໂຍງແບບນີ້ມີ node ປະກອບມີສອງລາຍຊື່ link ເພື່ອເປັນຕົວແທນຂອງຜູ້ ນໍາ ກ່ອນແລະແຫຼ່ງຂໍ້ມູນ source.
ຕໍ່ມາ, node ທີ່ຢູ່ເບື້ອງຕົ້ນຂອງພວກເຮົາຖືກເອີ້ນວ່າການເຊື່ອມຕໍ່ຊ້າຍ, ເຊິ່ງເປັນຕົວແທນໂດຍຕົວຊີ້ LLINK ແລະຕົວເຊື່ອມຕໍ່ທີ່ເປັນຕົວແທນຂອງ node source. ອັນທີ່ເອີ້ນວ່າການເຊື່ອມຕໍ່ທີ່ຖືກຕ້ອງ, ເຊິ່ງສະແດງໂດຍຕົວຊີ້ R R, ເຊິ່ງແມ່ນບັນຊີລາຍຊື່ການເຊື່ອມໂຍງທີ່ມີຄຸນສົມບັດ.

ພວກເຮົາເອີ້ນວ່າ "ບັນຊີລາຍຊື່ເສັ້ນລວດຄູ່" ຫຼື "ເສັ້ນຄູ່

ການເພີ່ມເຂົ້າໂນດໃນ (DOUBLE LINKED LIST)

ການສະແດງກິ່ງງ່າຂອງຂໍ້ໃນບັນຊີລາຍຊື່ການເຊື່ອມຕໍ່ຄູ່ແມ່ນຖືວ່າເປັນໄປໄດ້ໃນກໍລະນີຕໍ່ໄປນີ້:

1. ໃນເວລາທີ່ລາຍຊື່ການເຊື່ອມຕໍ່ແມ່ນຫວ່າງ ແທນທີ່ຈະ, ໃຫ້ ກໍາ ນົດຕົວຊີ້ L ແລະຕົວຊີ້ R ໄປທີ່ ຕໍາ ແໜ່ງ node ໃໝ່.
ແລະ ກາ ນົດລີ້ງເບື້ອງຊ້າຍແລະລີ້ງຂວາຂອງ node ໃໝ່ ໃຫ້ເປັນ NULL blockquote>
2. ເມື່ອໃສ່ node ໃໝ່ ຢູ່ໃຈກາງຂອງລາຍຊື່ link ບັນຊີລາຍຊື່ການເຊື່ອມໂຍງກ່ອນການແຊກແລະການຕິດຕາມການແຊກ
ຕົວຊີ້ວັດການປ່ຽນແປງແມ່ນມີຄວາມ ສໍາ ຄັນຫຼາຍ. ລໍາດັບທີ່ບໍ່ຖືກຕ້ອງອາດຈະເຮັດໃຫ້ເກີດ ນີ້ເຮັດໃຫ້ຂໍ້ທີ່ມີຄຸນຄ່າຕົ້ນສະບັບຖືກສູນຫາຍ.
3. ໃນເວລາທີ່ node ໃຫມ່ຖືກໃສ່ເຂົ້າໄປໃນເບື້ອງຊ້າຍຂອງຂໍ້ມູນທີ່ຢູ່ເບື້ອງຊ້າຍຂອງບັນຊີ, ມັນຈະເຮັດໃຫ້ເກີດ pointer L. ມີການປ່ຽນແປງ

23.ການລົບໂນດໃນ (Double Linked List)

ເພື່ອລົບປະເພດ NODE ນີ້ ມັນແຕກຕ່າງຈາກການລົບ NODE ໃນບັນຊີລາຍຊື່ LINK ດຽວໃນນັ້ນມັນບໍ່ ຈຳ ເປັນ. ຕ້ອງມີການຄົ້ນຫາຂໍ້ຂອງຂໍ້. ທີ່ມາກ່ອນຂໍ້ທີ່ຈະຖືກລຶບອອກ, ພຽງແຕ່ ກຳ ນົດທີ່ຕັ້ງຂອງຂໍ້ທີ່ຈະຕ້ອງຖືກລຶບອອກ, ມັນກໍ່ສາມາດຮູ້ໄດ້ ຕຳ ແໜ່ງ ຂອງໂຫດກ່ອນແລະ; ຂໍ້ທີ່ ອອກມາຫຼັງຈາກຂໍ້ນັ້ນ ຖ້າບັນຊີລາຍຊື່ການເຊື່ອມຕໍ່ມີ NODE ດຽວ ການລຶບຂໍ້ອອກຈາກລາຍຊື່ລົ້ງ ຈະຢູ່ ເພື່ອໃຫ້ໄດ້ຮັບລາຍຊື່ລົ້ງກະເປົ໋າແມ່ນຕົວຊີ້ເບື້ອງຊ້າຍ.

ຂໍຂອບໃຈ
ທີ່ຮັບຊົມແລະຕິດຕາມ
ຄຳຖາມວ່າແມວເລີ ?

ເອກະສານອ້າງອີງ: Website ດ້ານລຸ່ມ:

https://iamgique.medium.com/

http://comp-algo.blogspot.com/2011/01/infix-postfix.html

http://asproject.sci.ku.ac.th/Queue/Example1/about.html