ເອກະສານ

ວິຊາ ໂຄງສ້າງຖານຂໍ້ມູນ ແລະ ຂັ້ນຕອນວິທີ **2**

ສອນໂດຍ: ອ.ຈ ຈິດນາວັນ

ກຸ່ມ I

******ຜູ້ສົມຮູ້ຮ່ວມຄິດໃນຄັ້ງນີ້**

ນ.ແຕັກ

ທ.ວົງວິຈິດ

ທ.ຮືວ່າງ

ນ.ໜົມ

ທ.ອາລຸນໄຊ

ທ.ເຊັງວ່າງ

ທ.ເອກພະໄຊ

ທ.ຫຼ້າເບກແກ້ວ

**ຄຳນຳ**

ໃນບົດນີ້ເປັນພຽງແຕ່ວຽກບ້ານໜຶ່ງໃນ**ວິຊາ ໂຄງສ້າງຖານຂໍ້ມູນ ແລະ ຂັ້ນຕອນວິທີ ສຳລັບການພັດທະນາເວັບໄຊ້ 2** ເທົ່ານັ້ນ, ເຊິ່ງແມ່ນ **ອ.ຈ ຈິດນາວັນ** ເປັນຜູ້ມອບໝາຍໃຫ້ ແລະ ຂ້າພະເຈົ້າ ທ.ເອກພະໄຊ ເພົ້າລັດສະໝີ ຕາງໜ້າກຸ່ມທີ I ຈາກຫ້ອງ **2CW1** ໄດ້ພາກັນຊອກຫາຂໍ້ມູນເພື່ອຈະນຳໃຊ້ເຂົ້າໃນເອກະສານສະບັບນີ້ ບໍ່ວ່າຈະເປັນຈາກແຫຼ່ງຕ່າງໆ, ເນື້ອໃນທັງໝົດໃນເອກະສານສະບັບນີ້ເປັນພຽງແຕ່ຄວາມເຂົ້າໃຈ ແລະ ຄວາມຮູ້ຂອງພວກເຮົາເທົ່ານັ້ນ ເຊິ່ງພວກເຮົາ ກຸ່ມທີ I ບໍ່ໄດ້ມີເຈດຕະນາທີ່ຈະປ່ຽນແປງ ຫຼື ບິດເບືອນ ຄວາມຖືກຕ້ອງຂອງບົດຮຽນ ດັ່ງນັ້ນ, ພວກເຮົາຫວັງວ່າ ອາຈານ ແລະ ໝູ່ເພື່ອນທຸກຄົນທີ່ໄດ້ອ່ານເອກະສານສະບັບນີ້ຈະພິຈາລະນາ, ກວດສອບ ແລະ ແນະນຳໃນສິ່ງທີ່ພວກເຮົາຜິດພາດ ຫຼື ຂາດຫາຍໄປໃນເອກະສານສະບັບນີ້ດ້ວຍ.

ຮຽນມາດ້ວຍຄວາມເຄົາລົບທ່ານຜູ້ອ່ານທຸກຄົນ

ກຸ່ມທີ I

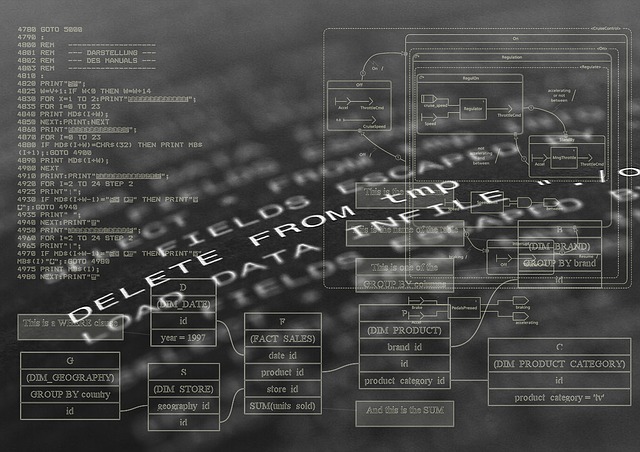
# ວິຊາ: ໂຄງສ້າງຖານຂໍ້ມູນ ແລະ ຂັ້ນຕອນວິທີ

## **ບົດທີ I**

## **ໂຄງສ້າງເບື້ອງຕົ້ນກ່ຽວກັບໂຄງສ້າງຂໍ້ມູນ ແລະ ຂັ້ນຕອນວິທີ**

1. ຄວາມຮູ້ພື້ນຖານກ່ຽວກັບໂຄງສ້າງຂໍ້ມູນ.

ໂຄງສ້າງຂໍ້ມູນ ແມ່ນ ຄວາມສຳພັນລະຫວ່າງຂໍ້ມູນທີ່ຢູ່ໃນໂຄງສ້າງນັ້ນໆ ລວມທັງຂະບວນການໃນການຈັດການຂໍ້ມູນ ເຊັ່ນ: ເພີ່ມ, ແກ້ໄຂ, ລົບ,...



ຕົວຢ່າງໂຄງຂໍ້ມູນປະເພດຕ່າງໆໄດ້ແກ່:

1. Array: ແມ່ນປະເພດຂອງຂໍ້ມູນທີ່ສາມາດເກັບຂໍ້ມູນປະເພດດຽວກັນແບບເປັນລຳດັບໄດ້ ໂດຍຂໍ້ມູນນັ້ນຈະຢູ່ໃນໂຕແປ ທີ່ເອີ້ນວ່າ Array ຂໍ້ມູນແຕ່ລະໂຕຂອງ ອາເລ່ນັ້ນຈະເອີ້ນວ່າ Element ແລະ ຂໍ້ມູນແຕ່ລະ Element ຈະມີໝາຍເລກເພື່ອໃຊ້ໃນການອ້າງອີງ ຈຶ່ງເອີ້ນຕົວເລກນີ້ວ່າ Index.

ການປະກາດ Array

type[] name;

type[] name = new type[size];

type[] name = new type[] {value1, value2, ...};

1. String: ແມ່ນປະເພດຂໍ້ມູນປະເພດຂໍ້ຄວາມ ຫຼື ການເອົາຕົວອັກສອນຫຼາຍໆອັກສອນມາຕໍ່ກັນ ຫຼື ເອີ້ນວ່າ Array ຂອງຕົວອັກສອນ ໂດຍຄວາມຍາວຂອງມັນສາມາດປ່ຽນແປງໄດ້ຕາມຄ່າທີ່ກຳໜົດໃຫ້ກັບໂຕແປ.

ການປະກາດໂຕແປ String

$str1 = "This is string declaration with double quote.";

$str2 = 'This is string declaration with single quote.';

1. Stack: ແມ່ນຢູ່ໃນພື້ນຖານຂອງຫຼັກການແບບ **Last In First Out**(LIFO) ໝາຍຄວາມວ່າສິ່ງທີ່ເພີ່ມເຂົ້າມາໃໝ່ສຸດຈະຖືກລົບອອກກ່ອນ.

ຄຳສັ່ງພື້ນຖານຂອງ Stack

* push()- ເປັນການເພີ່ມ Element ລົງໄປໃນສ່ວນເທິງສຸດຂອງ Stack
* pop()- ເປັນການລົບ Element ທີ່ຢູ່ສ່ວນເທິງສຸດຂອງ Stack
* peek()- ເປັນການເບິ່ງ Element ທີ່ຢູ່ສ່ວນເທິງສຸດຂອງ Stack
* isFull()- ເປັນການກວດສອບວ່າ Stack ເຕັມ ຫຼື ບໍ່
* isEmpty()- ເປັນການກວດສອບວ່າ Stack ຫວ່າງ ຫຼື ບໍ່

1. Queue: ເປັນ Linear Data Structure ເຊິ່ງຈະຄ້າຍຄືກັນກັບ Stack, Queue ແມ່ນຢູ່ໃນພື້ນຖານຂອງຫຼັກການແບບ **First In First Out**(FIFO) ໝາຍຄວາມວ່າສິ່ງໃດທີ່ເຂົ້າກ່ອນຈະໄດ້ອອກກ່ອນ ແລະ ສິ່ງໃດທີ່ເຂົ້າມານຳຫຼັງກໍຈະໄດ້ອອກນຳຫຼັງ.

ຄຳສັ່ງພື້ນຖານຂອງ Queue

* Enqueue()- ເປັນການເພີ່ມຂໍ້ມູນລົງໄປໃນ Queue
* Dequeue()- ເປັນການລົບຂໍ້ມູນລົງໄປໃນ Queue
* Peek()- ເປັນການເບິ່ງຂໍ້ມູນທີ່ຢູ່ຕຳແໜ່ງ Front ຂອງ Queue
* isFull()- ເປັນການກວດສອບວ່າ Queue ເຕັມ ຫຼື ບໍ່
* isEmpty()- ເປັນການກວດສອບວ່າ Queue ຫວ່າງ ຫຼື ບໍ່

1. ປະເພດຂອງໂຄງສ້າງຂໍ້ມູນ.

ປະເພດຂອງໂຄງສ້າງຂໍ້ມູນທີ່ນຳໃຊ້ໃນປັດຈຸບັນແບ່ງອອກເປັນ 2 ປະເພດຄື:

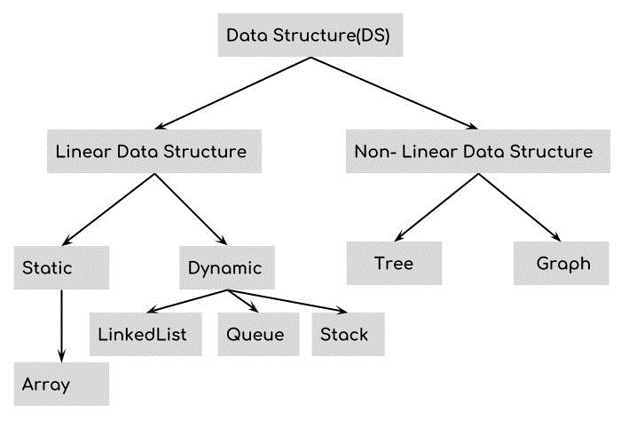
1. ໂຄງສ້າງຂໍ້ມູນທາງກາຍະພາບ.

ເປັນໂຄງສ້າງຂໍ້ມູນທີ່ນຳໃຊ້ທົ່ວໄປໃນພາສາຄອມພິວເຕີ ເຊິ່ງແບ່ງອອກເປັນ 2 ປະເພດຄື:

* 1. ຂໍ້ມູນເບື້ອງຕົ້ນເຊັ່ນ: Int, Float, Character
* Int: ແມ່ນຊະນິດຂໍ້ມູນທີ່ເປັນຕົວເລກຖ້ວນ
* Float: ແມ່ນຊະນິດຂໍ້ມູນທີ່ເປັນຕົວເລກຈຳນວນຈິງ ຫຼື ຕົວເລກທີ່ມີຈຸດ
* Char: ແມ່ນຊະນິດຂໍ້ມູນທີ່ເປັນຕົວອັກສອນ
  1. ຂໍ້ມູນໂຄງສ້າງເຊັ່ນ: Field, Record, File
* Field: ໝາຍເຖິງການນຳເອົາຂໍ້ມູນທີ່ສຳຄັນມາໄວ້ໃນອັນດຽວ
* Record: ໝາຍເຖິງການນຳເອົາຫຼາຍໆ Field ມາຮວມເຂົ້າກັນ
* File: ໝາຍເຖິງການນຳເອົາ Record ຫຼາຍໆ Record ທີ່ກ່ຽວຂ້ອງກັນໃນດ້ານໃດດ້ານໜຶ່ງມາລວມເຂົ້າກັນ

1. ໂຄງສ້າງຂໍ້ມູນທາງຕັກກະ.

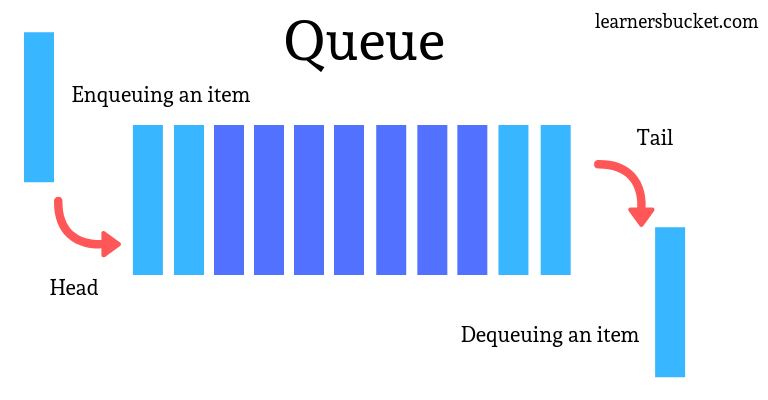
ເປັນໂຄງສ້າງທີ່ເກີດຈາກຈິນຕະນາການຂອງຜູ້ໃຊ້ ເພື່ອນຳໃຊ້ໃນການແກ້ໄຂບັນຫາໃນໂປຣແກຣມທີ່ສ້າງຂຶ້ນ ແບ່ງອອກເປັນ 2 ປະເພດຄື:



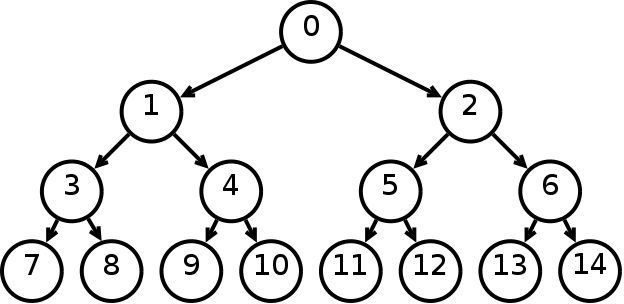
* 1. Linear Data Structure

ເປັນໂຄງສ້າງຂໍ້ມູນທີ່ຄວາມສຳພັນຂອງຂໍ້ມູນຈະລຽງຕໍ່ກັນ ເຊັ່ນ: List, Stack, Queue, String ເປັນຕົ້ນ.

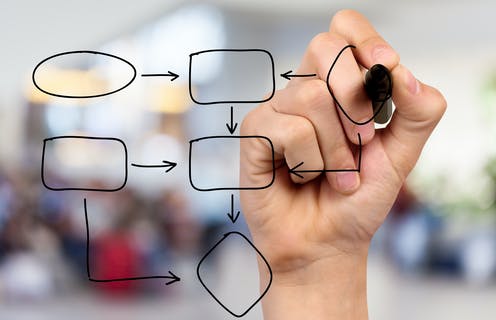


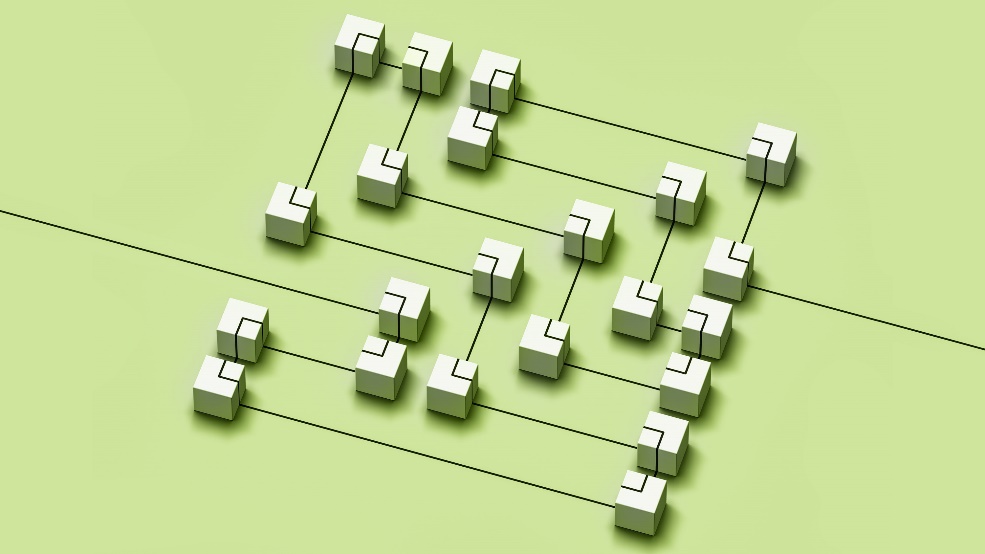


* 1. Non-Linear Data Structure

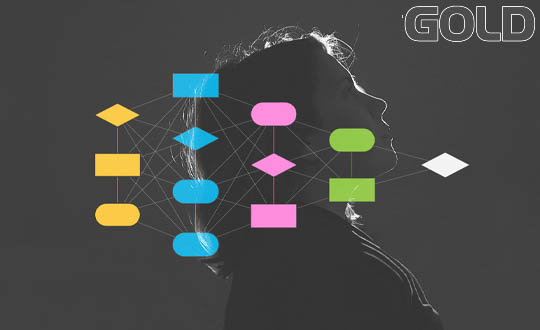
ເປັນໂຄງສ້າງຂໍ້ມູນທີ່ຂໍ້ມູນແຕ່ລະຕົວສາມາດມີຄວາມສຳພັນກັບຂໍ້ມູນອື່ນໄດ້ຫຼາຍຕົວ ເຊັ່ນ: Tree, Graph

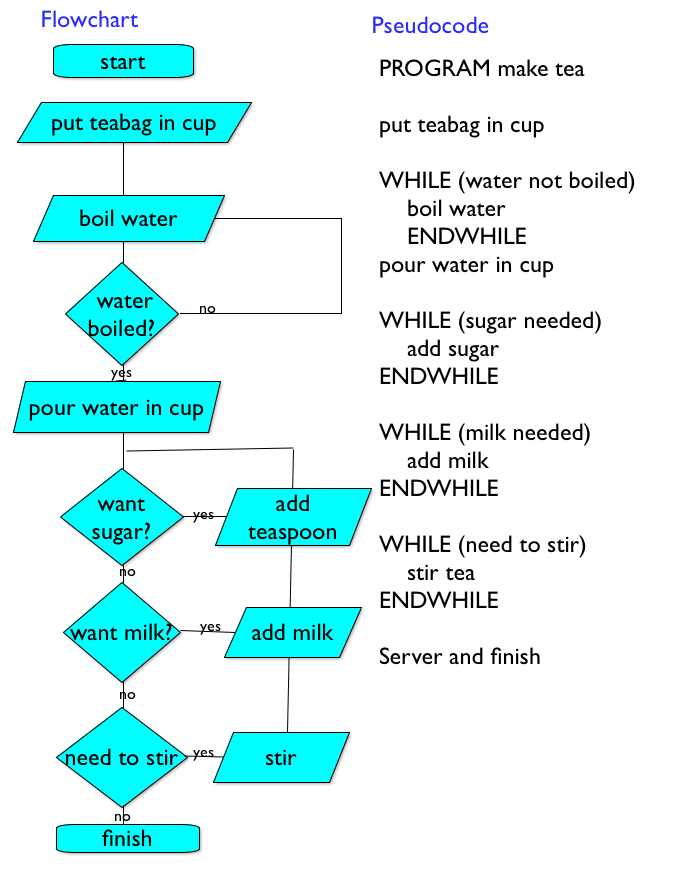
1. ການເລືອກໃຊ້ໂຄງສ້າງຂໍ້ມູນ.
2. ໂຄງສ້າງຂໍ້ມູນນັ້ນຈະຕ້ອງສ້າງຄວາມສຳພັນໃຫ້ກັບຂໍ້ມູນຊຸດນັ້ນໆໄດ້ຢ່າງສົມບູນທີ່ສຸດ
3. ໂຄງສ້າງຂໍ້ມູນນັ້ນຕ້ອງງ່າຍຕໍ່ການດຳເນີນການໃນຂະບວນການ





1. ຂັ້ນຕອນວິທີ (Algorithm).

ແມ່ນວິທີການແກ້ບັນຫາຕ່າງໆຢ່າງມີລະບົບ ມີລຳດັບຂັ້ນຕອນ ຕັ້ງແຕ່ຕົ້ນຈົນຈົບ, ສາມາດຂຽນໄດ້ຫຼາຍແບບ, ການເລືອກໃຊ້ຕ້ອງເລືອກໃຊ້ຂັ້ນຕອນທີ່ເໝາະສົມ, ກະທັດຮັດ ແລະ ມີປະສິດທິພາບທີ່ສຸດ.

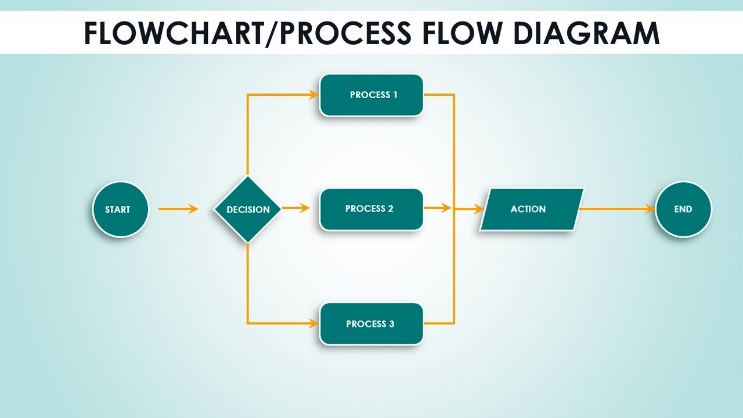
* ຂັ້ນຕອນວິທີທີ່ດີ ຕ້ອງມີຄຸນສົມບັດດັ່ງລຸ່ມນີ້:
* ມີຄວາມຖືກຕ້ອງ
* ໃຊ້ເວລາໃນການປະຕິບັດງານໜ້ອຍທີ່ສຸດ
* ສັ້ນກະທັດຮັດ ມີສະເພາະຂັ້ນຕອນທີ່ຈຳເປັນ
* ນຳໃຊ້ໜ່ວຍຄວາມຈຳໜ້ອຍທີ່ສຸດ
* ມີຄວາມຢືດຢຸ່ນໃນການໃຊ້ງານ
* ໃຊ້ເວລາໃນການພັດທະນາໜ້ອຍທີ່ສຸດ
* ງ່າຍຕໍ່ການທຳຄວາມເຂົ້າໃຈ

1. ການສະແດງຂັ້ນຕອນວິທີ

ການສະແດງຂັ້ນຕອນວິທີ ແມ່ນການສະແດງຂັ້ນຕອນທີ່

ໄດ້ນຳໃຊ້ ບໍ່ວ່າຈະເປັນ ການຂຽນຜັງງານ (Flowchart),

ລະຫັດຈຳລອງ ຫຼື ລະຫັດທຽມ ແລະ ພາສາທຳມະຊາດ.



1. ພາສາຂັ້ນຕອນວິທີ

ເປັນພາສາ ສຳລັບຂຽນຂັ້ນຕອນວິທີ ມີຮູບແບບທີ່ສັ້ນ ກະທັດຮັດ ແລະ ມີຂໍ້ກຳນົດດັ່ງຕໍ່ໄປນີ້:

* ຕົວປ່ຽນຈະຕ້ອງຂຽນແທນດ້ວຍຕົວອັກສອນ ຫຼື ຕົວອັກສອນປະສົມກັບຕົວເລກ
* ການກຳນົດຄ່າໃຫ້ກັບຕົວປ່ຽນຈະໃຊ້ເຄື່ອງໝາຍເທົ່າກັບ =
* ນິພົນທີ່ເປັນການຄຳນວນຈະມີລຳດັບຂັ້ນຂອງການຄຳນວນຕາມລຳດັບຄື: ວົງເລັບ, ຂຶ້ນກຳລັງ, ຄູນ, ຫານ, ບວກ ແລະ ລົບ ລໍາດັບຄວາມສໍາຄັນແມ່ນແຕ່ຊ້າຍຫາຂວາ
* ນິພົດທີ່ເປັນຕັກກະສາດ

ນິພົດທີ່ເປັນຕັກກະສາດ ຈະໃຊ້ເຄື່ອງໝາຍໃນການປຽບທຽບ ຄື:

== ເທົ່າກັບ

!= ບໍ່ເທົ່າກັບ

> ໃຫຍ່ກວ່າ

< ນ້ອຍກວ່າ

>= ໃຫຍ່ກວ່າ ຫຼື ເທົ່າກັບ

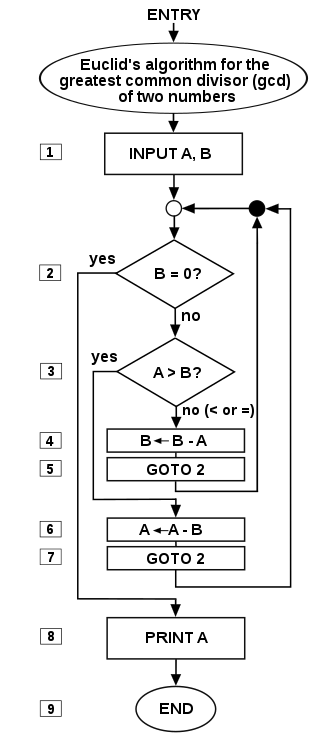
<= ນ້ອຍກວ່າ ຫຼື ເທົ່າກັບ

## **ບົດທີ 2:**

## **ການວິເຄາະປະສິດທິພາບຂອງຂັ້ນຕອນວິທີ**

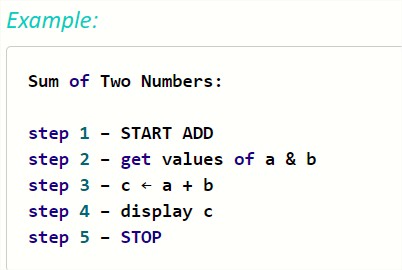
1. ການວິເຄາະປະສິບທີພາບຂອງຂັ້ນຕອນວິທີ

ປະສິດທິພາບຂອງອາກໍຮິທືມ ໂດຍທົ່ວໄປມີມາດຕະຖານການວັດຜົນສອງແບບໂດຍແບບທຳອິດແມ່ນ ການໃຊ້ໃນພື້ນທີ່ວ່າງ (space Utilization) ເປັນຈຳນວນຂອງຫນ່ວຍຄວາມຈຳທີ່ຕ້ອງໃຊ້ເພື່ອໃຫ້ງານສຳເລັດ ,ປະກອບດ້ວຍພື້ນທີ່ເກັບຄຳສັງທີ່ເກັບຂ້ໍມູນ ແລະພື້ນທີ່ສະພາວະສິ່ງແວດລ້ອມແຕກຕ່າງຄືແບບທີ່ 2 ປະສິດທິພາບຂອງເວລາພາວະເວລາ (time Efficiency) ເປັນຈຳນວນເວລາທີ່ໃຊ້ໃນການປະມວນຜົນ



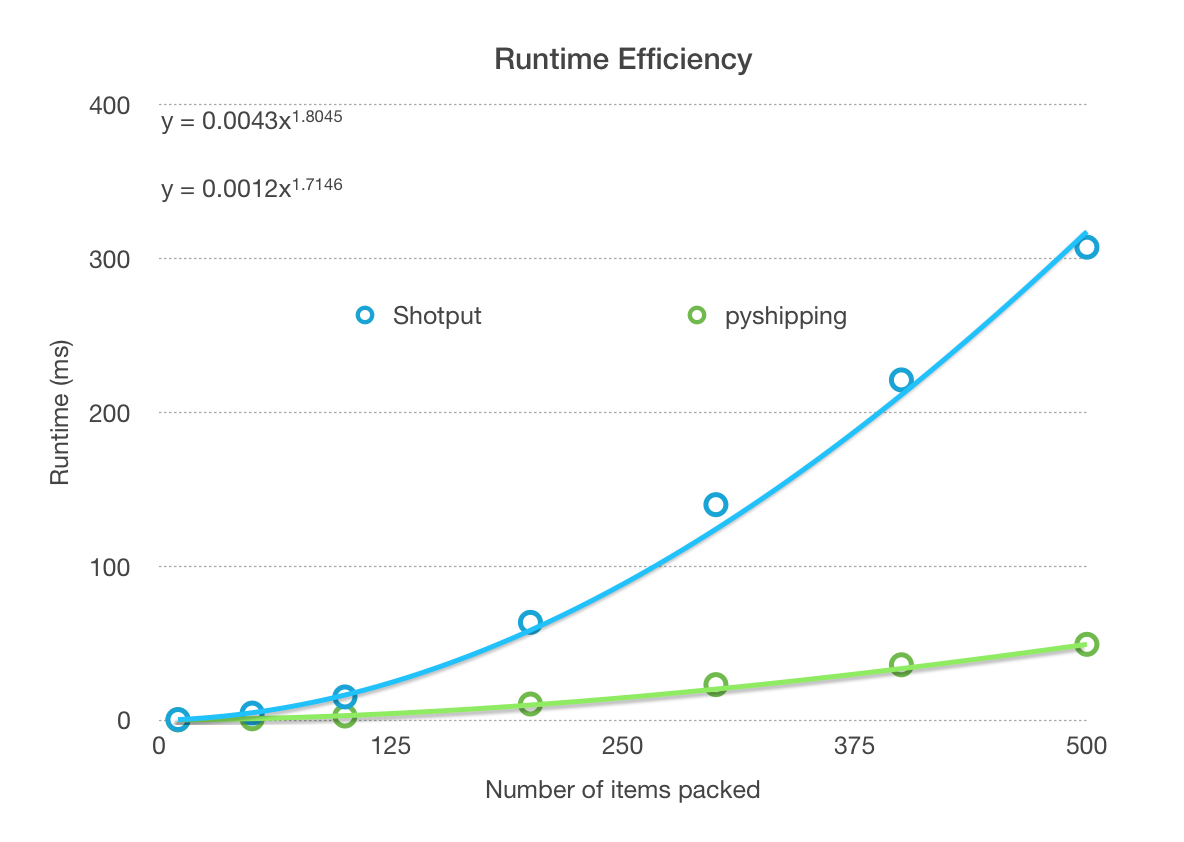
**ຄຸນລັກສະນາຂອງຂັ້ນຕອນວິທີທີ່ດີມີຄື:**

1. ເຮັດວຽກໄດ້ຢ່າງຖືກຕ້ອງ
2. ເຮັດວຽກໄດ້ໄວ
3. ໃຊ້ຊັບພະຍາກອນຢ່າງຄຸ້ມຄ່າ
4. ບໍ່ສັບຊ້ອນ (ເຂົ້າໃຈງ່າຍ)
5. ສາມາດນຳໄປໃຊ້ໄດ້ກັບວຽກທີ່ຫຼາກຫຼາຍ

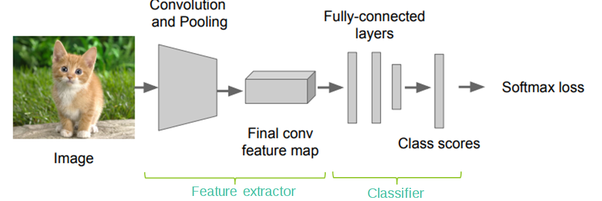


1. ປະສິດທິພາບຂອງຂັ້ນຕອນວິທີ

ປະສິດທິພາບຂອງ ຂັ້ນຕອນວິທີ ຫຼື ອານກໍຣິດທຶມ (Algorithm) ແມ່ນ ການປະເມີນຄ່າຊັບພະຍາກອນທີ່ຈຳເປັນຕ້ອງໃຊ້ໃນການທຳງານ ເຊັ່ນ: ເວລາ ຫຼື ໜ່ວຍຄວາມຈໍາ, ຂັ້ນຕອນວິທີສ່ວນຫຼາຍອອກແບບມາເພື່ອໃຫ້ສາມາດຮອງຮັບຈຳນວນຂໍ້ມູນນຳເຂົ້າ (Input) ໄດ້ບໍ່ຈຳກັດ ປົກກະຕິແລ້ວປະສິດທິພາບ ຫຼື ຄວາມສັບຊ້ອນຂອງຂັ້ນຕອນວິທີ ຈະວັດຈາກຄວາມສຳພັນຂອງຈຳນວນຂໍ້ມູນນຳເຂົ້າ ກັບ ເວລາທີ່ໃຊ້ໃນການທຳງານ ຫຼື ໜ່ວຍຄວາມຈຳທີ່ນຳໃຊ້

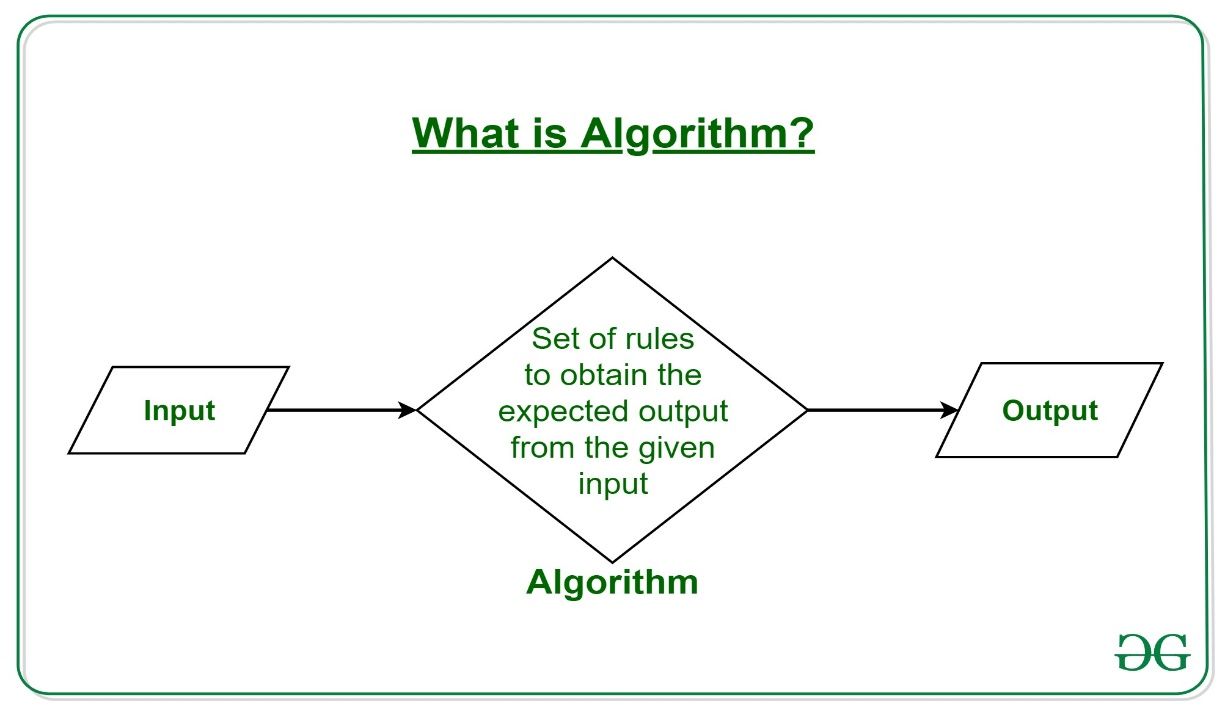


1. ວິທີວິເຄາະງານໃຊ້ຫນ່ວຍຄວາມຈຳທັງຫມົດ

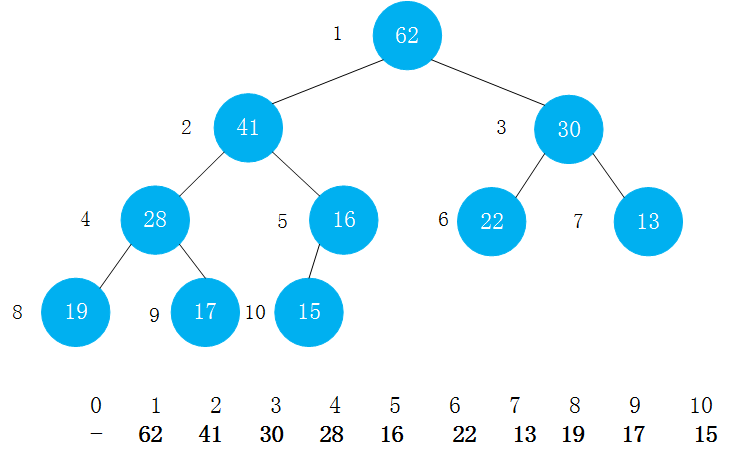


1. ການວິເຄາະໜ່ວຍຄວາມຈຳທັງໝົດທີ່ໂປຣແກຣມໃຊ້ໃນການປະມວນຜົນຂອງຂັ້ນຕອນວິທີ
2. ເພື່ອໃຫ້ຮູ້ເຖິງຂະໜາດຂໍ້ມູນທີ່ສາມາດປ້ອນ ຫຼື ຂໍ້ມູນເຂົ້າມາໃຫ້ຂັ້ນຕອນວິທີປະມວນຜົນແລ້ວບໍ່ເກີດຂໍ້ຜິດພາດ
3. ອົງປະກອບຂອງການວິເຄາະການໃຊ້ໜ່ວຍຄວາມຈຳທີ່ໃຊ້ໃນການປະມວນຜົນ

* Instruction Space: ແມ່ນ ຂະໜາດໜ່ວຍຄວາມຈຳທີ່ຈຳເປັນຕ້ອງໃຊ້ໃນເວລາຄອມໄຟລເລີ້ ໂປຣແກຣມ

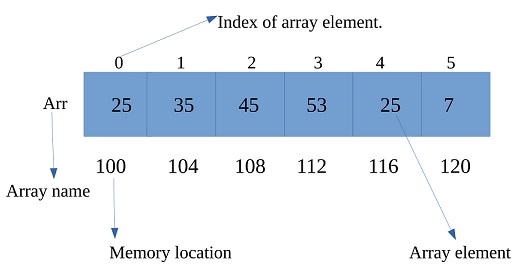


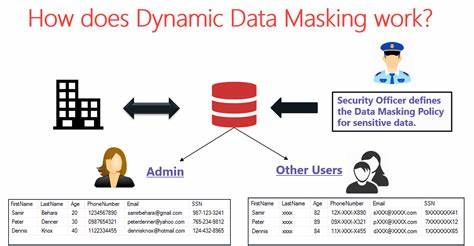
* Data Space: ແມ່ນ ຂະໜາດໜ່ວຍຄວາມຈຳທີ່ຈຳເປັນຕ້ອງໃຊ້ສຳລັບເກັບຂໍ້ມູນຄ່າຄົງທີ່ ແລະ ຕົວປ່ຽນທີ່ໃຊ້ໃນເວລາປະມວນຜົນໂປຣແກຣມ



ເຊິ່ງແບ່ງອອກເປັນ 2 ປະເພດ ຄື:

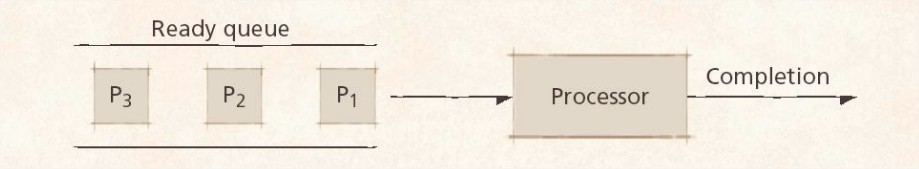
* Static memory: ຂະໜາດໜ່ວຍຄວາມຈໍາທີ່ຕ້ອງໃຊ້ໃນການປະມວນຜົນຢ່າງແນ່ນອນ ເຊັ່ນ: Array

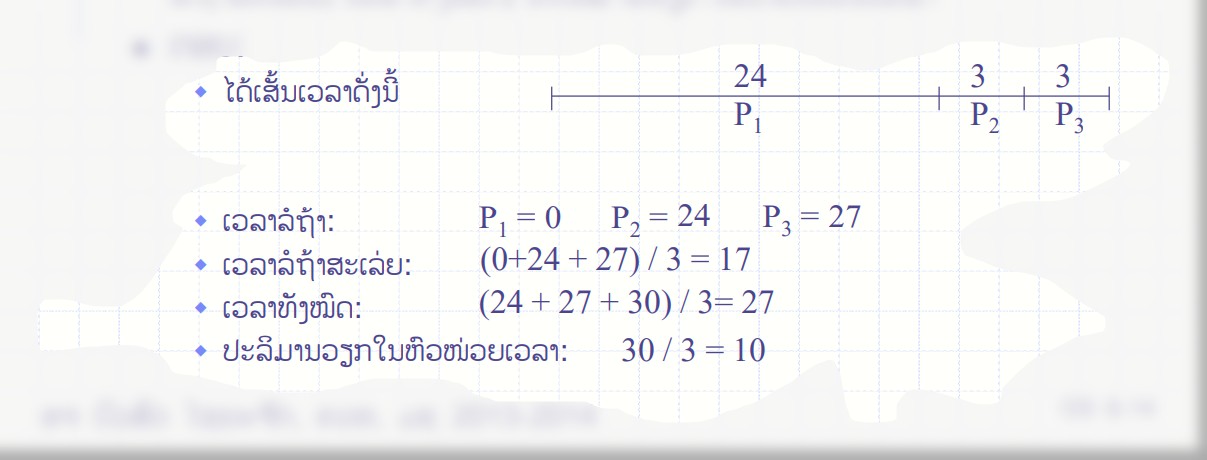


* Dynamic memory: ຂະໜາດໜ່ວຍຄວາມຈໍາທີ່ຕ້ອງໃຊ້ໃນການປະມວນຜົນທີ່ບໍ່ແນ່ນອນ ໝາຍຄວາມວ່າ ຈະໃຊ້ໜ່ວຍຄວາມຈໍານີ້ກໍ່ຕໍ່ເມື່ອໂປຣແກຣມຕ້ອງການໃຊ້ງານເທົ່ານັ້ນ ໂດຍບໍ່ມີການຈອງເນື້ອທີ່ໄວ້ລ່ວງໜ້າ
* Environment Stack Space: ແມ່ນໜ່ວຍຄວາມຈຳທີ່ໃຊ້ສຳລັບເກັບຂໍ້ມູນຜົນຮັບທີ່ໄດ້ຈາກການປະມວນຜົນ ເພື່ອລໍເວລາທີ່ຈະນຳກັບໄປໃຊ້ງານໃໝ່ໃນໂປຣແກຣມ ເຊິ່ງໜ່ວຍຄວາມຈຳປະເພດນີ້ຈະເກີດຂຶ້ນເມື່ອມີການໃຊ້ງານເທົ່ານັ້ນ



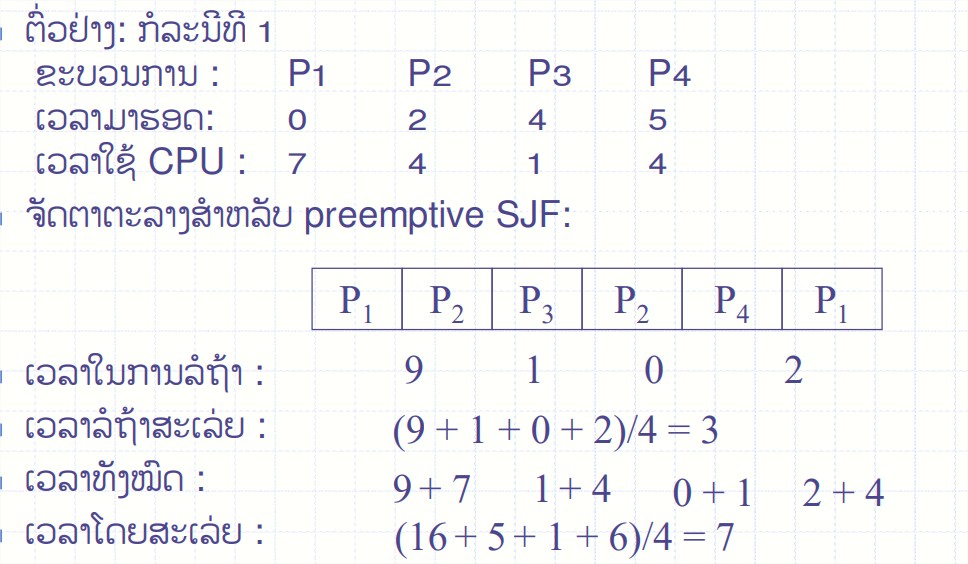
1. ການວິເຄາະເວລາທີ່ໃຊ້ໃນການປະມວນຜົນ

****

* ເພື່ອວິເຄາະຂັ້ນຕອນທີ່ຕ້ອງໃຊ້ໃນການດຳເນີນການປະມວນຜົນຂັ້ນຕອນວິທີ
* ເພື່ອໃຊ້ປະເມີນເວລາທີ່ໃຊ້ໃນການປະມວນຜົນ.
* ຮູ້ເຖິງປະສິດທິພາບການທຳງານຂອງໂປຣແກຣມ ແລະ ແກ້ໄຂປັນຫາໄດ້ຢ່າງຖືກຕ້ອງ
* ສາມາດເລືອກໃຊ້ຄອມພິວເຕີທີ່ເໝາະສົມກັບການປະມວນຜົນຂັ້ນຕອນວິທີ

ຫຼັກການພິຈາລະນາເວລາທີ່ໃຊ້ໃນການປະມວນຜົນ

* ເມືອປະມວນຜົນໂປຣແກຣມໃນເຄື່ອງຄອມພິວເຕີທີ່ມີປະສິດທິພາບໄວກວ່າ ກໍ່ຈະປະມວນຜົນຂໍ້ມູນໄວກວ່າ
* ເມື່ອຣັນໂປຣແກຣມທີ່ມີຜົນການເຮັດວຽກດຽວກັນ ໂປຣແກຣມທີ່ມີໂຄດນ້ອຍກວ່າຈະເຮັດວຽກໄດ້ໄວກວ່າ
* ຕົວປ່ຽນທີ່ມີຂະໜາດໜ່ວຍຄວາມຈຳນ້ອຍກວ່າຈະປະມວນຜົນໄດ້ໄວກວ່າ



**ປະເພດຂອງເວລາທີ່ໃນການປະມວນຜົນ**

* Compile timeເປັນເວລາທີ່ໃຊ້ໃນການກວດໄວຍະກອນການຂຽນ (Syntax)
* Run timeຫຼື Execution timeເປັນເວລາທີ່ຄອມພິວເຕີໃຊ້ໃນການປະມວນຜົນຂັ້ນຕອນວິທີ ເຊິ່ງຂຶ້ນຢູ່ກັບຊະນິດຂໍ້ມູນ, ຈໍານວນຕົວປ່ຽນທີ່ໃຊ້ໃນໂປຣແກຣມ ແລະ ຈໍານວນວົນຮອບ

ຕົວຢ່າງ: ວິເຄາະເວລາທີ່ໃຊ້ໃນການປະມວນຜົນຂອງໂປຣແກຣມລຸ່ມນີ້:

* int n = 20; 🡨 ກຳນົດຄ່າ 1 ຄັ້ງ
* int total = 0; 🡨 ກຳນົດຄ່າ 1 ຄັ້ງ
* while(n!=20) { 🡨 ປຽບທຽບຄ່າ n+1 ຄັ້ງ
* total += n; 🡨 ຄຳນວນ n ຄັ້ງ
* ++n; 🡨 ຄຳນວນ n ຄັ້ງ
* } // ຈົບຄຳສັ່ງ while
* System.out.println(“Total = ” + total); 🡨 ສະແດງຜົນ 1 ຄັ້ງ

ຄຳອະທິບາຍ

ກຳນົດໃຫ້ f(n) ແທນປະສິດທິພາບໃນການວິເຄາະເວລາທີ່ໃຊ້ໃນການປະມວນຜົນ

n ແທນຈຳນວນຮອບໃນການເຮັດວຽກ

ຈະໄດ້ວ່າຂັ້ນຕອນວິທີນີ້ມີປະສິດທິພາບ

f(n) = 1+1+(n+1)+n+n+1 = 3n+4

ຕົວຢ່າງ: ວິເຄາະເວລາທີ່ໃຊ້ໃນການປະມວນຜົນຂອງໂປຣແກຣມວົນຊໍ້າລຸ່ມນີ້:

factorial(in n:int):int 🡨 ຖືກເອີ້ນໃຊ້ n ຄັ້ງ

if(n == 0) 🡨 ກວດສອບເງື່ອນໄຂ n ຄັ້ງ

return 1 🡨 ຄືນຄ່າ 1 ຄັ້ງ

else return n\*(factorial(n-1)) 🡨 ເອີ້ນໃຊ້ຕົວເອງ n ຄັ້ງ

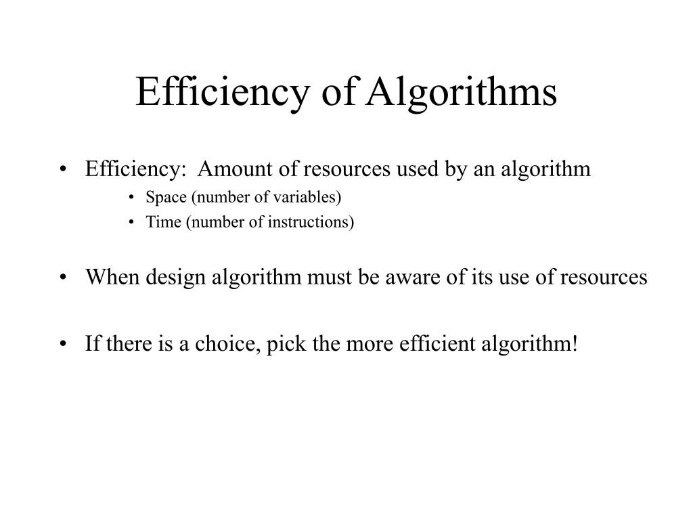
* ພິຈາລະນາຕາມຈຳນວນຮອບທີ່ວົນຊ້ຳ ຖ້າກຳນົດ n = 3 ຈະຕ້ອງທຳການວົນຊ້ຳ 3 ຄັ້ງ
* ໃຊ້ເນື້ອທີ່ໃນໜ່ວຍຄວາມຈຳເທົ່າກັບ 3\*4 = 12 byte
* ສະຫຼຸບໄດ້ວ່າ ຖ້າກຳນົດໃຫ້ຫາຄ່າ Factorial n ຈະຕ້ອງໃຊ້ເນື້ອທີ່ໜ່ວຍຄວາມຈຳເທົ່າກັບ n\*4

## **ບົດທີ 3:**

## **ປະສິດທິພາບຂອງ Algorithm**

1. ປະສິດທິພາບຂອງ Algorithm

ປະສິດທິພາບຂອງອະກໍຣິທຣືມແມ່ນອາໄສການພິຈາລະນາຈາກເວລາທີ່ໃຊ້ໃນການປະມວນຜົນ ແລະ ຈາກຈຳນວນຂອງໜ່ວຍຄວາມຈຳທີ່ໃຊ້ໃນການປະມວນຜົນນັ້ນແມ່ນບໍ່ຍູດຕິທຳພຽງພໍການປຽບທຽບຂອງອະກໍຣິທຣຶມວ່າອັນໃດດີກວ່າກັນນັ້ນຕ້ອງມີວິທີເໝາະສົມ ແລະ ຍຶດຕິທຳ.

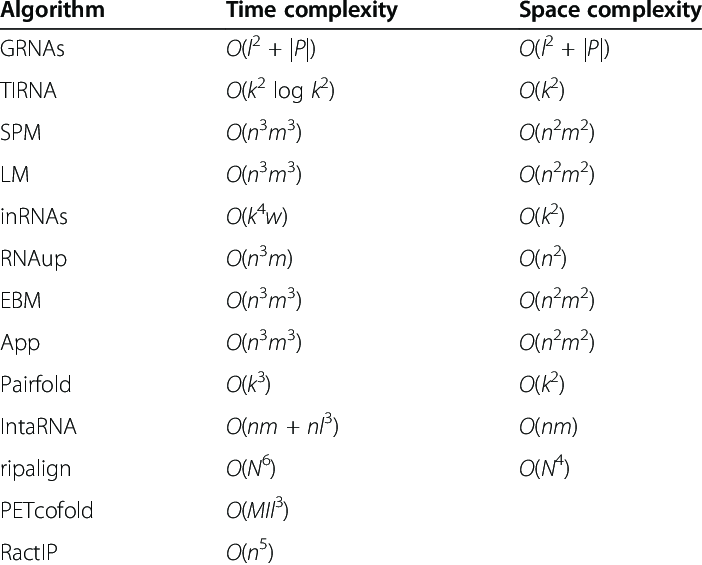


ປະສິດທິພາບຂອງອະກໍຣິທຣືມໄດ້ຖືກພິຈາລະນາຈາກ 2 ປັດໃຈຄື:

1. ການວິເຄາະ (Analysis) ໃຊ້ວິເຄາະວິທີການເຮັດວຽກຂອງອະກໍຣິທຣືມ
2. ການວັດແທກ (Measure) ໃຊ້ເພື່ອວັດແທກຈາກການທົດລອງຕົວຈິງ

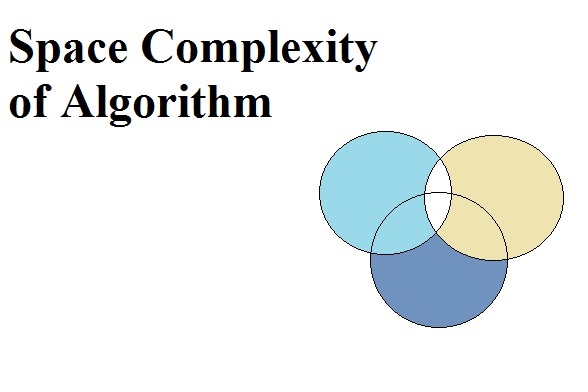
ປະສິດທິພາບຂອງອະກໍຣິທຣືມໂດຍທົ່ວໄປມີມາດຕະຖານການວິເຄາະ 2 ແບບຄື:

1. ການວິເຄາະໂດຍໃຊ້ໜ່ວຍຄວາມຈຳທີ່ຕ້ອງໃຊ້ໃນການປະມວນຜົນ (Space Complexity).
2. ການວິເຄາະເວລາທີ່ໃຊ້ໃນການປະມວນຜົນ (Time Complexity).



1. ການວິເຄາະໂດຍໃຊ້ໜ່ວຍຄວາມຈຳທີ່ຕ້ອງໃຊ້ໃນການປະມວນຜົນ (Space Complexity)

ແມ່ນການວິເຄາະທີ່ເຮັດໃຫ້ຮູ້ຈັກວ່າອະກໍຣິທຣືມນັ້ນສາມາດຮອງຮັບຂໍ້ມູນທີ່ສົ່ງເຂົ້າມາປະມວນຜົນໄດ້ຫຼາຍສຸດເທົ່າໃດ, ສຳລັບອະກໍຣິທຣືມທີ່ຕ້ອງປະມວນຜົນໃນຄອມພິວເຕີທີ່ໃຊ້ຮ່ວມກັນຫຼາຍຄົນຜ່ານລະບົບເຄືອຂ່າຍຈຳເປັນຕ້ອງຮູ້ຈຳນວນຂອງໜ່ວນຄວາມຈຳເພື່ອບໍ່ໃຫ້ກະທົບກັບການເຮັດວຽກຂອງຄົນອື່ນ ແລະ ຍັງສາມາດເລືອກສະເປັກຂອງຄອມພິວເຕີຊຶ່ງຈະນຳໂປຣແກຣມໄປຕິດຕັ້ງໄດ້ຢ່າງເໝາະສົມ.



1. ການວິເຄາະເວລາທີ່ໃຊ້ໃນການປະມວນຜົນ (Time Complexity)



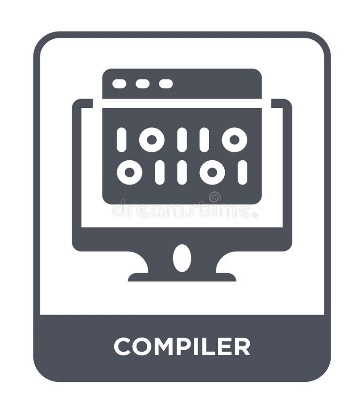
ກ. ຫຼັກການໃນການພິຈາລະນາເວລາທີ່ໃຊ້ໃນການປະມວນຜົນ

ໂປຣແກຣມຈະສາມາດປະມວນຜົນໄດ້ໄວກວ່າເມຶ່ອເຮັດວຽກຢູ່ເທິງເຄື່ອງທີ່ມີຄວາມໄວໃນການປະມວນຜົນສູງກວ່າ, ຖ້າໃຊ້ Compiler ຕົວດຽວກັນ Code ທີ່ສັ້ນກວ່າຈະໃຊ້ເວລາໃນການປະມວນຜົນທີ່ສັ້ນກວ່າ, ມີຕົວປ່ຽນທີ່ມີຈຳນວນ byte ນ້ອຍກວ່າຈະປະມວນຜົນໄດ້ໄວກວ່າ.

ຂ. ເວລາທີ່ໃຊ້ໃນການປະມວນຜົນຂອງໂປຣແກຣມ

- Compile Time ໄລຍະເວລາໃນການກວດສອບໄວຍະກອນຂອງ Code.

- Run Time / Execution Time ເວລາທີ່ເຄື່ອງຄອມພິວເຕີ້ທີ່ໃຊ້ໃນການເຮັດວຽກເຊິ່ງຂຶ້ນກັບຈຳນວນຕົວປ່ຽນ ແລະ ຄຸນລັກສະນະຂອງຕົວປ່ຽນທີ່ໃຊ້ໃນໂປຣແກຣມ.

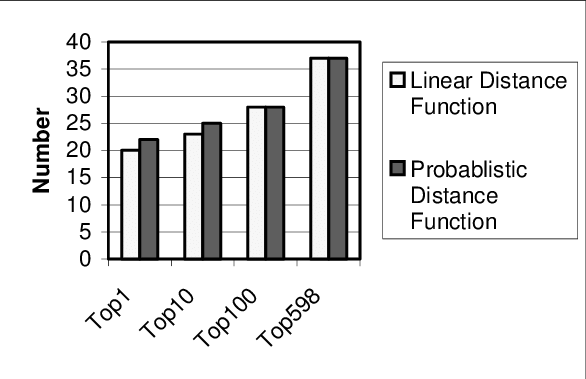


ຂະໜາດຂອງຂໍ້ມູນທີ່ໄດ້ຮັບເຂົ້າມາ

ແມ່ນຈຳນວນຂອງຂໍ້ມູນທີ່ຮັບເຂົ້າມາເຮັດວຽກຈະມີຜົນກະທົບຕໍ່ເວລາໃນການໃຊ້ງານການປະມວນຜົນຂໍ້ມູນຂອງລາຍການຂື້ນກັບຈຳນວນຂອງຂໍ້ມູນທີ່ໃນລາຍການ. ດັ່ງນັ້ນໃນເວລາເຮັດວຽກຂອງອະກໍຣິທຣືມຈະສະແດງໃນຮູບແບບ T(n) ຂອງຂໍ້ມູນທີ່ໄດ້ຮັບຂໍ້ມູນມາມີຂະໜາດ n.

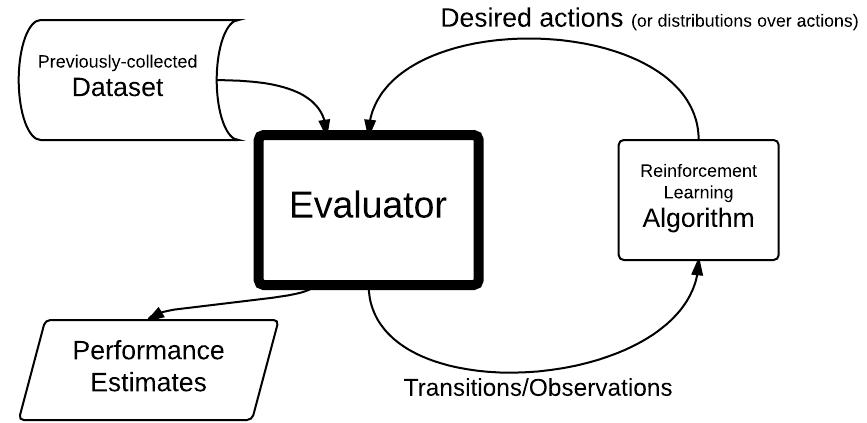
ການປຽບທຽບ

ການປຽບທຽບແມ່ນອາໄສການພິຈາລະນາຈາກເວລາທີ່ໃຊ້ໃນການປະມວນຜົນ ແລະ ຈາກຈຳນວນຂອງໜ່ວຍຄວາມຈຳທີ່ໃຂ້ໃນການປະມວນຜົນນັ້ນແມ່ນບໍ່ຍຸດຕິທຳພຽງພໍ. ການປຽບທຽບອະກໍຣິທຣຶມວ່າອັນໃດດີກວ່າກັນນັ້ນຕ້ອງມີວິທີເໝາະສົມ ແລະ ຍຸດຕິທຳ.



ການປະເມີນ

ວິທີການປະເມີນປະສິດທິພາບຂອງອະກໍຣິທຣຶມໂດຍການວິເຄາະສະເພາະອະກໍຣິທຣຶມພຽງຢ່າງດຽວເປັນວິທີທີ່ໃຊ້ສັນຍາລັກ.



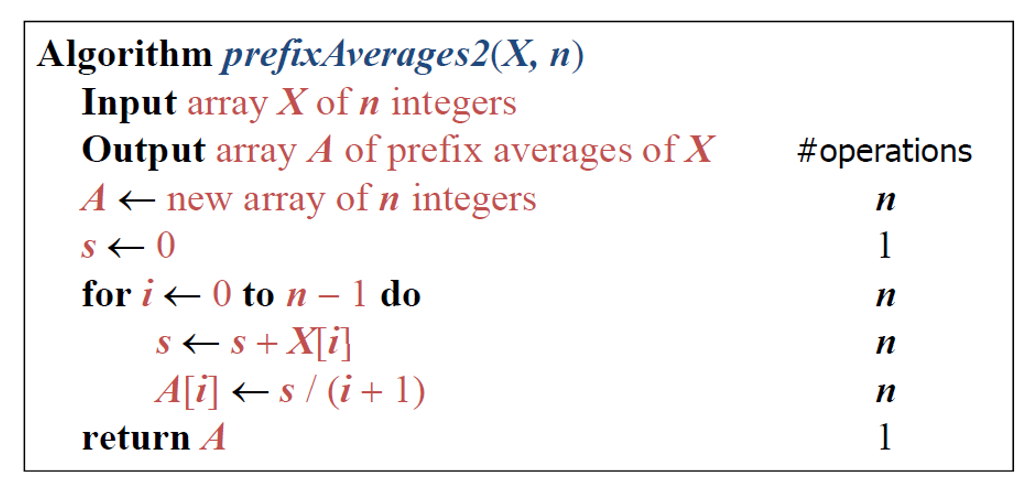
**Step Count**

ແມ່ນຄ່າຕຳລາທີ່ໃຊ້ອະທິບາຍພຶດຕິກຳການປະມວນຜົນຂອງອະກໍຣິທຣຶມເປັນຄ່າຕຳລາທາງຄະນິດສາດທີ່ສະແດງເຖິງຄວາມສຳພັນລະຫວ່າງປະລິມານຂໍ້ມູນທີ່ກຳລັງຖືກປະມວນຜົນກັບເວລາທີ່ຕ້ອງໃຊ້ໃນການປະມວນຜົນຂໍ້ມູນນັ້ນ.

ການຊອກຫາຄ່າເວລາທີ່ໃຊ້ໃນການປະມວນຜົນຂອງອະກໍຣິທຣຶມໃດໜຶ່ງມີ 2 ວິທີຄື:

1. Operation Count

ເປັນວິທີການຄຳນວນເວລາໃນການປະມວນຜົນໂດຍພິຈາລະນາຈາກ Operation ທີ່ຢູ່ພາຍໃນອະກໍຣິທຣຶມນັ້ນຊຶ່ງເປັນການນັບຈຳນວນການປະມວນຜົນ Operation ຕ່າງໆທັງໝົດໃນອະກໍທຣຶມ.

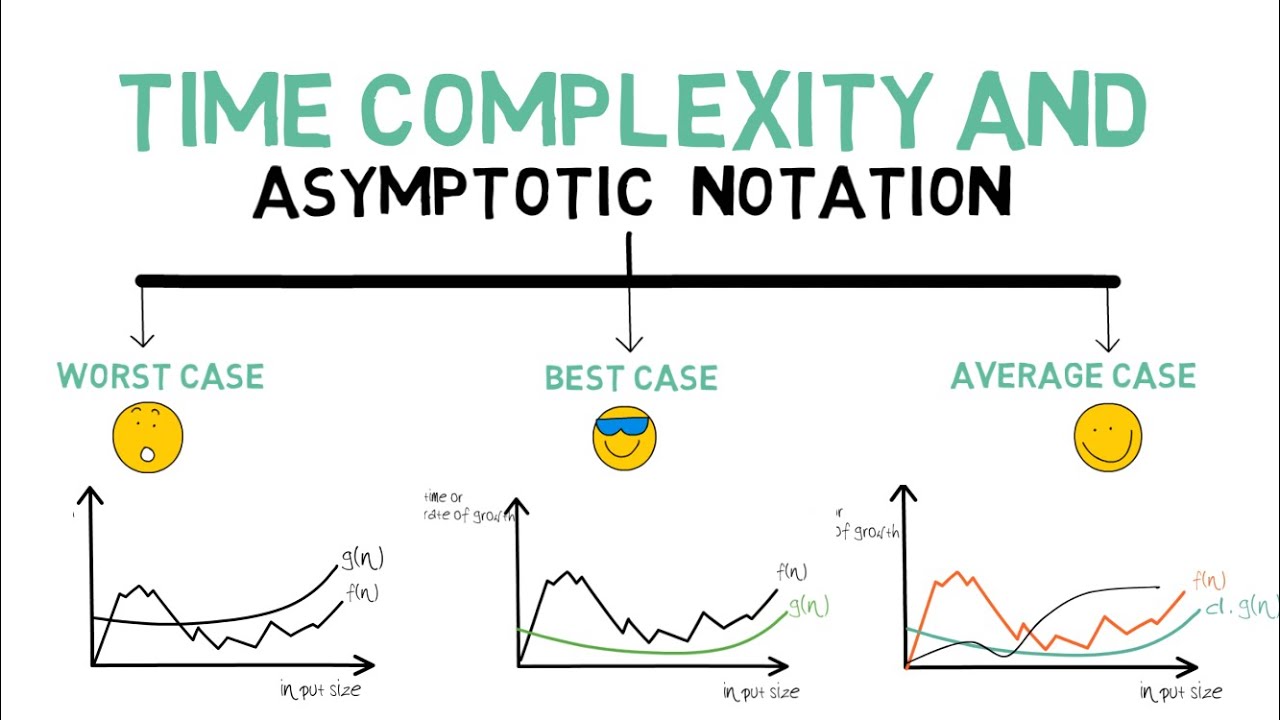


1. Step Count

ເປັນວິທີຄຳນວນເວລາໃນການປະມວນຜົນໂດຍພິຈາລະນາຈຳນວນວຽກທັງໝົດທີ່ອະກໍຣິທຣຶມເຮັດ ຫຼືເປັນການຊອກຫາວ່າຈະຕ້ອງປະມວນຜົນຄຳສັ່ງທັງໝົດທີ່ຢູ່ພາຍໃນອະກໍຣິທຣຶມນັ້ນທັງໝົດຈັກຄັ້ງ.

ສັນຍາລັກ

ສັນຍາລັກເປັນຕຳລາຂອງເວລາທີ່ໃຊ້ໃນການປະມວນຜົນຂອງອະກໍຣິທຣຶມໃດໜຶ່ງໂດຍຈະພິຈາລະນາຈາກຄ່າເມື່ອອະກໍຣິທຣຶມນັ້ນມີປະລິມານຂໍ້ມູນທີ່ຫຼາຍທີ່ສຸດ.



* ຜົນປະໂຫຍດຂອງສັນຍາລັກ Asymptotic
* ເພື່ອໃຊ້ຊັບພະຍາກອນເວລາທີ່ຕ້ອງການຈະໃຊ້ໃນການປະມວນຜົນຂອງອະກໍຣິທຣືມໃດໜື່ງໂດຍສັງເຂບໄດ້ໃນປະລິມານຂໍ້ມູນຕ່າງໆ.
* ໃຊ້ໃນການປຽບທຽບປະສິດທິພາບຂອງອະກໍຣິທຣືມທີ່ເຮັດວຽກຢ່າງດຽວ.
* ການວິເຄາະສັນຍາລັກຈາກອະກໍຣິທຣືມ

ການວິເຄາະສັນຍາລັກຈາກອະກໍຣິທຣືມແມ່ນການພິຈາລະນາຄຳສັ່ງຕ່າງໃນອະກໍຣິທຣືມຊອກຫາຄ່າຂອງສັນຍາລັກ Asymptotic ໃດໜື່ງທີ່ຕ້ອງການ.

* ວິທີອ່ານຄ່າຂອງອະກໍຣິທຣືມ

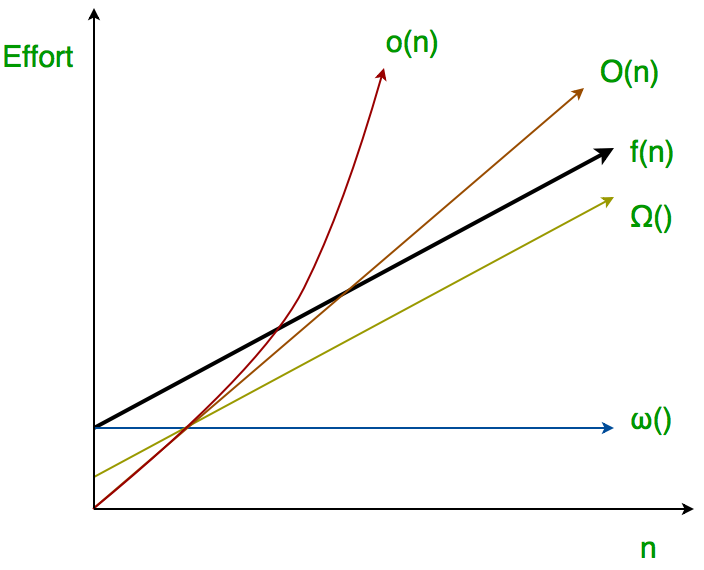
ແມ່ນການອ່ານວ່າອະກໍຣິທຣືມອັນໃດມີປະສິດທິພາບດີກວ່າແມ່ນຄ່າຈາກຕຳລາຂອງອະກໍຣິທຣືມນັ້ນ ໂດຍຕຳລາທີ່ຄ່າໜ້ອຍກວ່າຈະມີປະສິດທິພາບຫຼາຍກວ່າ. ປະເພດຂອງສັນຍາລັກມີຄື:

1. Big O ໃຊ້ເຄື່ອງໝາຍ O()

2. Omega ໃຊ້ເຄື່ອງໝາຍ Ω()

3. Theta ໃຊ້ເຄື່ອງໝາຍ Θ()

4. Little O ໃຊ້ເຄື່ອງໝາຍ o()

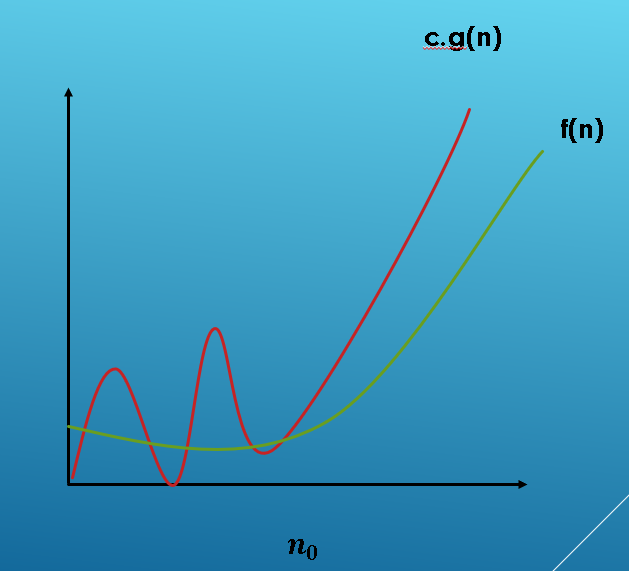


Big O

ແມ່ນການຊອກຫາຂອບເຂດເທິງຂອງຕຳລາຂອງເວລາ (Upper-bound function)

ສຳລັບອະກໍຣິທຣຶມໃດໜຶ່ງ

f(n) = O(g(n)) ກໍຕໍ່ເມື່ອມີຄ່າຄົງທີ່ C>0 ແລະ >0 ທີ່ເຮັດໃຫ້ f(n) ≤￼cg(n) ສຳຫຼັບທູກໆຄ່າຂອງ n, ເມື່ອ n ≥



ຕົວຢ່າງ: ຊອກຫາຄ່າ Big O ຂອງອະກໍຣິທຣຶມທີ່ມີຕຳລາເວລາດັ່ງນີ້: f(n) = 4n+7

ແກ້

ເມື່ອ n ≥ 7

->4n + 7 ≤ 4n + n ≤ 5n

->4n + 7 ≤ 5n

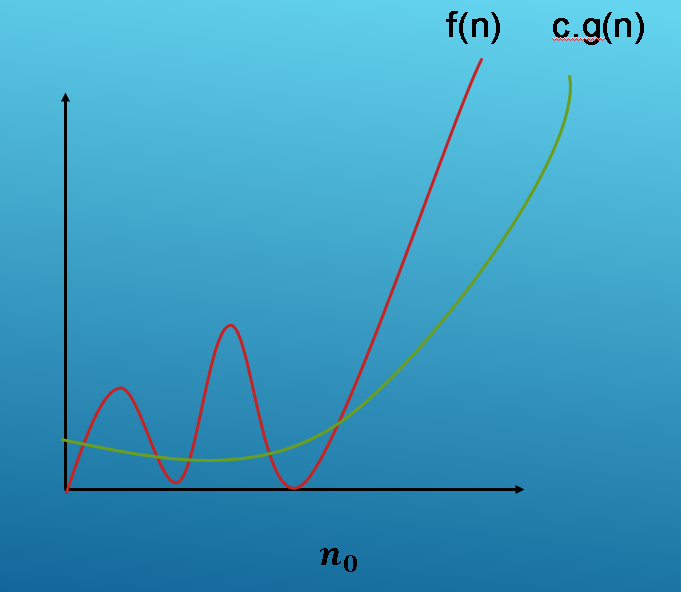
->f(n) ≤ 5n

ດັ່ງນັ້ນ f(n) = O(n) ສຳລັບ n ≥ , ເມື່ອ c=5 ແລະ =7

Omega

ເປັນສັນຍາລັກທີ່ກົງກັນຂ້າມກັບ Big O ເປັນຂອງເຂດລຸ່ມຂອງຕຳລາຂອງເວລາຂອງອະກໍຣິທຣຶມໃດໜຶ່ງວ່າເວລາທີ່ໃຊ້ໃນການນປະມວນຜົນໜ້ອຍທີ່ສຸດແມ່ນເທົ່າໃດ.

f(n) = Ω(g(n)) ກໍຕໍ່ເມື່ອມີຄ່າຄົງທີ່ C>0 ແລະ >0 ທີ່ເຮັດໃຫ້ f(n) ≥ cg(n) ສຳລັບທຸກໆຄ່າຂອງ n, ເມື່ອ n ≥



ຕົວຢ່າງ: ຊອກຫາຄ່າຂອງ Omega ຂອງຕຳລາຂອງເວລາຕຳລາຂອງເວລາຕໍ່ໄປນີ້ f(n) = 5n+17

ແກ້

5n+17 > 5n

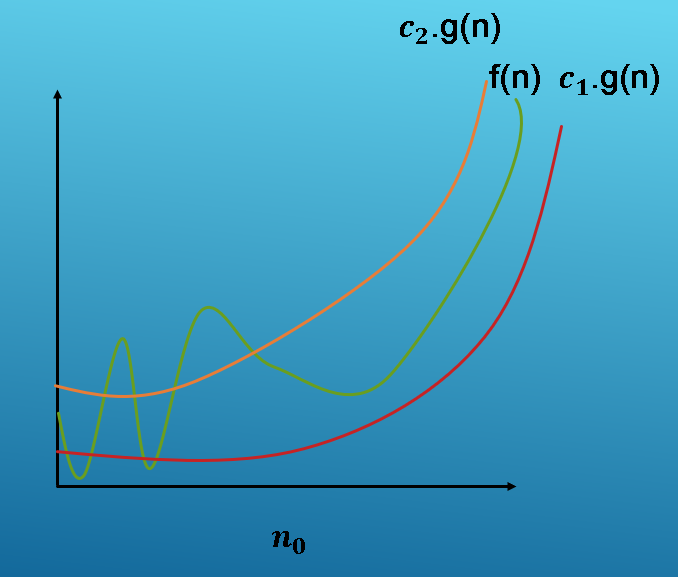
f(n) > 5n

ດັ່ງນັ້ນ ເຮົາໄດ້ f(n) = Ω(n) ∀n, ເມື່ອ c=5

Theta

ເປັນການຊອກຫາຄ່າເຄິ່ງກາງຂອງຕຳລາຂອງເວລາ ໝາຍຄວາມວ່າຊອກຫາຂອບເທິງ ແລະ ຂອບລຸ່ມຂອງຕຳລາດັ່ງກ່າວ

f(n) = Θ(g(n)) ກໍຕໍ່ເມື່ອມີຄ່າຄົງທີ່ >0, >0 ແລະ > 0 ທີ່ເຮັດໃຫ້ g(n) ≤ f(n) ≤ g(n) ສຳລັບທຸກໆຄ່າຂອງ n, ເມື່ອ n ≥



ຕົວຢ່າງ: ຊອກຫາຄ່າຂອວ Theta ຂອງຕຳລາຂອງເວລາຕໍ່ໄປນີ້ f(n)=9n+8

ແກ້

ເມື່ອ n ≥ 8 -> 9n+8 ≤ 10n

9n ≤ f(n) ≤ 10n

f(n)= Θ(n) ∀n ≥ 8, ເມື່ອ = 9 ແລະ = 10

Little O

ແມ່ນມີລັກສະນະຄ້າຍຄືກັບສັນຍາລັກຂອງ Big O, ເປັນການຊອກຫາຄ່າຂອງຕຳລາ

f(n) ທີ່ມີຄ່ານ້ອຍກວ່າຕຳລາ g(n) ຢ່າງຂາດຕົວ f(n) = o(g(n)) ກໍຕໍ່ເມື່ອ f(n) = O(g(n)) ແລະ f(n) ≠ Θ(g(n))

ຕົວຢ່າງ: ຊອກຫາຄ່າຂອງ Little O ຂອງຕຳລາຂອງເວລາຕໍ່ໄປນີ້

f(n) = 4n+5

ແກ້

f(n) = O() ແລະ f(n) ≠ Θ() ເມື່ອ > 25

ສະນັ້ນ f(n) = o()

ຂໍຂອບໃຈ