Chapter 01. Computational Thinking Concept

IT대학 컴퓨터학부
박세영

개요

- What is Computational Thinking?
  - Definition and Vision
  - Common Examples
- Computational Thinking Concepts
  - Abstraction
  - Logical thinking
  - Algorithms
  - Decomposition
  - Debugging
  - …

Computational Thinking: What It Is and Is Not

프로그래밍 자체가 computer science는 아니다
프로그래밍은 문제를 해결하기위해 컴퓨터에게 일을 시키는 방법이다. 일을 시키기 전에 굉장히 많은 과정이 필요.
일을 해결하기 위해 생각하는 방법이 먼저 되어야한다. 그런 것 전체가 computer science에 포함되는 것

- Conceptualizing, not programming
  - Computer Science는 바로 프로그래밍 그 자체가 아닙니다.
- Fundamental, not rote skill
  - 암기하는 것이 아니라 모든 사람들이 알아야 할 필요가 있는 기술
- A way that humans, not computers, think
  - 인간은 영리하고 창의적이고, 컴퓨터는 멍청합니다.
- Idea, not artifacts
  - 컴퓨터 자체를 배우는 것이 아니라 문제해결을 위한 아이디어를 공부하는 것입니다.
- For everyone, everywhere
  - 앞으로는 누구에게나 어디에서나 필요한 것입니다.

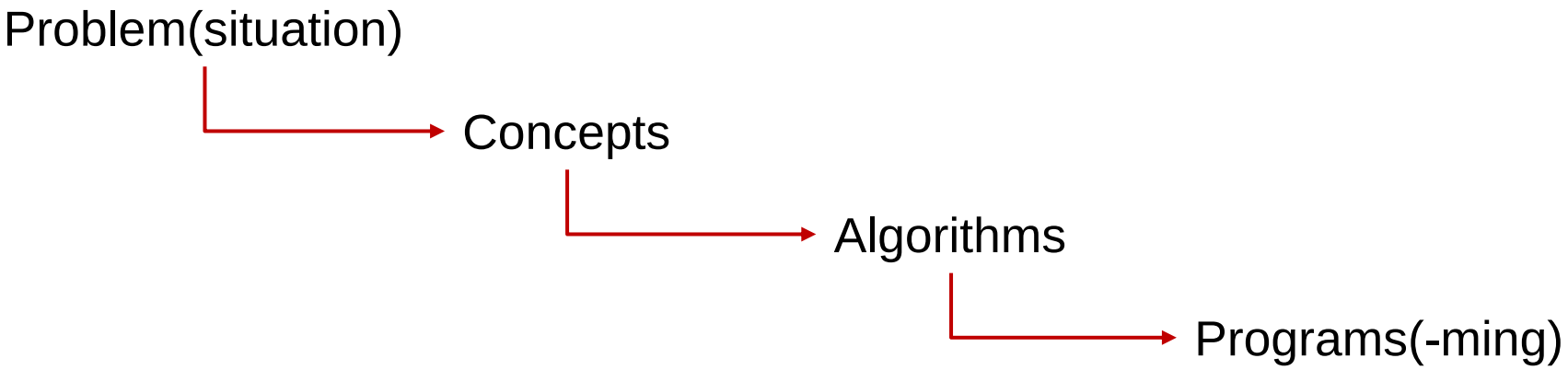
The new curriculum – General Comments

- Remember

  Computational thinking

  프로그래밍 과정 : 요구사항 > Function define > Design > coding > te

  - Computer Science ≠ Programming =! coding
- Before tools, before programming:



- 컴퓨터없이 Computer Science를 배울 수 있다.
  - Computer Science Unplugged (예, Bebras tasks)
- 적당한 때, 컴퓨터를 사용하면 된다.

Computational Thinking Concepts

- Abstraction
  - 주된 아이디어를 정의할 관련 정보만 뽑아 내는 것
- Algorithm Design
  - 문제를 풀기 위한 명령어의 순서를 만들어 내는 것
- Automation
  - 컴퓨터나 기계가 반복적인 일을 하는 것
- Data Analysis
  - 패턴을 발견하거나 통찰력을 개발하여 데이터를 이해하는 것
- Data Representation
  - 적당한 그래프, 차트, 단어, 이미지 등으로 데이터를 묘사하는 것

Computational Thinking Concepts

- Decomposition
  - 데이터, 프로세스나 문제를 풀 수 있는 정도로 나누는 것
- Parallelization
  - 큰 일을 보다 작은 단위로 나누어 동시에 처리하는 것

Decomposition**을 하더라도 순서가 정해져 있어서** Parallelization**하게 할 수 없는 일도 있다.**

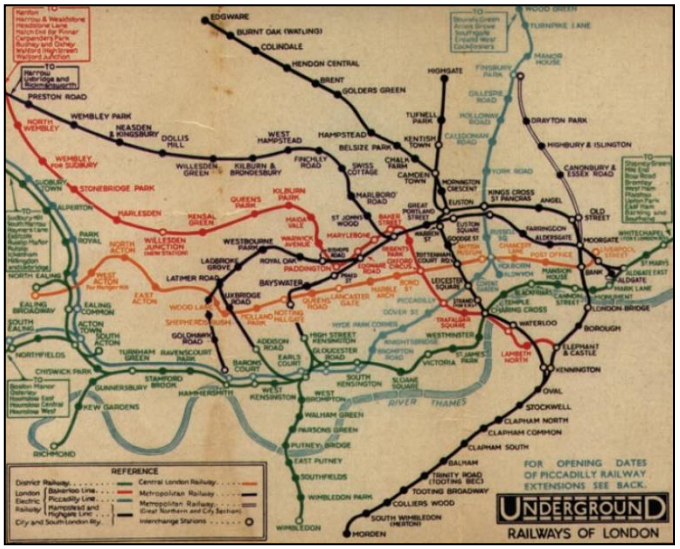
- Pattern Generalization
  - 관찰된 패턴에서 새로운 것을 예측할 수 있는 모델, 규칙, 원리나 이론을 만들어 내는 것
- Pattern Recognition
  - 데이터에서 패턴, 트랜드, 그리고 규칙을 찾아 내는 것
- Simulation
  - 실세계의 프로세스를 닮은 모델을 개발하는 것

실세계의 상황을 컴퓨터로 구축하여 시뮬레이션 해보는 것

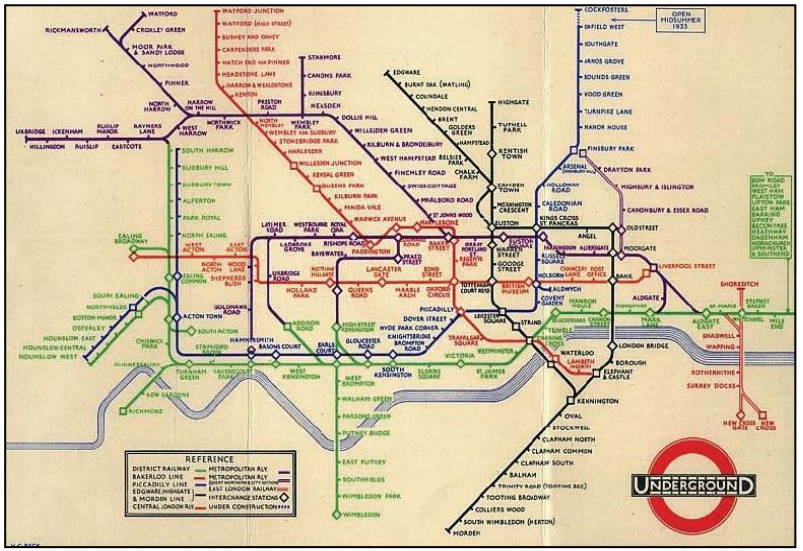
Abstraction

Abstraction

- 중요한 세부사항만 빼서 다른 분야에도 적용시킬 수 있는 원리를 찾아 내는 것
- 자세한 것은 없애서 전체를 간단하게 만들고 공통적이고 기본적인 것에 집중하는 것
- 자세한 것의 수준을 신중하게 선택하는 것



1928 map of London underground system



1931 map of London underground system

First in first out

LIFO vs. FIFO

- 다음 중 다른 것과 다른 것은 ?

[1] People standing in line at the store

[2] List of print jobs waiting to be printed

[3] Set of tennis balls in their container LIFO

[4] Vehicles lined up behind a toll booth

[5] Patients waiting to see the doctor

Tree vs. Graph

- 다음 중 다른 것과 다른 것은 ?

[1] Files and directories on a hard disk.

[2] Parents and children in a pedigree chart.족보

[3] NCAA basketball tournament.

[4] My closest friends on Facebook / Twitter. GRAPH

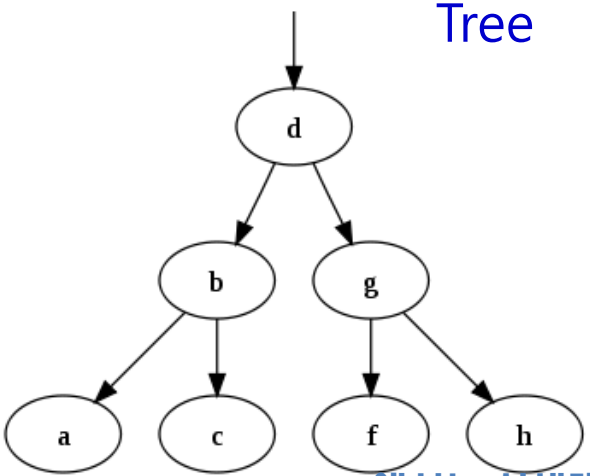
[5] Classification of animals.

Abstraction: Tree vs. Graph

Tree

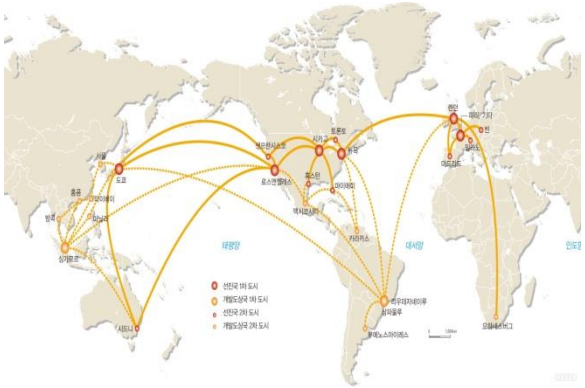
- 토너먼트, 디렉토리

Tree

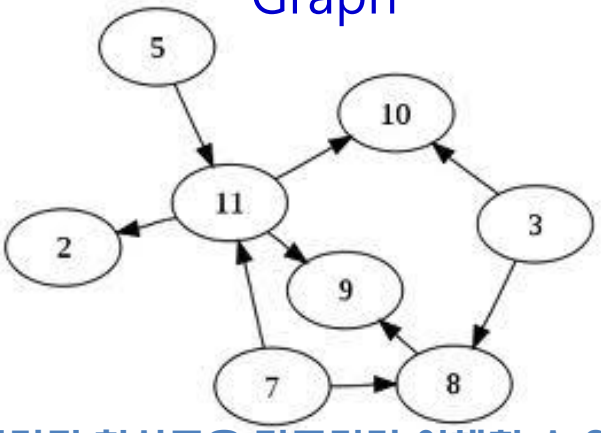


Graph

- 고속도로, Facebook



Graph



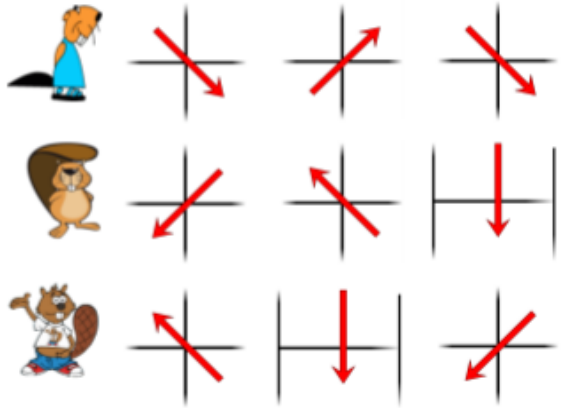
COMPUTATIONAL THINKING**에서는 실세계에서 일어나는 여러가지 현상들을 컴퓨터가 이해할 수 있는 모양으로 바꾸는 것**

앞으로TREE**나** GRAPH**를 어떤 식으로 컴퓨터에게 가르치고 표현하게 될 것인가를 배움.**

Algorithm Exploring

Problem Description

- 3 마리 비버가 오른 쪽 명령어를 사용하여 버섯을 찾을 수 있다.



Question

- 이 명령어를 따라 마지막으로 각 비버들이 도착한 곳은 어디인가 ?

Algorithms Design

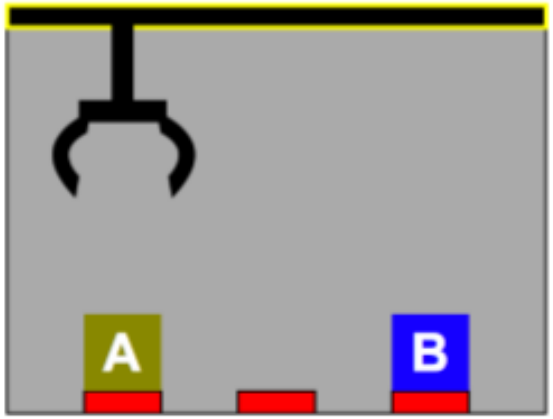
- 무엇이 알고리즘인가 ?
  - 문제를 푸는 스텝(Steps)의 나열(Sequence)이다.
- 왜 중요한가 ?
  - 컴퓨터 프로그램은 특별한 업무를 처리하기 위해 알고리즘을 수행한다.
- 당신에게 주어진 업무
  - 맛있는 라면을 끓이기 위해 어떤 절차를 따라야 하나 ?

Algorithm Design

Problem Description

- 다음 크레인은 6개의 명령어로 움직이고 있다.

left
right
up
down
grab
let go



Question

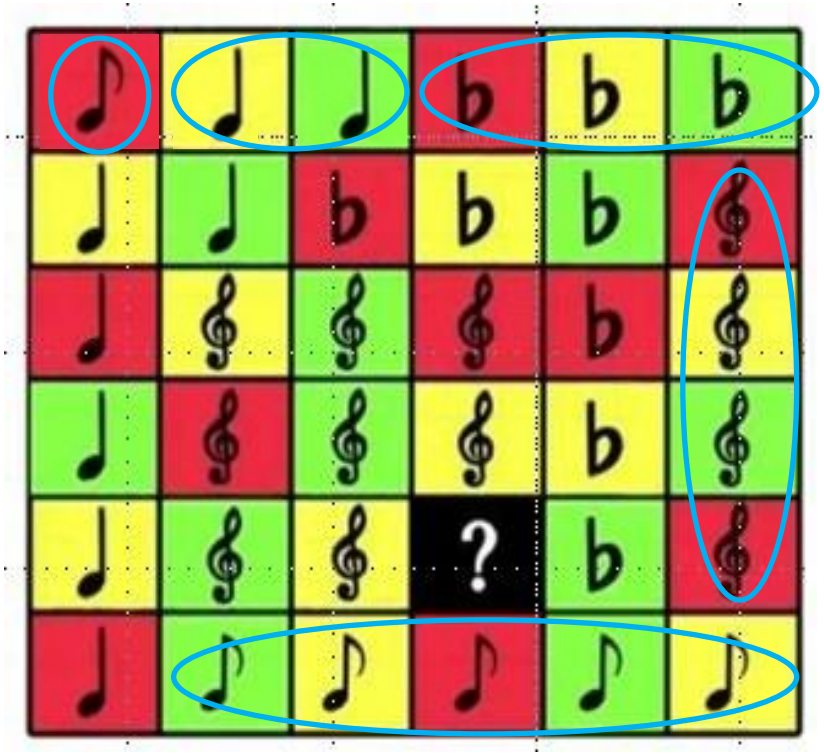
- 위의 명령어를 사용하여 A와 B의 위치를 바꿀 수 있는 알고리즘을 만드세요.

Pattern Recognition

패턴을 이해하면 그 다음에 일어날 일을 예측할 수 있기 때문에 패턴 인식은 컴퓨터 사이언스에서 굉장히 중요한 공부할 거리가 된다

Problem Description

- 물음표에 들어 갈 음표와 색깔은 무엇인가 ?



빨간색 바탕에 b

Patterns Recognition

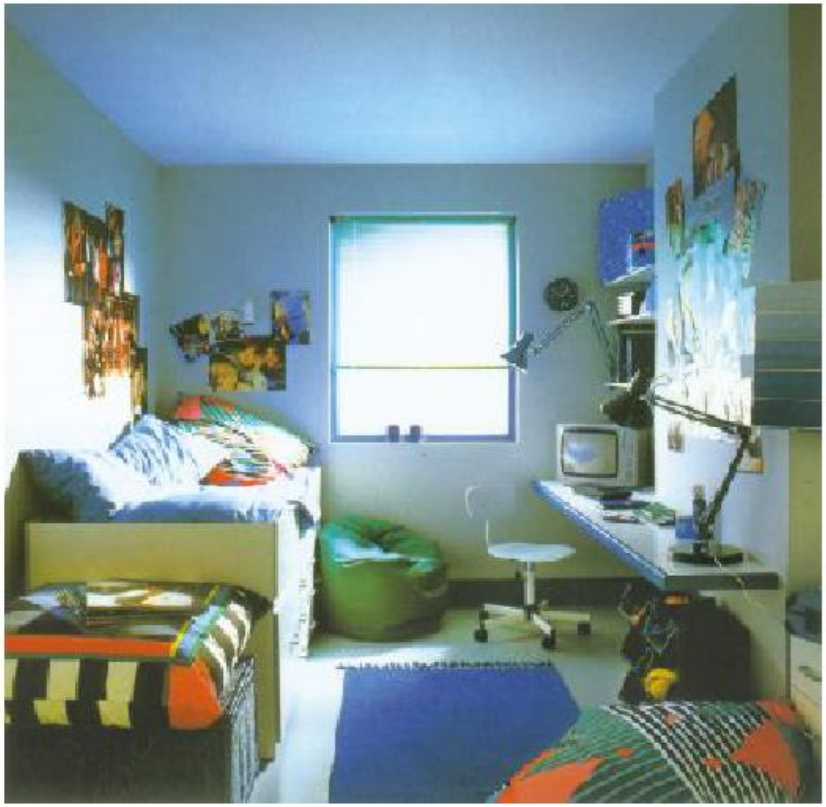
- 문제를 당신에게 쉽게 만드는 방법은 무엇인가?
  - 패턴을 찾음으로써 우리는 일반적인 문제에 대해 예측(Prediction) 하고 규칙을 만드는 일로 일반적인 문제를 푼다.
  - 아이들은 음악에서 반복되는 멜로디를 인식함으로써 다음에 나오는 음을 예측할 수 있다.
- 패턴을 인식하면 미래를 예측할 수 있다.
  - 다음 규칙을 찾아 빈칸을 채우세요.
  - 7 ♥ 2 = 1
  - 39 ♥ 9 = 3
  - 26 ♥ (9 ♥ 6) = 2
  - 161 ♥ (13 ♥ 5) = □

Caching

- 미리 짐을 싸두는 것

  컴퓨터의 cach메모리

  - 한 번 방문한 웹 사이트는 더 빨리 볼 수 있다.



home

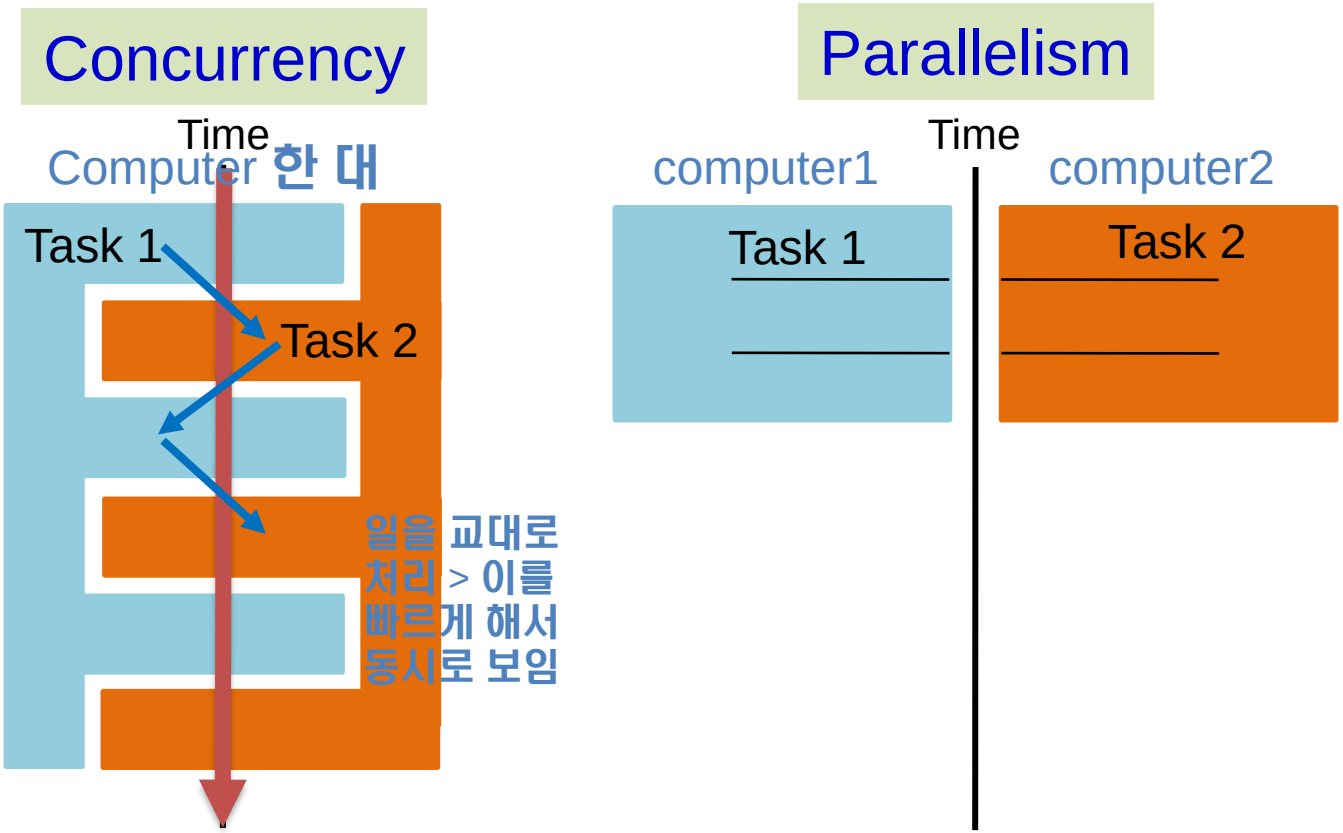
locker

knapsack

Concurrency(동시성)

Concurrency vs. Parallelism

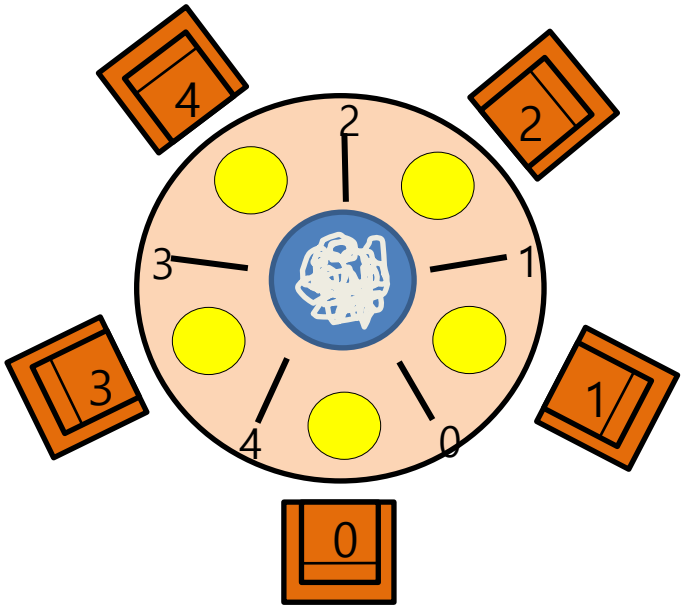
- Concurrency는 동시에 실행되는 것처럼 보이는 논리적인 용어
- Parallelism은 실제로 여러 개의 컴퓨터에서 작업이 동시에 처리되는 물리적인 용어



Concurrency

Dining Philosophers

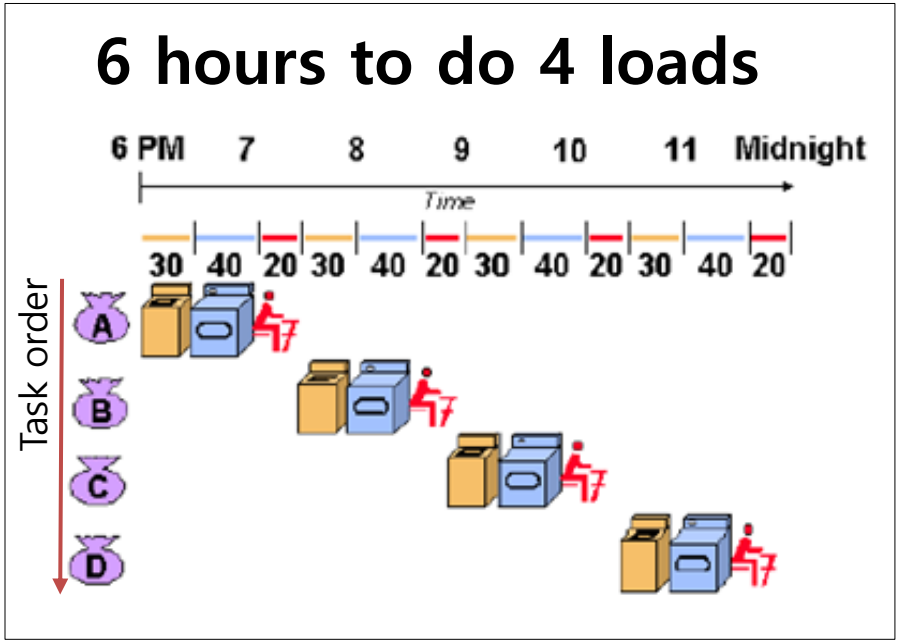
- 다섯 명의 철학자가 둥근 테이블에 앉는다.
- 각 철학자는 생각하고 먹는 것을 번갈아 가면서 한다.
- 테이블의 중앙에는 큰 접시에 스파게티가 담겨있다.
- 한 철학자가 스파게티를 먹는 데는 2 개의 포크가 필요하다.
- 어떻게 하면 누구도 배고프지 않게 식사가 계속될 수 있을까 ?



Pipelining

Problem Description

- 세탁기 하나, 건조기 하나가 있다. 빨래를 마치고 난 다음 20분은 쉬어야 한다.



Question

- 시간을 단축하기 위해 어떻게 해야 하나 ?

Debugging

- 무엇이 debugging인가 ?
  - 프로그램에서 버그를 찾고 고쳐서 원래 원하는 대로 동작하게 만드는 일
- Debug
  - Info
  - Notice
  - Warning

  심한 순서에 따라
- Error
  - Critic
  - Alert
  - Panic

  심한 순서에 따라

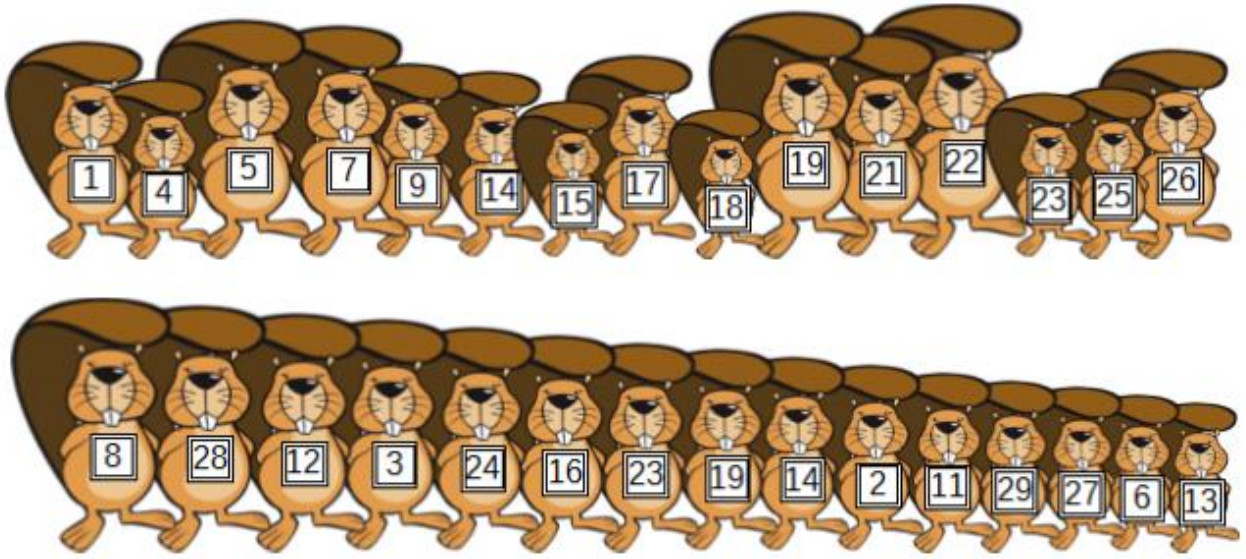
Sorting and Searching

- 컴퓨팅 알고리즘의 기초
  - Sorting: 크기 순서대로 줄을 세우는 작업.
  - Searching: 내가 원하는 것을 빨리 찾는 방법.

Searching

Problem Description

- 아래와 같이 등번호가 새겨진 15명의 선수들이 있다.



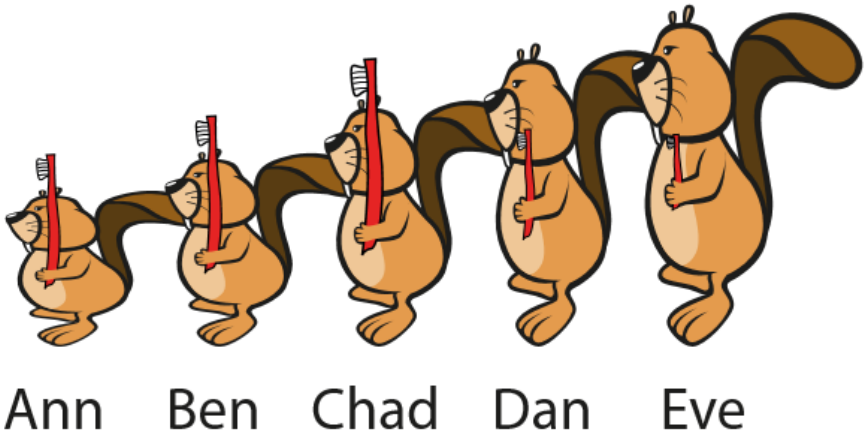
Question

- 두 번째 팀에서 첫 번째 팀과 같은 등번호를 가진 비버는 몇 명인가 ?
- 첫 번째 팀에서 두 번째 팀과 같은 등번호를 가진 비버는 몇 명인가 ?

Sorting

Problem Description

- 각 비버들은 몸에 맞지 않는 칫솔을 가지고 있다.
- 그래서 엄마가 Eve와 Chad 칫솔을 서로 바꾸라고 했다.
- 그런 다음 다시 Ann과 Chad의 칫솔을 바꾸라고 했다.

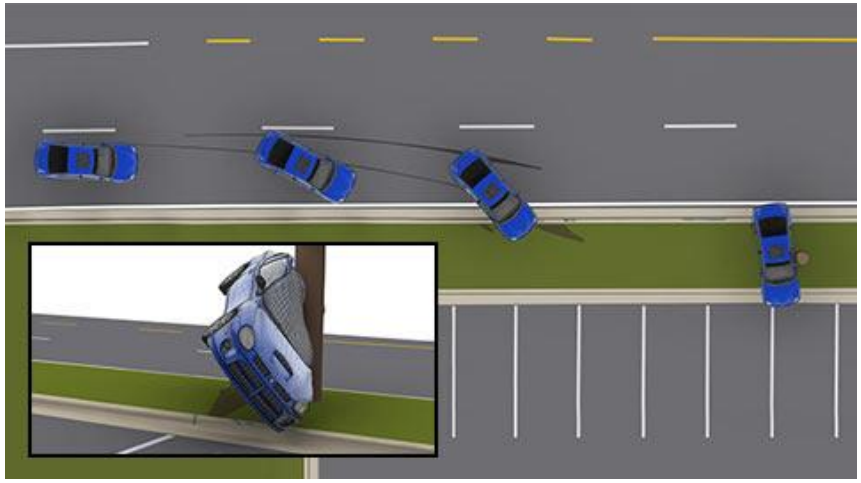


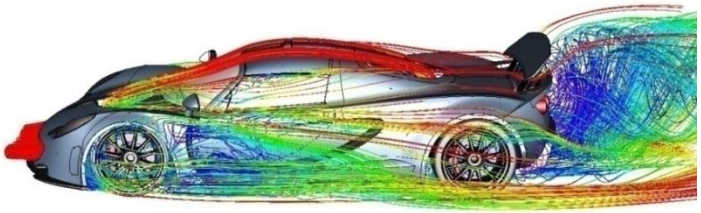
Question

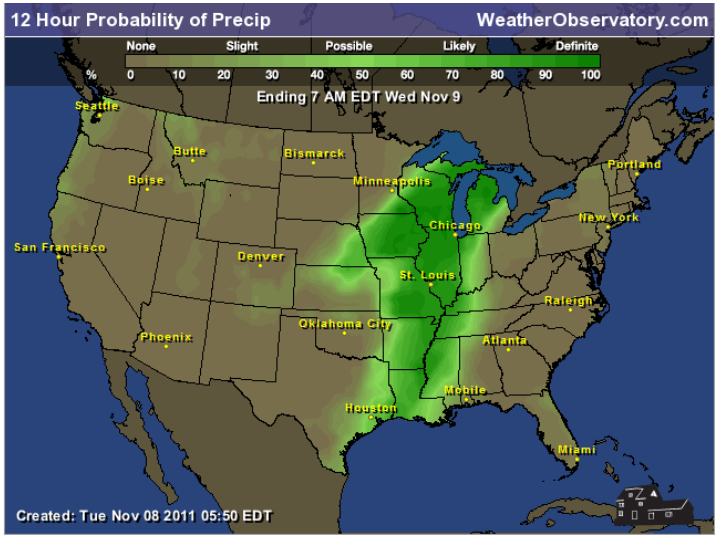
- 아직도 누구와 누구의 칫솔을 서로 바꾸어야 할까요 ?

Simulation

- 차량의 컴퓨터 시뮬레이션
  - 실세계의 프로세스를 닮은 모델을 개발하는 것



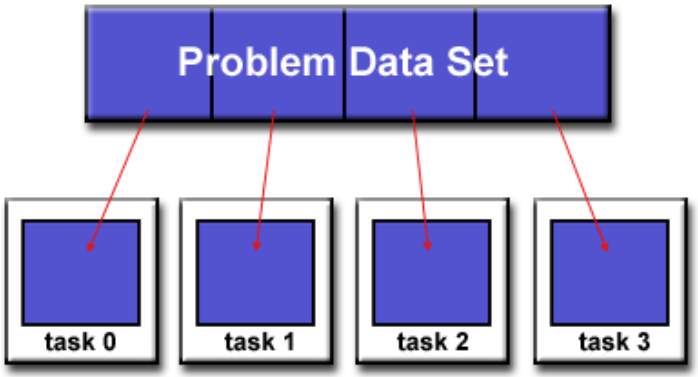




Decomposition

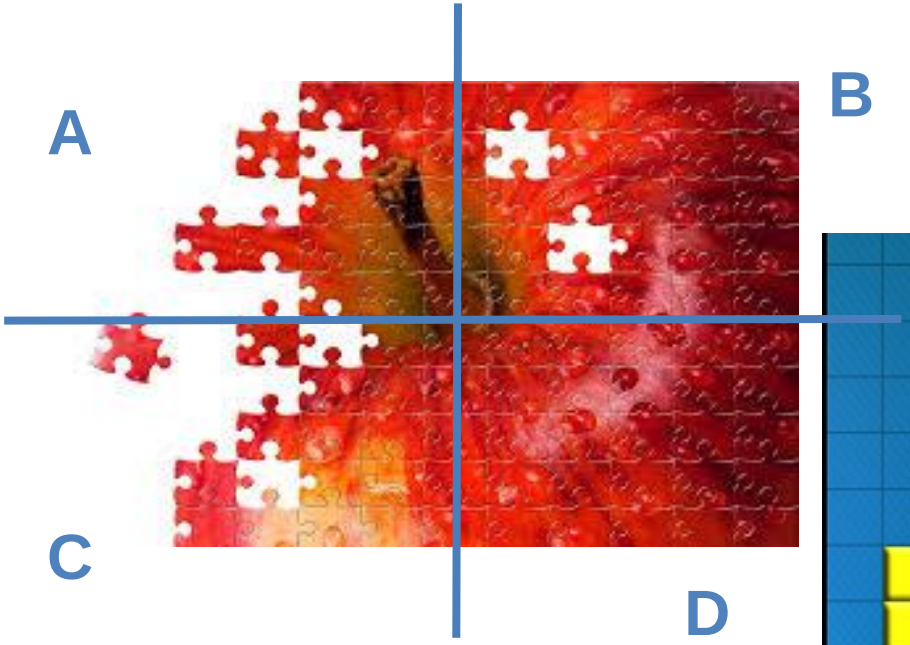
- 큰 문제를 우리가 풀 수 있는 작은 문제로 쪼갠다.
  - 컴퓨터 게임과 같이 큰 규모의 프로그래밍 프로젝트

(Storytelling, Planning, Design, Algorithm, Coding, Animation, Graphics, Sound, Debugging)

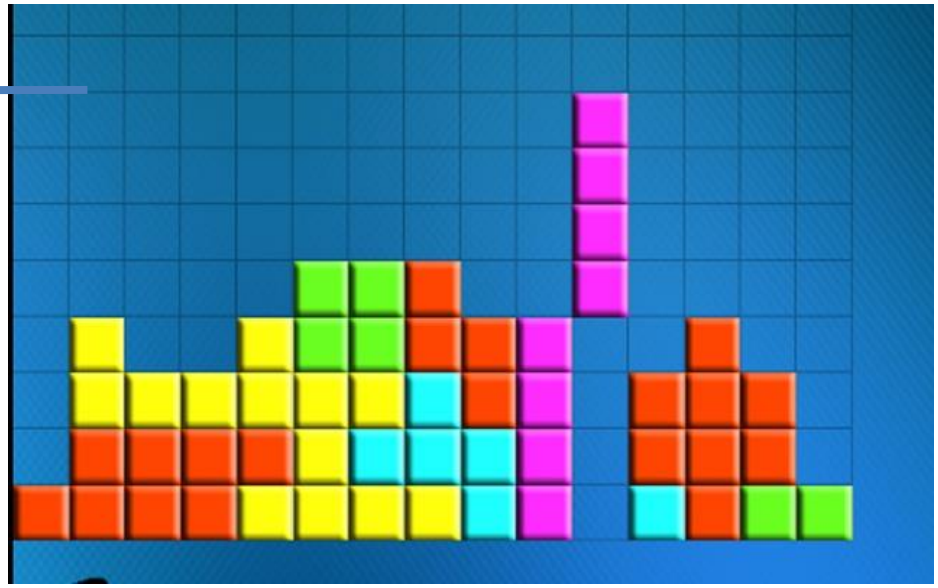


Composition

- 작은 문제의 해결책을 모아서 큰 해결을 하는 것
  - 퍼즐 맞추기, 테트리스 게임



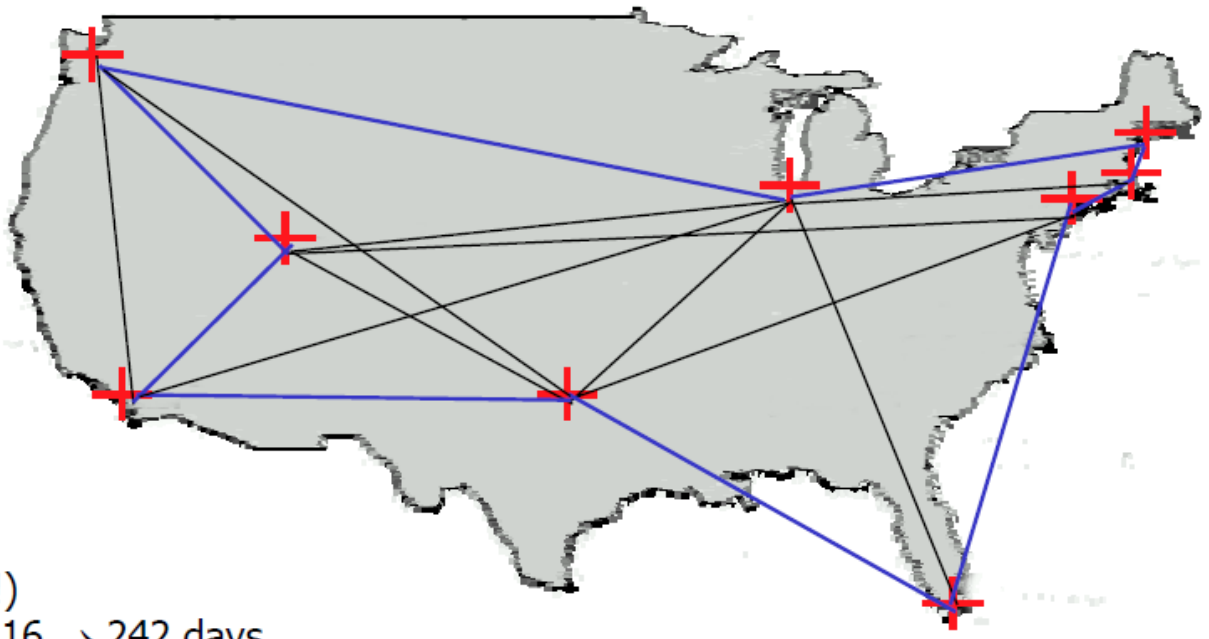
ABCD **사람이 나누어서 동시에 일을 해결하고 결과를 모으는 것을** COMPOSITION



이렇게 나누어서 일을 하는 것을 Parallelization

Traveling Salesman

- 여러 도시들을 방문해야 할 때
  - 가장 짧은 길이의 여행은 무엇인가 ?

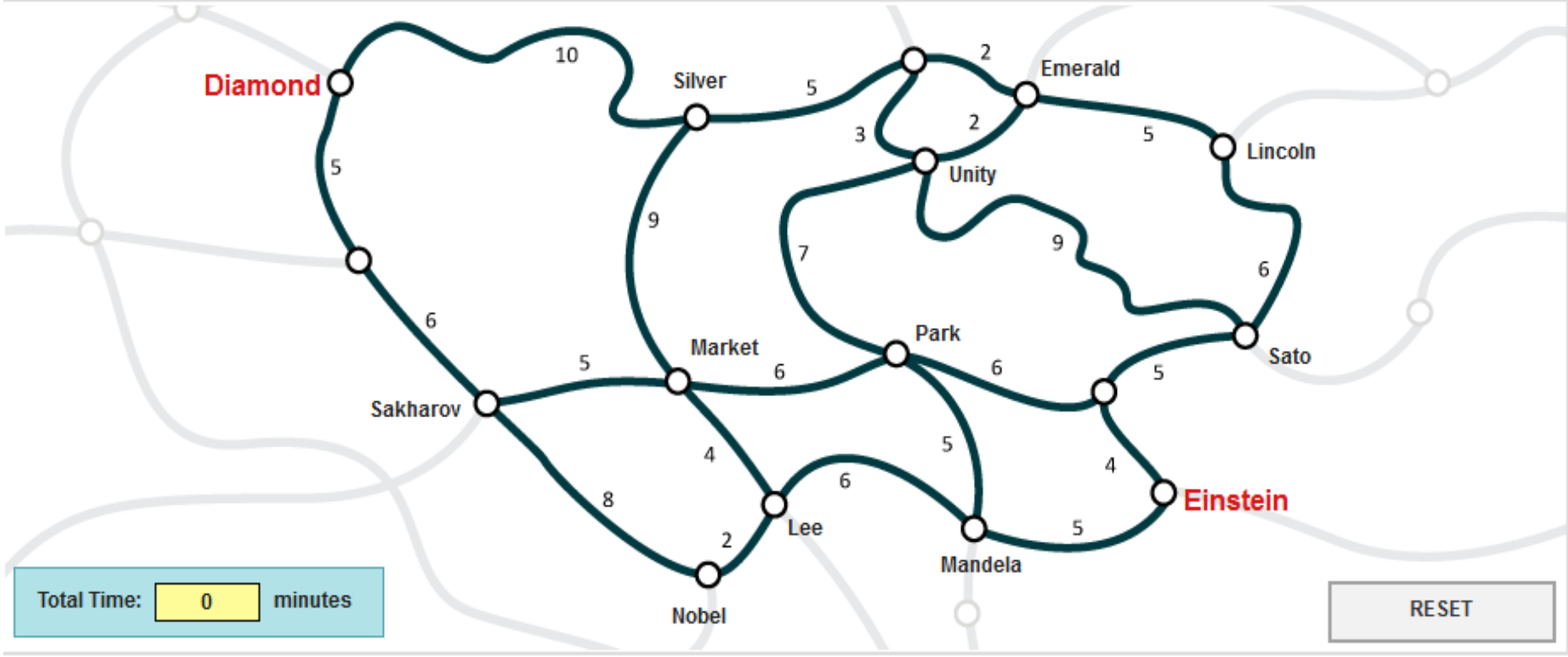


O(n!)
n = 16 → 242 days
n = 25 → 5x10^15 centuries

Shortest Path – PISA task

빠른 길 찾기

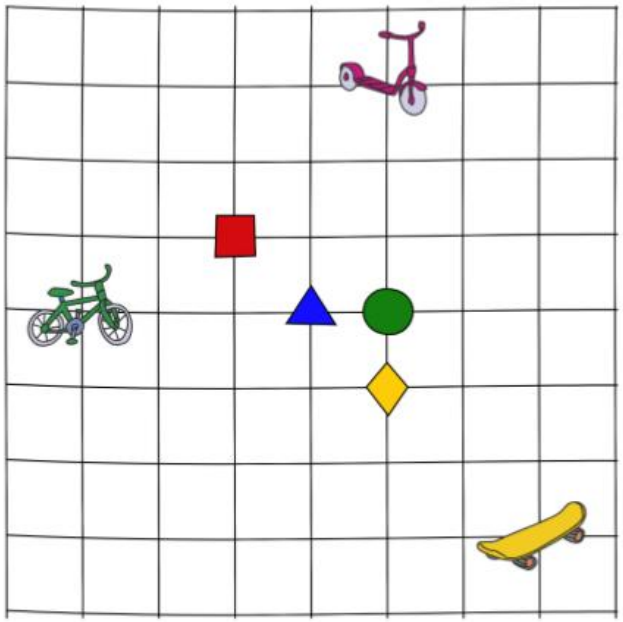
- Einstein에서 Diamond 까지 31 min이 걸린다면 어떤 길일까 ?



Three Friends

Problem Description

- 세 명의 친구가 만나기로 했다.

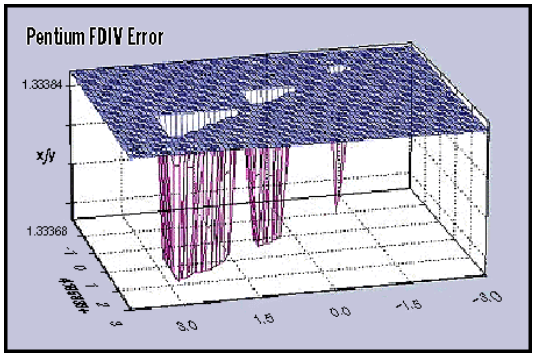


Question

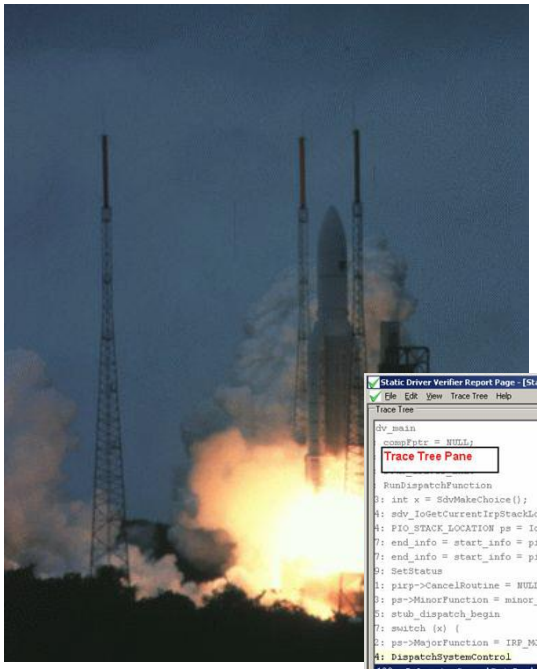
- 어떤 장소에서 만나는 것이 세 친구가 움직이는 전체 거리가 가장 작은가 ?

Correctness =에러를 줄이는 일

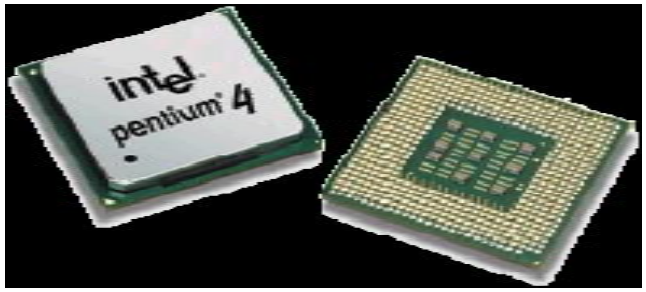
- 시간과 돈을 절약하기 위해 에러를 고치는 일
  - 실패를 줄이는 일



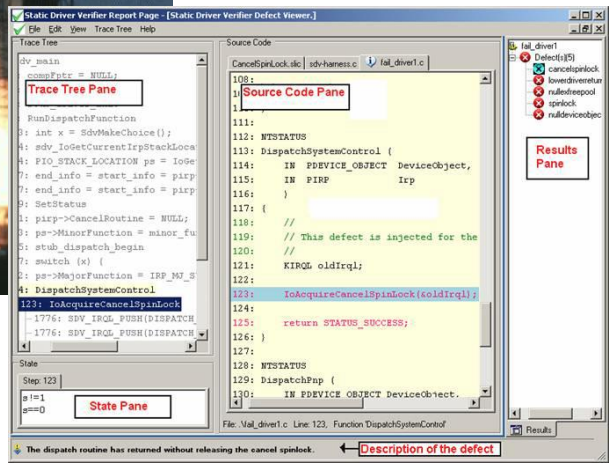
Intel Pentium FPU error



Ariane 5 failure



Now Intel uses formal verification



Now Microsoft uses formal verification

Distributed Computing

Big Data Center

- 하나의 일을 여러 컴퓨터에 분산해서 처리하는 것

CT는 매일 일어나는 일

전화번호부에서 내 이름 빨리 찾기

- 어떻게 하면 1,000,000명에서 빨리 내 이름을 찾지 ?

기차 표 빨리 사기

- 사람이 서있는 줄이 세 줄일 때 어떻게 하면 표를 빨리 살까 ?

Other Daily Examples (by Jeanette Wing)

- Putting things in your child's knapsack for the day
  - Pre-fetching and caching
- Taking your kids to soccer, gymnastics, and swim practice
  - Traveling salesman (with more constraints)
- Cooking a gourmet meal
  - Parallel processing: You don't want the meat to get cold while you're cooking the vegetables
- Cleaning out your garage
  - Garbage collection: Keeping only what you need vs. throwing out stuff when you run out of space
- Storing away your child's Lego pieces scattered on the LR floor
  - Using hashing (e.g., by shape, by color)
- Doing laundry, getting food at a buffet
  - Pipelining the wash, dry, and iron stages; plates, salad, entrée, dessert stations
- Even in grade school, we learn algorithms (long division, factoring , GCD, …) and abstract data types (sets, tables, …)

CT is Everywhere

- Computational Thinking에 의한 새로운 발견
  - DNA mapping
  - 새로운 물질의 개발
  - 새로운 공식
  - 새로운 기술