알고리즘

담당교수: 염대현

교과 기본 사항

- LMS 공지사항의 과목 기본사항 및 출석 규정 필독

- 담당교수
  - 염대현 (daehyun.yum@gmail.com)
  - 문의는 LMS가 아닌 이메일 이용!

- 평가
  - 출석 10% + 기말고사 90%

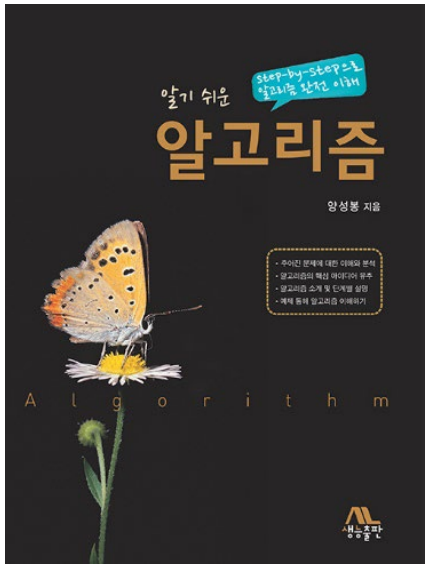
- 출석
  - 주별 1차시만 생성되며, 전체 수업일수 5분의 4이상 출석해야 함
  - 강의 전체 시청 + 과제물 제출

- 강의자료
  - LMS 홈페이지

교재

- Textbook
  - 제목: (알기 쉬운) 알고리즘
  - 저자: 양성봉
  - 출판사: 생능출판사
  - 주의: 본 교과는 프로그래밍 과목이 아님!

초판

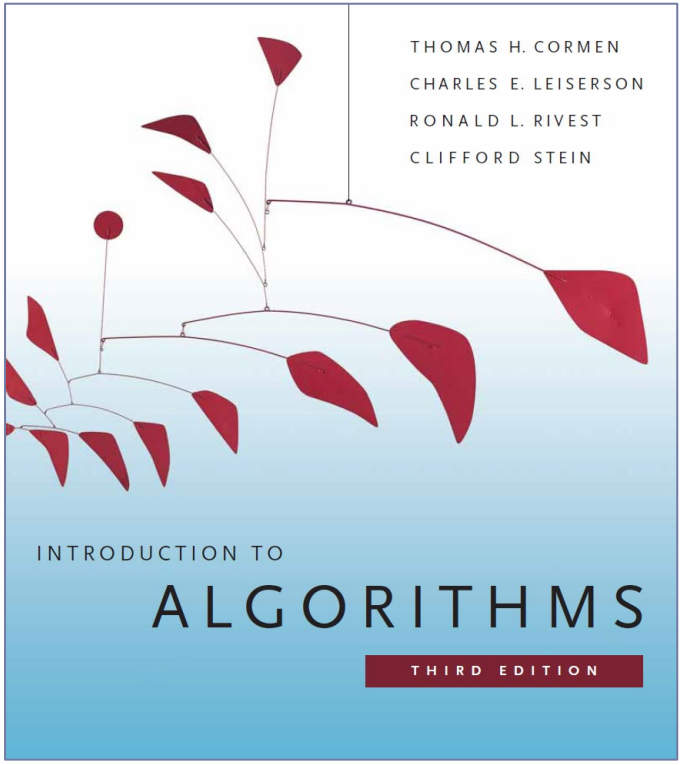


개정판

[참고] 본 수업을 위해서는 초판, 개정판 어느 것을 사용해도 무방함

참고도서

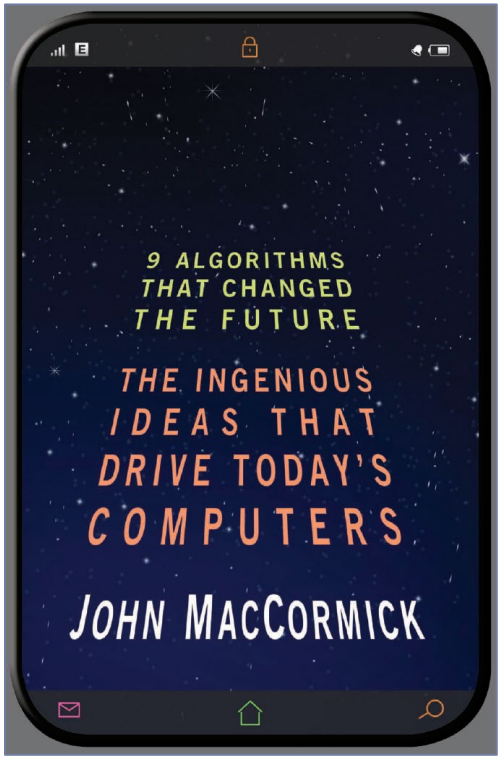
- Introduction to Algorithms



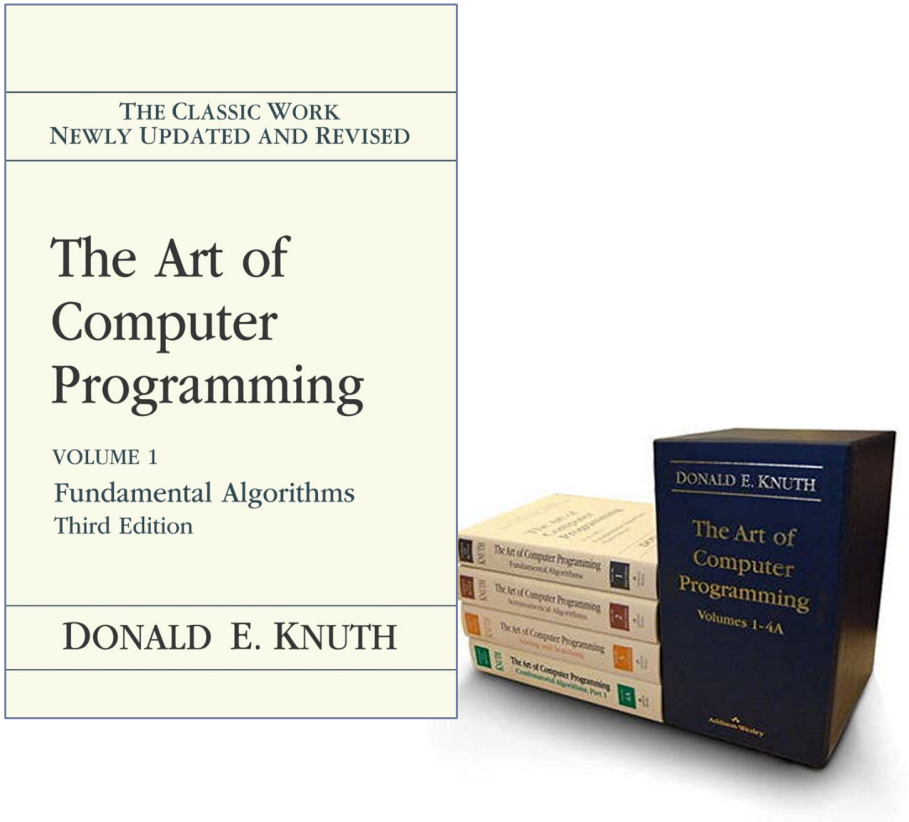
- Algorithms Unlocked

참고도서

- Nine Algorithms That Changed the Future



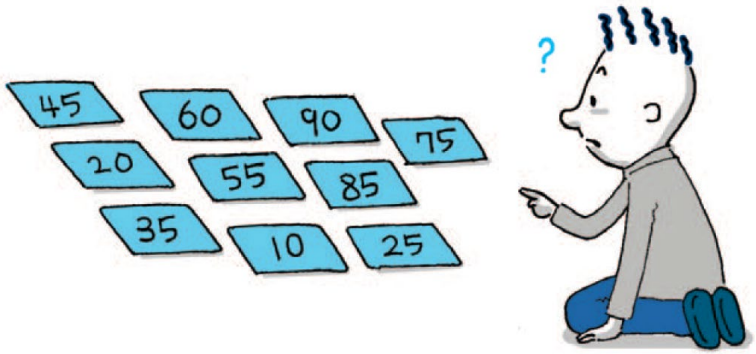
- The Art of Computer Programming



Chapter 1. 알고리즘의 첫걸음

1.1 최대 숫자 찾기

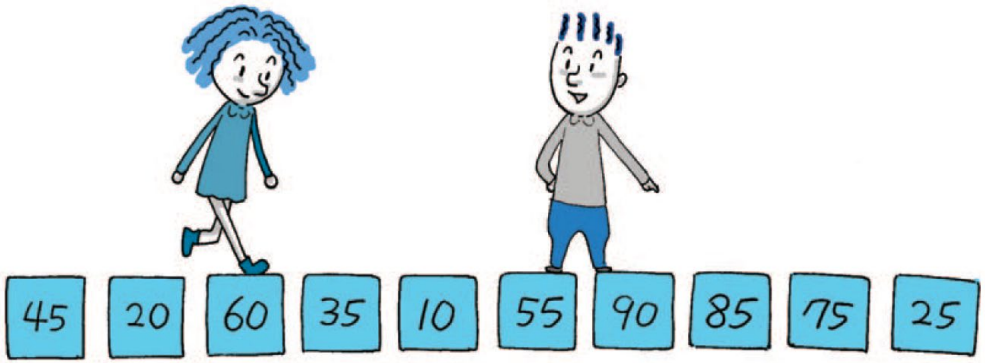
- 문제
  - 최대 숫자 찾기



- 알고리즘
  - 가장 큰 숫자가 적힌 카드를 찾는 한 가지 방법은 카드의 숫자를 하나씩 비교하면서 본 숫자들 중에서 가장 큰 숫자를 기억해가며 진행하는 방법일 것이다. 마지막 카드의 숫자를 본 후에, 머릿속에 기억된 가장 큰 숫자가 적힌 카드를 바닥에서 집어든다.
  - 순차탐색(Sequential Search): 카드를 한 장씩 차례대로 (주어진 순서대로) 읽어가며 찾는 방법

1.2 임의의 숫자 찾기 (1)

- 문제
  - 특정 숫자가 적힌 카드를 찾는 것을 생각해 보자. 45, 20, 60, 35, 10, 55, 90, 85, 75, 25가 각각 적힌 카드가 바닥에 펼쳐져 있다. 이 중에서 85가 적힌 카드를 찾아보자.



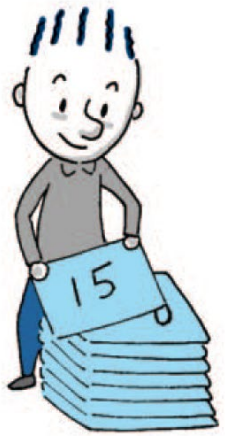
- 알고리즘
  - 최대 숫자 찾기처럼 머릿속에 85를 기억하고 바닥에 펼쳐진 카드를 차례대로 한 장씩 읽으며 85가 적힌 카드를 찾는다. 이 역시 순차탐색을 이용한 것이다.

1.2 임의의 숫자 찾기 (2)

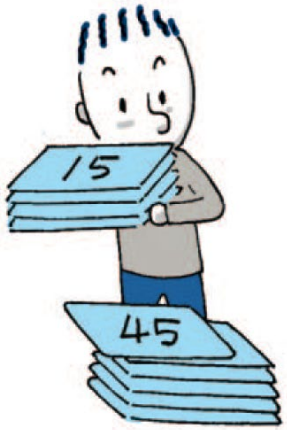
- 문제
  - 10장의 카드가 오름차순으로 정렬되어 있다고 가정하자. 만일, 10장의 카드가 옆의 그림처럼 쌓여 있고, 첫번째 카드의 숫자 15만 보이는 경우, 그중에서 임의의 숫자인 85를 찾고자 할 때, 순차탐색보다 더 효율적인 방법은 없을까? 즉, 카드가 정렬되어 있다는 정보를 어떻게 활용할 수 있을까?

- 핵심아이디어
  - 순차탐색으로 찾을 경우, 앞쪽 8장의 카드를 읽은 후 85를 찾는다. 그러나 카드가 정렬되어 있으므로, 만일 85가 카드의 뒷부분에 있는 것이 확실하면 앞부분은 탐색할 필요가 없다. 그러면 85가 뒷부분에 있다는 것을 어떻게 알 수 있을까? 중간에 있는 카드의 숫자인 45(혹은 55)와 85를 비교해보면 된다.

1.2 임의의 숫자 찾기 (3)



순차탐색으로 1번 비교 후



이진탐색으로 1번 비교 후

- 중간 카드 한 장을 읽어 85와 비교해보는 것이 순차탐색보다 훨씬 빠르게 목표(85가 적힌 카드)에 다가감을 알 수 있다.

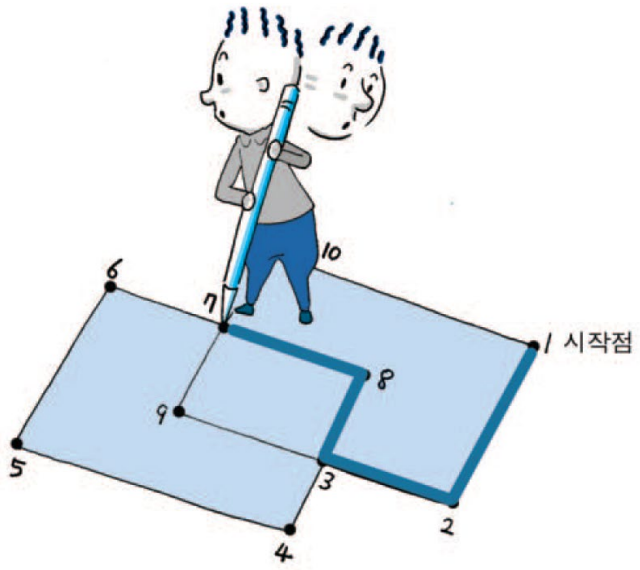
- 알고리즘
  - 이처럼 오름차순으로 정렬된 데이터를 반으로 나누고, 나누어진 반을 다시 반으로 나누고, 이 과정을 반복하여 원하는 데이터를 찾는 탐색 알고리즘을 이진탐색 (Binary Search)이라고 한다.

1.3 동전 거스름돈

- 문제
  - 물건을 사고 거스름돈을 동전으로 받아야 한다면, 대부분의 경우 가장 적은 수의 동전을 받기 원한다.
    - 예: 730원 = 500원 1개 + 100원 2개 + 10원 3개 = 총 6개
- 알고리즘
  - 동전 거스름돈 문제를 해결하는 알고리즘은 남은 거스름돈 액수를 넘지 않는 한도에서 가장 큰 액면의 동전을 계속하여 선택하는 것이다.
  - 이러한 종류의 알고리즘을 그리디(Greedy) 알고리즘이라고 하며, 동전 거스름돈 문제에 대한 그리디 알고리즘은 4.1절에서 보다 상세히 살펴볼 것이다.

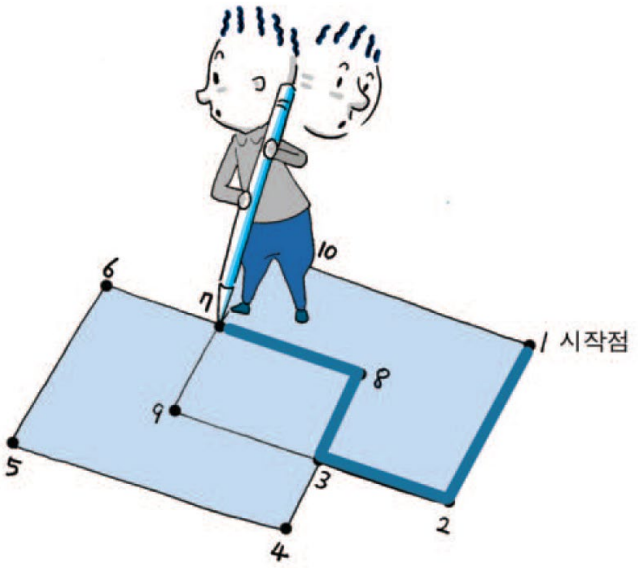
1.4 한붓그리기 (1)

- 문제
  - 한붓그리기 문제는 그래프의 어느 한 점에서 출발하여 모든 선분을 한 번만 지나서 출발점으로 돌아오되, 궤적을 그리는 동안 연필이 종이에서 떨어져서는 안 된다. 단, 한 점을 여러 차례 방문하여도 괜찮다.
- 핵심아이디어
  - 오른쪽 그림은 점 1에서 출발하여 점 2를 지나고, 점 3과 점 8을 거쳐서 점 7에 도착한 상태를 나타내고 있다. 점 7이 현재 점이라 하자. 현재 점으로부터 점 6, 9 또는 10 중에서 어디로 진행해야 할까?



1.4 한붓그리기 (2)

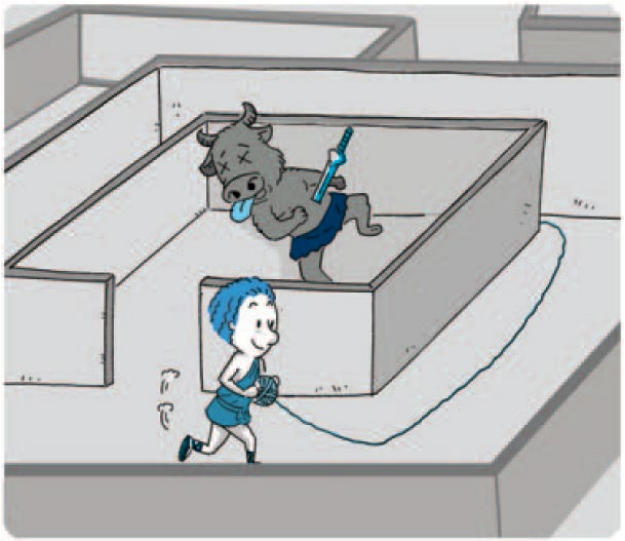
- 핵심아이디어
  - 점 6으로 가면 5, 4, 3, 9, 7, 10을 거쳐서 점 1로 돌아올 수 있다.
  - 점 9로 가면 3, 4, 5, 6, 7, 10을 거쳐서 점 1로 역시 돌아올 수 있다.
  - 점 10으로 가면, 점 1로 갈 수밖에 없고 3, 4, 5, 6, 7, 9 사이의 선분을 지나가기 위해서는 연필을 떼어야 한다.



- 알고리즘
  - 선택한 선분들이 삭제된다고 생각했을 때, 그래프가 disconnect 되지 않도록 다음 선분을 선택한다.

1.5 미로 찾기 (1)

- 문제
  - 그리스 신화에서 지중해 크레타 섬의 미노스 왕은 황소 머리에 하반신은 사람인 미노타우로스에게 제물로 받치기 위해 아테네의 젊은 남녀를 조공으로 요구하였다. 조공으로 바쳐진 젊은이들은 지하 미로에 갇혔고, 미로를 탈출하지 못하면 미노타우로스에게 잡혀먹혔다.

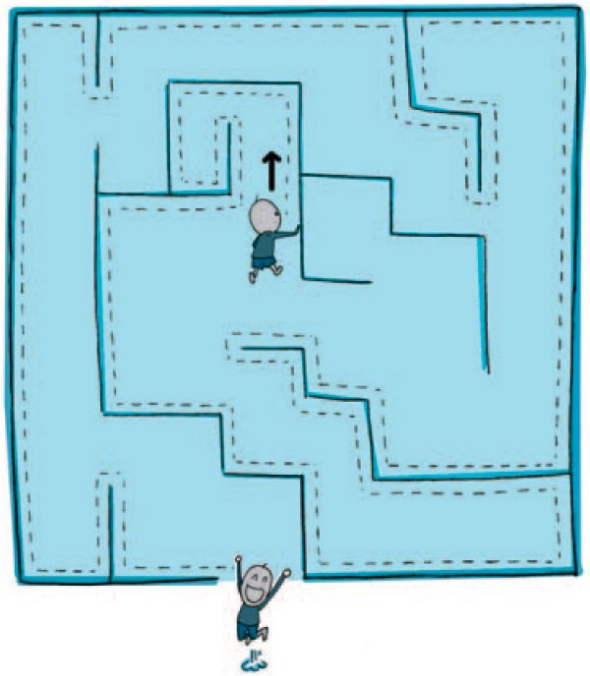


  - 아테네의 청년 테세우스는 자발적으로 제물이 되기로 결심하여 조공으로 바쳐졌다. 그는 다행히 미노스 왕의 딸 아리아드네의 충고로 칼과 함께 실타래를 가지고 실을 풀면서 미로에 들어갔다. 그리고 마침내 테세우스는 미노타우로스를 칼로 죽이고, 실을 다시 감으면서 미로를 빠져나왔다.

1.5 미로 찾기 (2)

- 만약 실타래나 도와주는 사람이 없는 상황이라면 어떻게 미로를 빠져나올 수 있을까?
- 지하의 미로에서는 지나갔던 곳을 표시하더라도 빛이 없어서 식별할 수 없다.

- 알고리즘
  - 현 위치에서 한 방향을 선택하고, 벽에 오른손을 댄다. 그리고 출구가 나올 때까지 계속 오른손을 벽에서 떼지 않고 걸어간다. 이 방법은 실타래나 특별한 표시가 필요 없이 항상 출구를 찾게 해준다.



1.6 가짜 동전 찾기 (1)

- 문제
  - 아주 많은 동전 더미 속에 1개의 가짜 동전이 섞여 있다. 가짜 동전은 매우 정교하게 만들어져 눈으로 식별할 수 없다. 그러나 가짜 동전의 무게는 정상적인 동전보다 약간 가볍다. 가짜 동전 찾기 문제는 양팔 저울만 사용하여 가짜 동전을 찾아내는 것인데, 가능한 한 저울에 동전을 다는 횟수를 줄여야 한다.

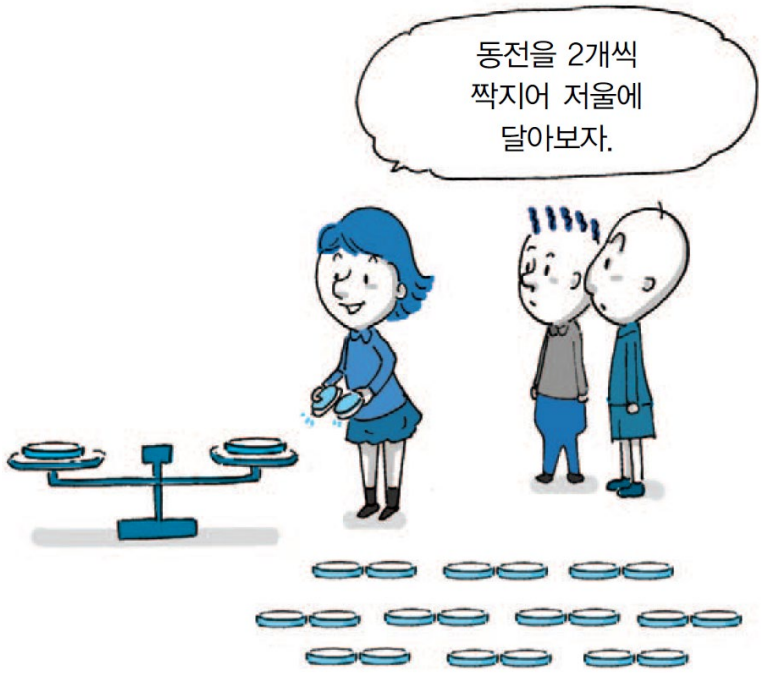
- 아이디어
  - 양팔 저울은 저울 양쪽이 같은 무게인지 아닌지 만을 판별해준다. 따라서 동전 더미를 분할하여, 저울에 올려서 한쪽으로 기우는지 아닌지를 알아내어 가짜 동전을 가려내야한다.

1.6 가짜 동전 찾기 (2)

- 철수의 생각
  - 임의의 동전 1개를 저울 왼편에 올리고, 나머지 동전을 하나씩 오른편에 올려서 가짜 동전을 찾아보자.
  - 철수의 제안은 운이 좋으면 1번만에 가짜 동전을 찾는다. 왼편에 미리 올려놓은 동전이 가짜이거나, 아니면 오른편에 처음으로 올려놓은 동전이 가짜인 경우이다.
  - 최악의 경우 마지막에 가짜 동전을 저울 오른편에 올려놓게 된다. 만일 총 동전 수가 n이라면 (n-1)번 저울을 재야 한다.

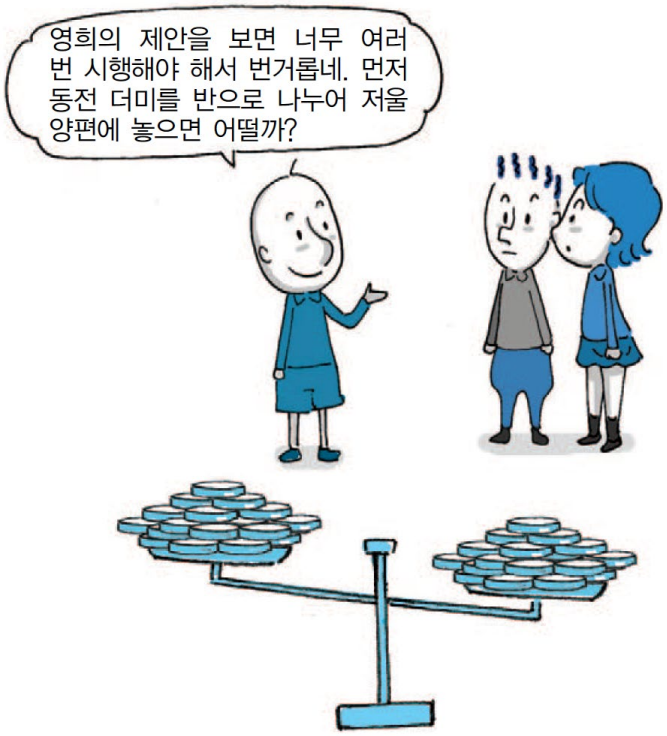
1.6 가짜 동전 찾기 (3)

- 영희의 생각
  - 동전을 2개씩 짝을 지어, n/2 짝을 각각 저울에 달아서 가짜 동전을 찾아보자.
  - 영희의 제안도 운이 좋으면 첫 번째 짝을 저울에 올렸을 때 가벼운 쪽의 동전이 가짜임을 알 수 있다.
  - 최악의 경우 가장 마지막 짝을 잴 때 불균형이 일어나고, 최대 $\lfloor n/2 \rfloor$번의 저울을 재야 한다.
  - 영희의 알고리즘은 철수의 알고리즘에 비해 최악의 경우 저울에 다는 횟수가 거의 1/2로 줄었다.



1.6 가짜 동전 찾기 (4)

- 광수의 생각
  - 영희의 제안을 보면 짝의 수가 너무 많다. 먼저 전체 동전 더미를 반으로 나누어 저울 양편에 놓으면 어떨까?
  - 광수의 제안은 한 번에 어느 쪽에 가짜 동전이 있는지 알 수 있다. 그 다음에는 가벼운 쪽을 다시 반씩 나누어 저울에 잰다. 남은 동전 수는 계속 1/2로 줄어들게 되고, 나중에 2개가 남았을 때 가짜 동전을 가려낼 수 있다.
  - 광수의 알고리즘은 운이 좋을 때가 없다. 왜냐하면 마지막에 가서야 가짜 동전을 찾기 때문이다.



1.6 가짜 동전 찾기 (5)

- 알고리즘
  - 동전 더미를 반으로 나누어 저울에 달고, 가벼운 쪽의 더미를 계속 반으로 나누어 저울에 단다. 분할된 더미의 동전 수가 1개씩이면, 마지막으로 저울을 달아 가벼운 쪽의 동전이 가짜임을 찾아낸다.
  - 광수 알고리즘은 동전이 1,024개 있을 때 몇 번 저울에 달아야 할까? 먼저 512개씩 양쪽에 올려놓고 저울을 재고, 다음은 256개씩, 128개씩, 64개씩, 32개씩, 16개씩, 8개씩, 4개씩, 2개씩, 마지막에는 1개씩 올려서 저울을 잰다. 총 10번이고 ($\log_2 1{,}024 = 10$), 일반적인 n에 대해서는 $\log_2 n$번이다.
  - 최악의 경우 철수는 1,023번 저울에 달아야 하고, 영희는 512번 달아야 하는데, 광수는 10번이면 가짜 동전을 찾는다. 이 차이는 n이 커지면 더욱 더 커진다.

1.7 독이 든 술단지 (1)

- 문제
  - 어느날 이웃 나라의 스파이가 임금님의 창고에 들어가서 술단지 하나에 독을 넣고 나오다가 붙잡혔다. 스파이는 눈으로 확인할 수 없는 독을 사용하였고, 하나의 단지에만 독을 넣었다고 실토하고는 숨을 거두었다.
  - 스파이가 사용한 독의 특징은 독이 든 단지의 술을 아주 조금만 맛보아도 술을 맛본 사람은 정확히 일주일 후에 죽는다는 것이다. 임금님은 독이 든 술단지를 반드시 일주일 만에 찾아내라고 신하들에게 명하였다.
  - 문제는 희생되는 신하의 수를 줄이는 것이다. (& 알고리즘에 사용되는 신하 수를 최소화)

1.7 독이 든 술단지 (2)

- 술단지의 수가 2개일 때
  - 한 명의 신하가 하나의 술단지의 술을 맛보고 일주일 후 살아 있으면 맛보지 않은 단지에 독이 있는 것이고, 죽는다면 맛본 술단지에 독이 들어있는 것이다.
- 술단지의 수가 4개일 때
  - 철수의 제안: 두 명의 신하가 각각 한 단지씩 맛보게 하자.
  - 문제점: 두 사람이 맛보지 않은 나머지 2개의 단지 중 하나에 독이 들어 있으면, 일주일 후 두 사람은 살아 있을 것이고, 나머지 2개의 단지 중 어느 하나에 독이 들어 있는지를 알 수 없다.

1.7 독이 든 술단지 (3)

- 술단지의 수가 4개일 때
  - 광수의 제안: 4개의 단지를 2개의 그룹으로 나누어, 각 그룹에 한 사람씩 할당한다. 이 경우는 맨 처음 고려했던 술단지의 수가 2인 경우가 2개가 생긴 셈이다. 과연 답을 찾을 수 있을까?
  - 문제점: 일주일 후에 만일 두 사람 다 살아 있으면, 어떤 단지가 독이 들어 있단 말인가? 각 그룹에서 맛을 안 본 단지가 하나씩 있으므로, 이 2개의 단지 중 하나에 독이 들어 있는 것이다. 그러나 어느 단지인지를 알려면 또 일주일이 필요하므로 광수의 제안 역시 실패이다.

1.7 독이 든 술단지 (4)

- 핵심 아이디어
  - 한 신하가 하나의 단지의 술만 맛 볼 필요가 없다. 즉, 광수의 제안에서 총 4개의 단지 중에 시음하지 않은 2개의 단지가 있는데, 이 2개의 단지 중에 하나를 두 신하에게 동시에 맛보게 하는 것이다. 이렇게 하면 다음과 같이 4가지의 결과가 생긴다. 두 신하를 각각 A와 B라고 하자.

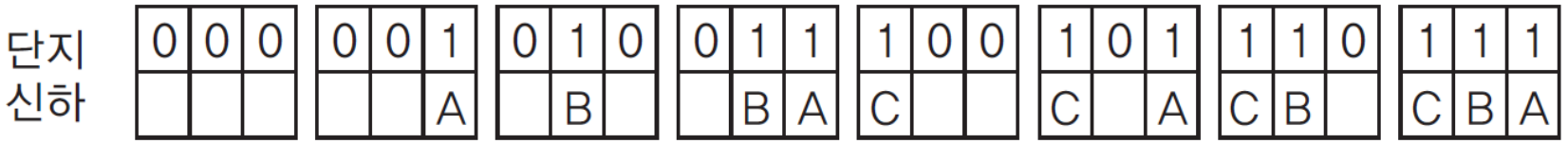
  - 아무도 시음하지 않은 단지에 독이 있으면, 일주일 후 두 신하 모두 다 살아있다.
  - A가 혼자 시음한 단지에 독이 있으면, 일주일 후 A만 죽는다.
  - B가 혼자 시음한 단지에 독이 있으면, 일주일 후 B만 죽는다.
  - A와 B 둘 다 시음한 단지에 독이 있으면, 일주일 후 둘 다 죽는다.

1.7 독이 든 술단지 (5)

- 알고리즘 (n=4)
  - 단지 수가 많을 때에는 어떻게 해야 할까? 각 단지에 2진수를 0부터 부여하고, 각 신하가 술 맛을 보면 1 안 보면 0으로 하여, 다음과 같이 단지와 신하를 짝지어 보는 것이다.
  - 단지의 번호를 00부터 11의 2진수로 표기한 후 A는 오른쪽 비트가 1인 단지 01과 단지 11을 맛보고, B는 왼쪽 비트가 1인 단지 10과 단지 11을 맛본다.

1.7 독이 든 술단지 (6)

- 알고리즘 (n=8)
  - 술 단지를 2진수로 000부터 111로 표기하고, 각 자릿수마다 한 명의 신하를 할당한다. 다음과 같이 첫 번째 비트는 C, 두 번째 비트는 B, 마지막 비트는 A에 할당한다.



- 알고리즘
  - 술단지 개수를 2진수로 표기하여 각 비트당 1명의 신하를 할당하고, 각 신하는 자신이 할당 받은 비트가 1인 단지들을 모두 맛보게 한다.
  - 술단지 개수가 n인 경우 $\log_2 n$명의 신하가 필요하고, 일주일 후 최소 희생자 수는 0명, 최대 희생자 수는 $\log_2 n$명이다.