[CSE3081(2반)] 알고리즘 설계와 분석

2020학년도 2학기

강의자료

(2020.09.01 화요일)

서강대학교 공과대학 컴퓨터공학과
임 인 성 교수

- 본 강의에서 제작하여 제공하는 PDF 파일, 동영상, 그리고 예제 코드 등의 강의 자료의 저작권은 특별히 명기되어 있지 않은 한 서강대학교에 있습니다.
- 본인의 학습 목적 외에 공개된 장소에 올리거나 타인에게 배포하는 등의 행위를 금합니다. 협조 부탁합니다.

과목 소개

- **과목명:** 알고리즘 설계와 분석 (Design and Analysis of Algorithms)
- **과목 번호 및 분반:** CSE3081 (2반)
- **강의 시간/강의실:** 화목 10:30-11:45 / 온라인 강의
- **담당 교수:** 임 인 성 (AS-905, ihm@sogang.ac.kr)
- **담당 조교:** 윤 제 형 (AS-914, dudrms5975@sogang.ac.kr)

- **과목 중요 공지 관련**
  - **담당 교수와 조교가 수시로 공지 사항을 사이버 캠퍼스 공지사항에 게시할 예정임.**

- **수업 목표**
  - 2학년 1학기까지 습득한 **C/C++ 프로그래밍 능력과 자료구조 이론의 이해 및 구현 능력**을 바탕으로, **컴퓨터를 통한 추상적인 문제의 효과적인 해결에 기초가 되는 알고리즘의 설계/분석/구현 기법**을 익힘을 목표로 한다.
  - 이를 위하여,
    ① **문제 분석/풀이 기법 도출/풀이 기법의 비용 분석** ← 주요 내용
    ② C/C++ 언어를 통한 자신의 풀이 기법의 최적의 구현, 그리고
    ③ 자신의 소프트웨어 구현물에 대한 분석 과정에 대하여 익히도록 한다.

- **컴퓨터공학 전공자로서 본 수업을 통하여 습득하려는 능력**
  - Algorithm에 대한 정의와 complexity와 computability 개념에 대한 이해
  - Asymptotic analysis of time/space complexity 개념에 대한 이해
    - Worst-case versus average-case
    - Recursion 개념의 활용
  - (Mathematical induction을 통한) algorithm correctness 증명 능력
  - Dynamic set의 표현 및 응용 능력
    - Priority queue 구조
    - Disjoint set 구조
  - Divide-and-conquer 기법에 대한 이해 및 응용 능력
  - Sorting 방법에 대한 이해 및 구현 기법 습득
  - Dynamic programming 기법에 대한 이해 및 응용 능력
  - Greedy approach에 대한 이해 및 응용 능력
  - Graph 구조 표현 기법 구현 및 관련 알고리즘 응용 능력
  - Intractable Problem과 근사 알고리즘에 대한 이해 **(희망 사항)**

➢ **이러한 능력을 습득하기 위하여 상당한 시간에 걸쳐 반복적인 노력이 필요함**

- **수강대상: 컴퓨터공학과 2학년 2학기생 기준 과목**
  - **[CSE3080 자료구조]**를 수강한 학부생 중, **학번이 짝수**인 학부생 ← 학기수는 상관 없음.
  - 학번이 **홀수인 학생 중** 중간 고사 기간이 지난 후, 타 과목과 시간이 겹친다는 사실을 증명하는 공식 서류를 조교에게 제출하는 학생 ← **미제출시 최종 성적 100점 만점에 상당한 점수 감점.**

- **선수과목 미이수 학생 수강 관련:**
  - 수년 전에 한 컴퓨터공학과 학생이 선수 과목을 미이수한 과목의 인정 문제에 관하여 규정상에 문제가 있었음.
  - 현재 **입학년도/학과/프로그램이 매우 다양하게 다른 학생들이 수강을 하고 있고 따라서 전공과 관련하여 적용되고 있는 학칙이 서로 다름**.
  - 선수 과목 규정과 관련하여 담당 교수가 일일이 내용을 책임 있게 확인하기가 매우 힘든 상황이니, 이와 관련해서는 학과/프로그램 규정을 확인하고 해당 직원 선생님에게 문의하기 바람.
  - **규정상 인정이 안될 경우 교수가 임의로 규정을 변경할 수 없으며, 이에 대해서는 본인의 책임 하에 해결하기 바람**.

- **평가 방법**
  - 중간고사 및 기말고사 각 30% 총 60%
  - 프로그래밍 숙제 30%
  - 기타 과제물 및 참여도 10%

➢ 상기 비율은 대략적인 수치이며, 수업 진행 상황에 따라 약간의 변화가 있을 수 있음.

➢ 중간 고사와 기말 고사는 대면 시험이 예정되어 있음.

➢ 프로그래밍 숙제에 대해서는 철저한 카피 체크가 있을 예정이며, **복사한 사람 및 복사 당한 사람 모두**에게 **0점 처리 및 기타 불이익**을 가할 예정임.

예상 진도 (2020학년도 2학기)

W1: 9/1, 9/3 Introduction to Design and Analysis of Algorithm	**W2:** 9/8, 9/10 Priority Queues and Applications **(Review)**	**W3:** 9/15, 9/17 Practice of Complexity Analysis through Example Problems	**W4:** 9/22, 9/24 Divide-and-Conquer Techniques
W5: 9/29, **10/1**추석 Divide-and-Conquer Techniques and Sorting	**W6:** 10/6, 10/8 Dynamic Programming Techniques	**W7:** 10/13, 10/15 Dynamic Programming and Applications	**W8:** 10/19 ~ 10/23 정확한 시험 시간은 추후 공지 예정 **MIDTERM**
W9: 10/27, 10/29 Greedy Techniques	**W10:** 11/3, 11/5 Greedy Techniques and Scheduling Algorithms	**W11:** 11/10, 11/12 Introduction to Graph Data Structures	**W12:** 11/17, 11/19 Graph Algorithms and Applications I
W13: 11/24, 11/26 Graph Algorithms and Applications II	**W14:** 12/1, 12/3 Introduction to NP-Completeness	**W15:** 12/8, 12/10 *Intractable Problems and Approximation Algorithms* **또는 진도 보충**	**W16:** 12/14 – 12/18 정확한 시험 시간은 추후 공지 예정 **FINAL**

- **강의 순서는 수업의 효율 제고를 위하여 적절히 변경할 예정임.**

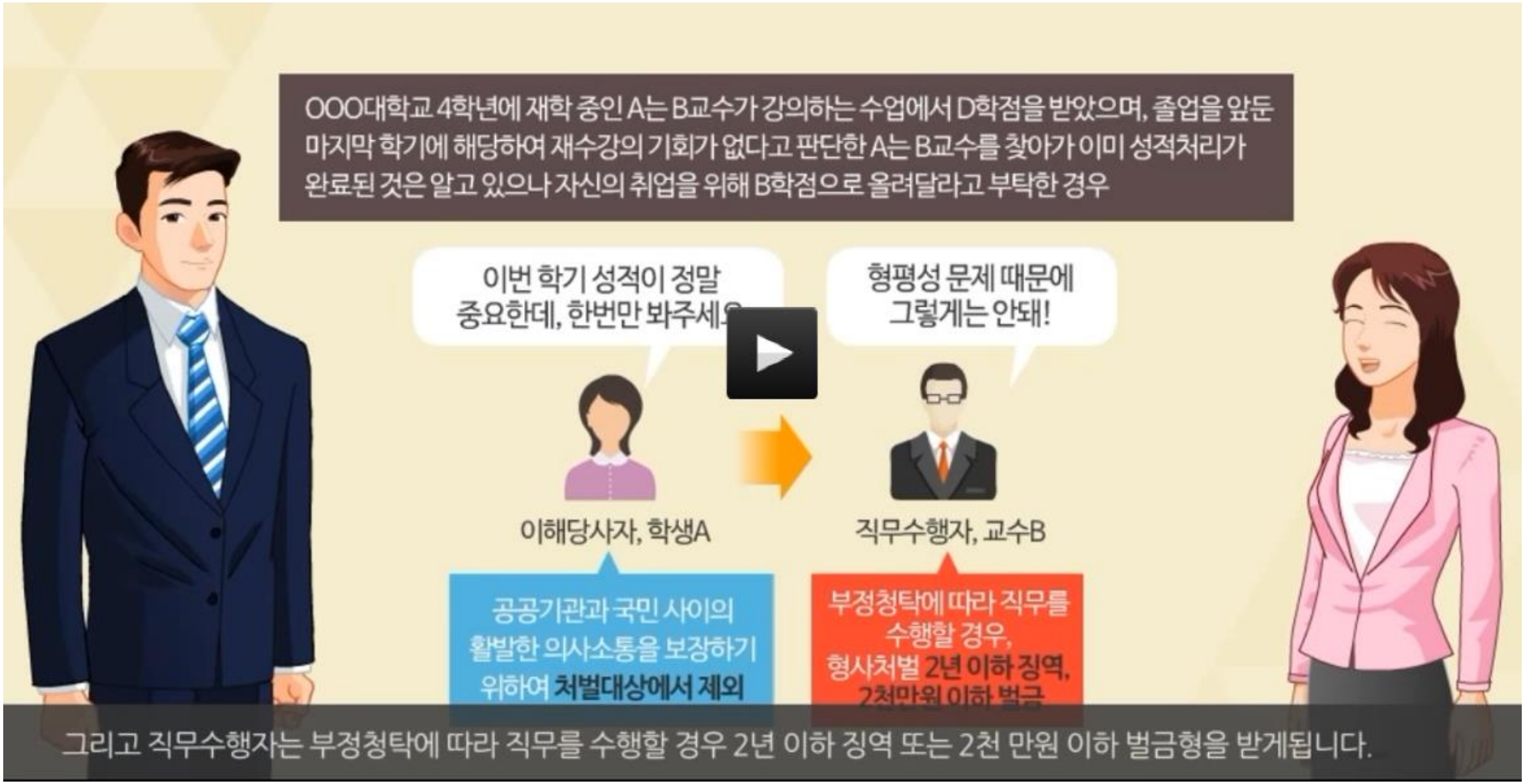
교재 및 참고 도서

적절한 교재를 선택하여 관련된 내용을 자세하게 읽기를 권장함

- R. Neapolitan, ***Foundations of Algorithms* (5th ed.),** Jones & Bartlett, 2015.
- T. Cormen et al., ***Introduction to Algorithms*** (3rd ed.), The MIT Press, 2009.
- T. Roughgarden, ***Algorithms Illuminated,*** *Part 1~3*, Soundlikeyourself Publishing, 2018.
- J. Kleinberg and E. Tardos, ***Algorithm Design***, Addison Wesley, 2005.
- R. Sedgewick and K. Wayne, **Algorithms** (4th ed.), Addison-Wesley, 2011.
- S. Dasgupta et al., ***Algorithms***, McGraw-Hill Education, 2006.
- S. Baase and A. Van Gelder, *Computer Algorithms: Introduction to Design and Analysis*, Addison Wesley, 2000.
- E. Horowitz et al., *Fundamentals of Data Structures in C*, Computer Science Press, 1993.
- S. Skiena, *The Algorithm Design Manual* (2nd ed.), Springer, 2008.
- A. Aho, J. Hopcroft, and J. Ullman, *Design and Analysis of Algorithms*, Addison-Wesley, 1974.
- M. Weiss, *Data Structure and Algorithm Analysis in C* (2nd ed.), Pearson, 1997.
- A. Levitin, *Introduction to the Design and Analysis of Algorithms*, Addison Wesley, 2003.
- A. Aho, J. Hopcroft, and J. Ullman, *Data Structures and Algorithms*, Addison-Wesley, 1983.
- E. Horowitz, S. Sahni and S. Rajasekaran, *Computer Algorithms/C++*, Computer Science Press, 1997.
- R. Sedgewick, *Algorithms in C: Parts 1-4* (3rd ed), Addison-Wesley, 1998.
- R. Sedgewick, *Algorithms in C: Parts 5* (3rd ed), Addison-Wesley, 2002.

강의 자료 순서

- [주제 1] Introduction to Algorithms and Complexity
- [주제 2] Heap-based Priority Queues and Heap Sort (Review)
- [주제 3] Divide-and-Conquer Techniques and Sorting Techniques
- [주제 4] Dynamic Programming
- [주제 5] Greedy Methods
- [주제 6] Graph Algorithms
- [주제 7] Intractable Problems and Approximation Algorithms
- [주제 8] More on Priority Queues and Hashing


OOO대학교 4학년에 재학 중인 A는 B교수가 강의하는 수업에서 D학점을 받았으며, 졸업을 앞둔 마지막 학기에 해당하여 재수강의 기회가 없다고 판단한 A는 B교수를 찾아가 이미 성적처리가 완료된 것은 알고 있으나 자신의 취업을 위해 B학점으로 올려달라고 부탁한 경우
이번 학기 성적이 정말 중요한데, 한번만 봐주세요
형평성 문제 때문에 그렇게는 안돼!
이해당사자, 학생A
직무수행자, 교수B
공공기관과 국민 사이의 활발한 의사소통을 보장하기 위하여 처벌대상에서 제외
부정청탁에 따라 직무를 수행할 경우, 형사처벌 2년 이하 징역,
그리고 직무수행자는 부정청탁에 따라 직무를 수행할 경우 2년 이하 징역 또는 2천 만원 이하 벌금형을 받게됩니다.

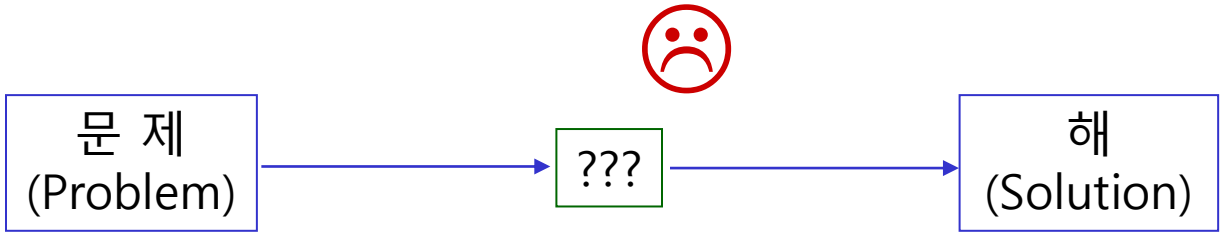
[주제 1]
Introduction to Algorithms and Complexity

Computational Thinking

From Wikipedia

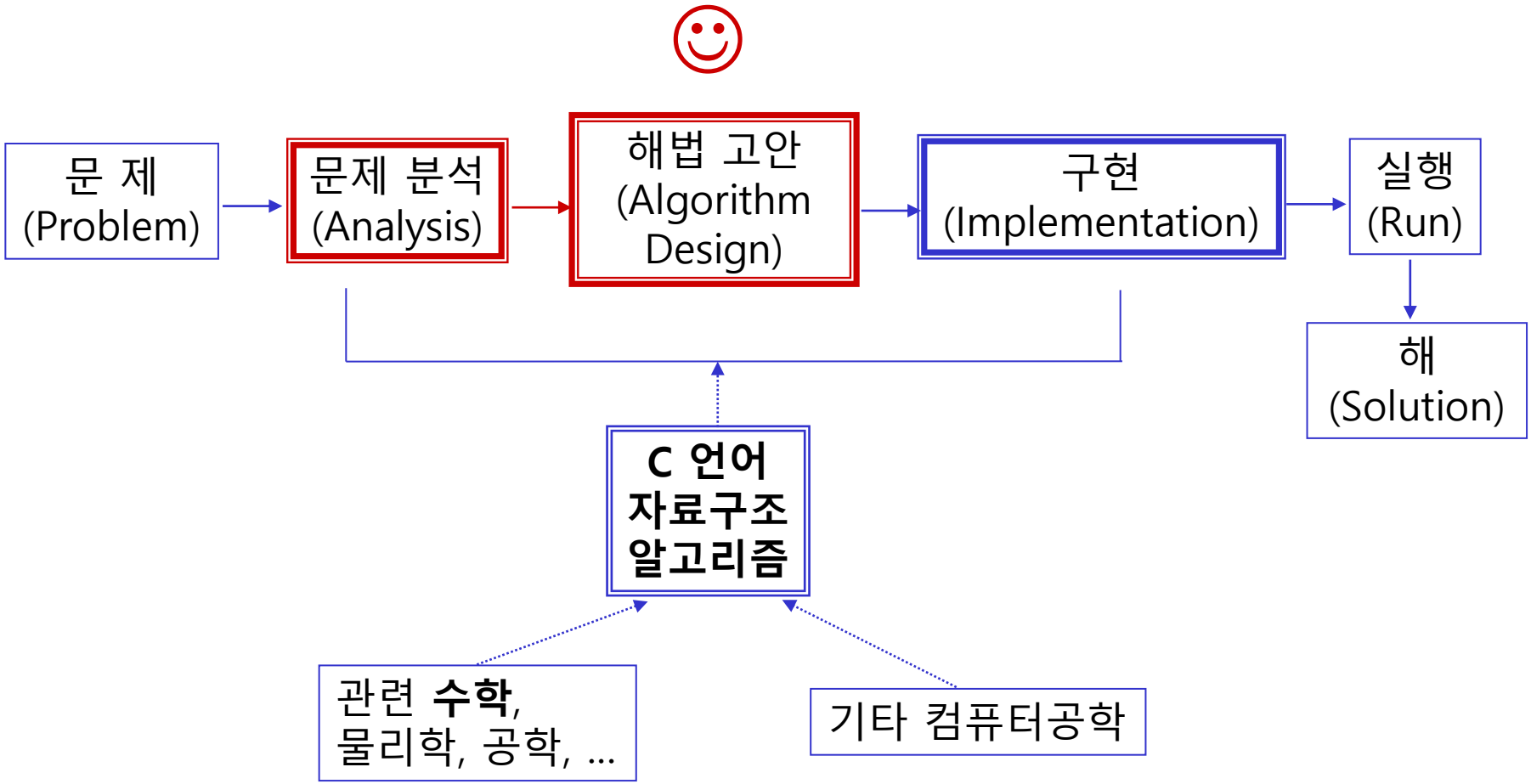
- Definition of computational thinking
  - **The thought processes involved in (i) formulating a problem and (ii) expressing its solutions in such a way that a computer --human or machine-- can effectively carry out.**
    ① Problem formulation (abstraction)
    ② Solution expression (automation)
    ③ Solution execution & evaluation (analyses)
- Characteristics of computational thinking
  - Formulating problems in a way that enables us to use a computer and other tools to help solve them
  - Logically organizing and analyzing data → Data structure
  - Representing data though abstractions such as models and simulations → Data Structure
  - Automating solutions through algorithmic thinking (a series of ordered steps) → Algorithm
  - Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources → time and space complexity
  - Generalizing and transferring the problem solving process to a wide variety of problems

Problem Solving in Computer Science and Engineering



가상 현실, 문서작성,
홈뱅킹, 인터넷 신문,
문서 번역, 회로 설계,
유전자 분석, 무인 자동차,
온라인 게임, 비디오 편집,
자료 검색, 영화 제작,
음성 인식, 가상 수술,
건축 설계, 기상 예측,
주가 예측, 인공 지능, 대용량
과학 계산,
...

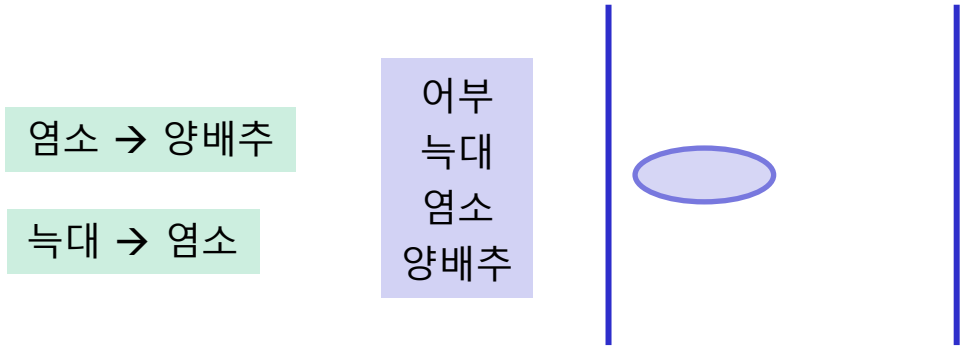
Problem Solving Pipeline



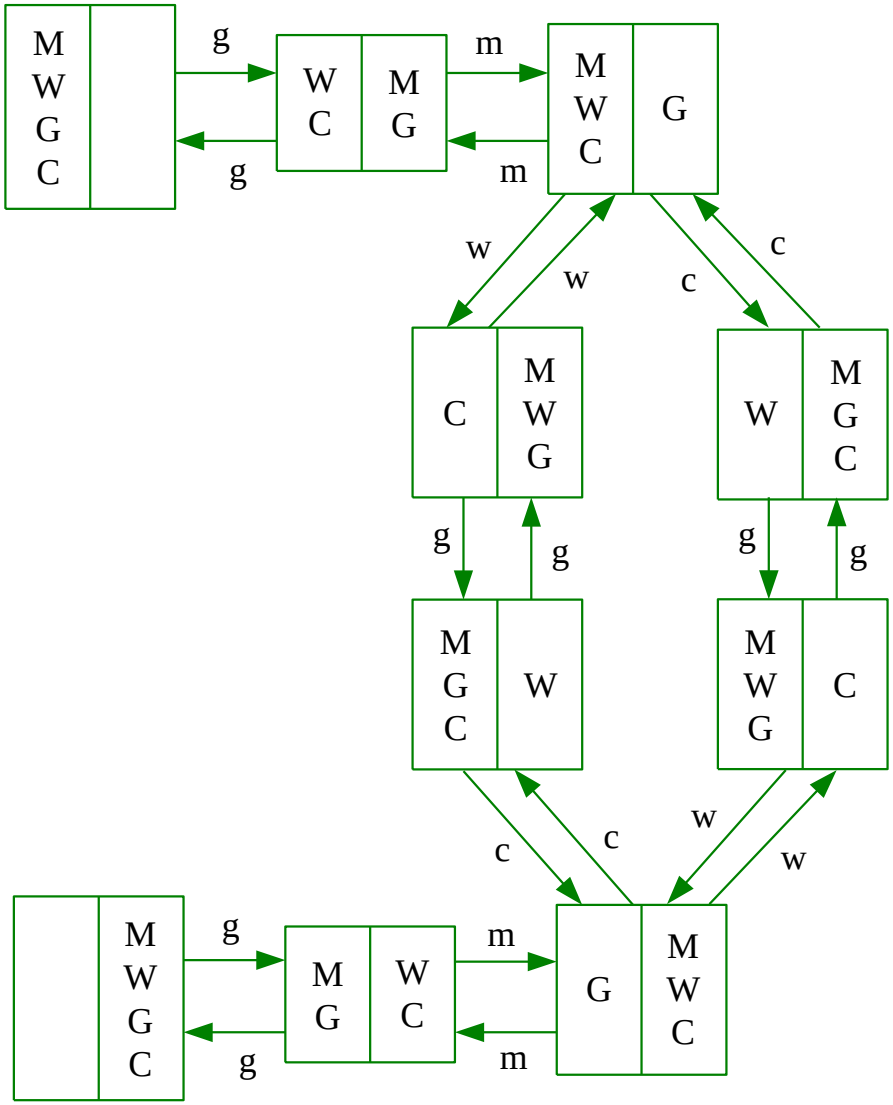
문제: 도강 문제

- 한 어부(M)가 늑대(W), 염소(G), 양배추(C)를 강 한 쪽에서 다른 쪽으로 옮기려 한다. 어부가 배를 타고 강을 건널 때 어부 자신 외에 늑대, 염소, 양배추 중 하나만 배에 가지고 갈 수가 있는데, 문제는 어부가 늑대를 싣고 가는 동안, 염소가 양배추를 같은 쪽에 남겨두면 염소가 양배추를 먹어버리게 되고, 양배추를 싣고 갈 때 늑대와 염소를 같은 쪽에 남겨둘 경우 늑대가 염소를 잡아 먹게 된다.

과연 어떻게 하면 어부가 가장 적은 회수로 강을 건너면서 세 가지를 모두 안전하게 옮길 수 있을까?



문제 분석


MWGC
g
g
WC
MG
m
m
MWC
G
w
w
c
c
C
MWG
W
MGC
g
g
g
g
MGC
W
MWG
C
c
c
w
w
MWGC
g
g
MG
WC
m
m
G
MWC

해법 고안

- Graph, search, and so on →

  Which data structures and algorithms?

- Cost, time, space, and so on →

  What complexities?

- [연습] 이 문제에 대한 알고리즘과 시간/공간 복잡도를 컴퓨터학의 용어를 써서 기술한다면,

  ???

* **무슨 말인지 전혀 모르겠으면 [43-080 자료구조]를 재수강한 후 이 과목을 들을 것!**

구현

어떻게 하면 좋은 구현 결과를 얻을 수 있는가?
- 동일한 프로세서 상에서 더 빠르게
- 적은 메모리만 사용하게
- 안정적이게
- ...

✓ ***Programming is an art!***

- 어떻게 하면 주어진 알고리즘을 가장 효과적으로 구현을 할 수 있을까?
  - 어떻게 하면 C/C++를 사용하여 주어진 알고리즘을 가장 최적으로 구현할 수 있을까?
    - 원시 코드 레벨의 측면
    - 어셈블러 레벨의 측면
    - 시스템 레벨의 측면
    - 기타

✓ 과연 내가 http://acm.uva.es/problemset/에 있는 문제들을 스스로 "문제 분석->해법 고안->구현" 과정을 통하여 효과적으로 해결할 수 있을까???

Programming Challenges by S. Skiena and M. Revilla, Springer, 2003.

- 구현 예 1

```
#define MATDIM2 8192
double MatA[MATDIM2][MATDIM2], MatB[MATDIM2][MATDIM2], MatC[MATDIM2][MATDIM2];

void MinStride_1(double c[][MATDIM2], double a[][MATDIM2],
                                      double b[][MATDIM2]) {
   for (int i = 0; i < MATDIM2; i++)
      for (int j = 0; j < MATDIM2; j++)
         c[i][j] = c[i][j] + a[i][j] * b[i][j];
}
```

0.265968초

3.4GHz Intel Core i7 CPU

```
void MinStride_2(double c[][MATDIM2], double a[][MATDIM2],
                                      double b[][MATDIM2]) {
   for (int j = 0; j < MATDIM2; j++)
      for (int i = 0; i < MATDIM2; i++)
         c[i][j] = c[i][j] + a[i][j] * b[i][j];
}
```

4.862961초

- 구현 예 2

```
char *Implementation_1(int c) {
  switch(c) {
     case 0: return "EQ"; case 1: return "NE"; case 2: return "CS";
     case 3: return "CC"; case 4: return "MI"; case 5: return "PL";
     case 6: return "VS"; case 7: return "VC"; case 8: return "HI";
     case 9: return "LS"; case 10: return "GE"; case 11: return "LT";
     case 12: return "GT"; case 13: return "LE"; case 14: return "";
     default: return 0;
  }
}
```

어떻게 하면 "양질의 코드"를 작성할 수 있을까?

```
char *Implementation_2(int c) {
   if ((unsigned) c >= 15) return 0;
   return "EQ\0NE\0CS\0CC\0MI\0PL\0VS\0VC\0HI\0LS\0GE\0LT\0GT\0LE\0\0"
          + 3*c;
}
```

Data Structure → Algorithm → Theory of Computation

- **어떻게 하면 주어진 복잡한 문제를 이진수 형태의 낮은 수준의 명령어만 이해하는 '단순한' 컴퓨터 상에서 효율적으로 해결할 수 있을까?**
  1. **[Data Structure]** 주어진 추상적인 문제를 어떠한 자료 구조를 사용하여 컴퓨터의 구조에 최적화된 형태로 표현할 수 있을까?
  2. **[Algorithm]** 주어진 추상적인 문제를 어떠한 알고리즘을 사용하여 컴퓨터를 사용하여 가장 효율적으로 해결할 수 있을까?
  3. **[Complexity]** 과연 컴퓨터가 주어진 문제를 효율적으로 해결할 수 있을까 ?
  4. **[Computability]** 과연 컴퓨터가 세상의 모든 문제를 해결할 수 있을까?

✓ 이 과목에서는 [CSE3080 자료구조] 과목에 이어, 1번과 2번을 집중적으로 살펴보고, 3번 문제에 대하여 어느 정도 살펴볼 예정임. 4번 문제는 [CSE3085 자동장치이론] 과목에서 다룸.