5. 정보(문제)의 표현

2010 데이터로 표현하는 세상 요약본
고려대학교 김현철 교수
hkim64@gmail.com

5-1 정보의 표현

우리는 이전 장에서, 데이터가 어떻게 효율적으로 분해되어 인코딩 될 수 있는지를 살펴보았으며, 또한 문제해결에 필요한 정보가 어떻게 분해되어 구조화 될 수 있는지를 살펴보았다. 두 가지 모두 단위정보들로의 '분해'와 그리고 그들의 '재구조' 혹은 '재조합'을 통하여 효율성을 높이고 정보의 가치를 높이고자 하는 것이었다.

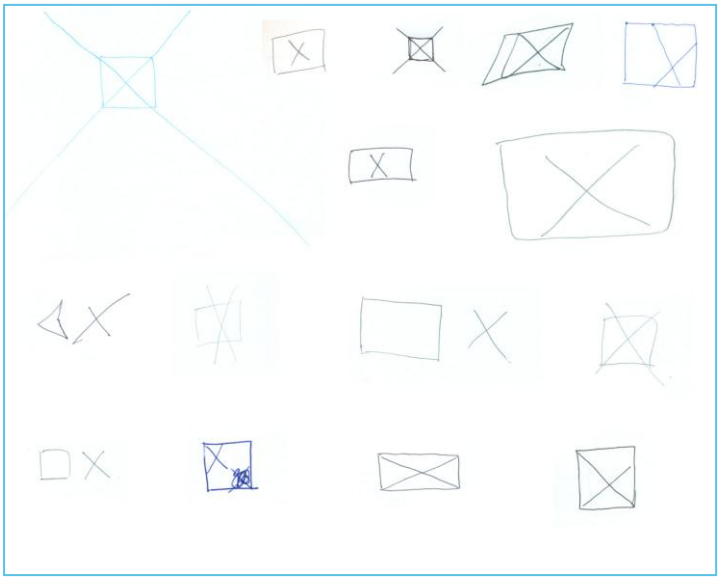
이번 단원에서는 그러한 '분해와 재조합'의 방식을 조금 더 확장하여 문제해결(Problem solving)에 적용하는 것에 대하여 알아보도록 하자.

머리 속 생각의 표현

내 머릿속에 있는 생각 (즉, 주관적 의미를 포함한 정보)을 명시적인(explicit) 형태로 밖으로 꺼내어 표현하는 것은 쉬운 일은 아니다. 만약, 정확하게 표현하지 않으면 그것을 받아들이는 다른 사람은 그것을 다시 자신의 주관적인 관점과 기준으로 변형하여 그 정보를 받아들이기 때문이다. 여기서의 핵심은 주관적인 요소가 없이 어떻게 정확하게 정보를 표현할 것인가 이다.

- 명확성 (비모호성)
- 완전성
- 간략성

가장 일반적으로 사용하는 explicit form은 '언어'라고 할 수 있다. 자연어는 우리의 생각을 정확하게 표현하고 있는가?
사례〉 사각형을 그리고 X 모양으로 두 개의 선을 그리시오



Fact는 X자 모양의 직선 두 개와 사각형이고, 나머지는 "상상력". 자신의 개인적 경험, 상식, 순간적 감정에 의하여 만들어진 주관적 fact이다.

백문불여일견(百聞不如一見).
보는 것이 믿는 것이다. (seeing is believing)
인간에게는 눈에 보이는 것이 명확한 정보를 제공해 준다. 만약, 눈에 보이지 않으면 자기 나름대로의 주관에 의하여 (주로 과거의 경험, 내재된 암묵적 지식) 상상으로 그 보이지 않는 부분을 정의하려고 한다. 과거의 경험, 내재된 암묵 지식은 사람마다 다르기 때문에 결국 보이지 않는 부분 때문에 정보는 각 개인에게 다른 의미로 전달될 가능성이 있다.
소설은 읽는 사람마다 다른 의미와 감동, 무한한 상상력을 준다. 완전하지 않은 그림은, 즉 빠진 부분이 있는 부분이 있다면 그것으로 인하여 다른 의미와 오해를 줄 수가 있을 것이다.

우리는 '언어' 이외에 또 어떤 다른 표현 방법을 사용하고 있는가.
사진, 동영상, 그림, 음악, 무용, 동작 등.
전달하려고 하는 메시지가 무엇인지에 따라서 더 효율적인 방법이 있을 수 있다.

우리는 위와 같은 방법 이외에도 명확성과 완전성을 위하여 여러 방법을 사용하기도 한다. 그 중에 가장 대표적인 적이 테이블이나 다이어그램과 같은 것이다. 예) 지하철 노선도, 지도에서 빠른 길 찾기.
이러한 방법으로 우리의 막연하고 애매모호 했던 생각을 정확하고 명확하게 표현해 낼 수 있게 도와 준다. 남의 머리 속에 있는 생각을 끄집어 낼 때에도 사용된다.

〈예시〉

✓ Turnstile problem

■ Consider a software-controlled turnstile situated at the entrance to a zoo. When the turnstile is fed a coin, it unlocks, allowing a visitor to push through the turnstile and enter the zoo. Once an unlocked turnstile has rotated enough to allow one entry, the turnstile locks again, to prevent another person entering without payment.

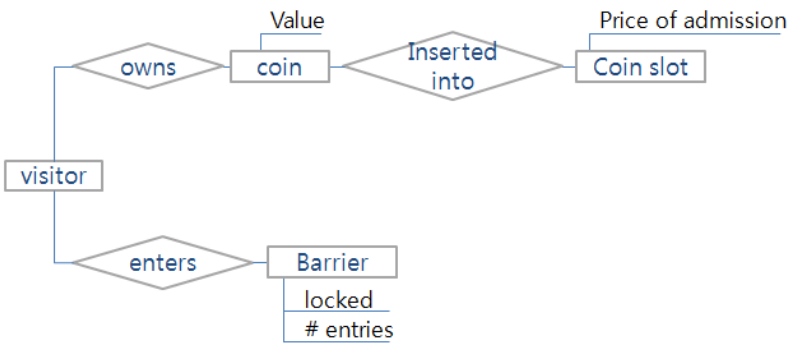
✓ Library problem

■ The library needs to track its texts and other materials, its loan records, and information about its patrons. Popular items are placed on *reserve*, meaning that their loan periods are shorter than those of other books and materials, and that the penalty for returning them late is higher than the late penalty for returning unreserved items.

ER(Entity-Relationship) Diagram

구성요소

✓ Entity
✓ Relationship
✓ Attribute

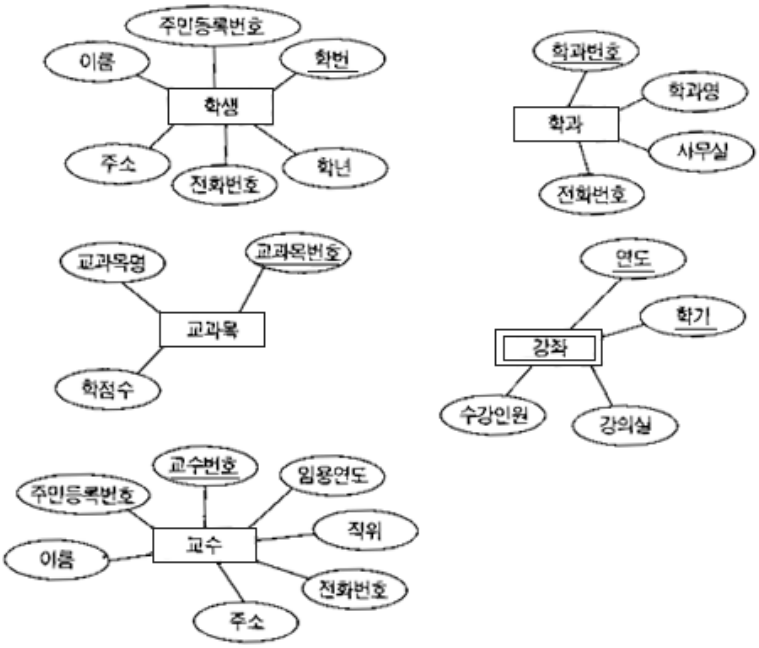


*** 연습문제 ***

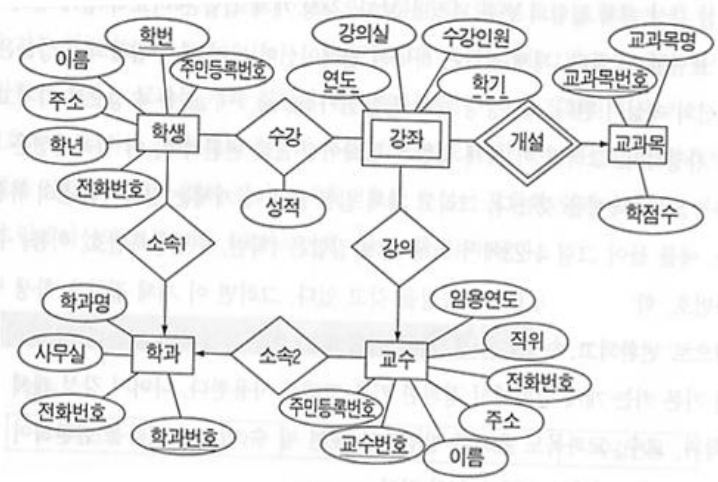
〈 요구사항 〉
학생과 교수로 구성된 대학의 구조를 이용한 ER-Diagram을 작성한다.

〈 조건 〉
- 학생에게는 고유의 학번이 부여되며 주민등록번호, 이름, 주소, 전화번호, 학년의 정보를 갖는다.
- 교수는 교수의 고유번호인 교수번호가 부여되며 주민등록번호, 이름, 주소, 전화번호, 직위, 임용연도 등의 정보를 갖는다.
- 학생과 교수는 학과에 소속되며 학과는 학과번호, 학과명, 사무실, 전호번호 등이 있다.
- 각 교과목은 교과번호, 교과목명, 학점수를 가지며 한 학기에 하나의 강좌만 개설될 수 있다.
- 개설된 강좌는 강의실과 한 명의 교수가 배정된다. 학생은 한 학기에 하나 이상의 개설된 강좌를 수강할 수 있고 성적이 부여된다.

〈 개체 집합들 〉



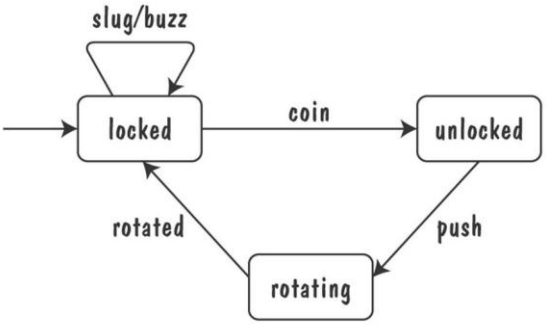
〈 완성된 개체-관계 다이어그램 〉

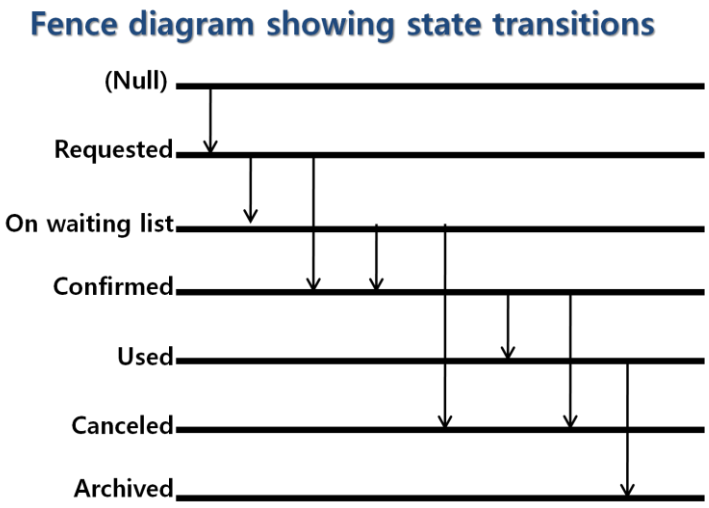


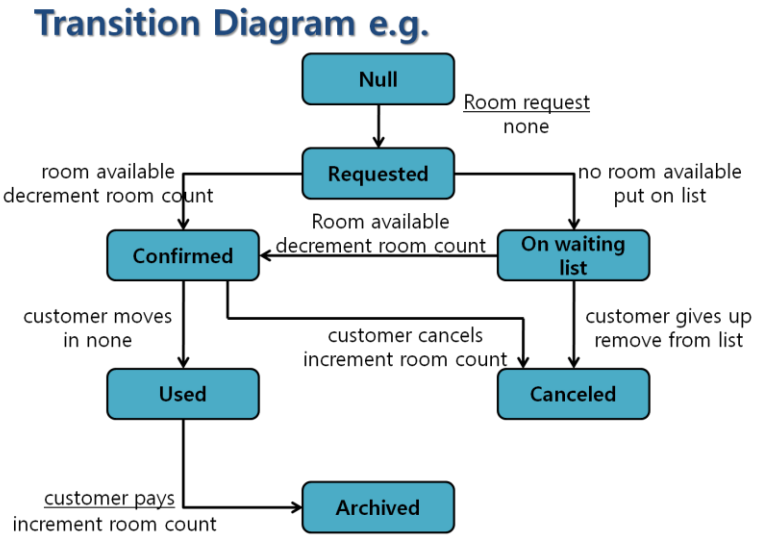
State machine

구성요소

- ✓ node (state)
- ✓ Edge (transition)



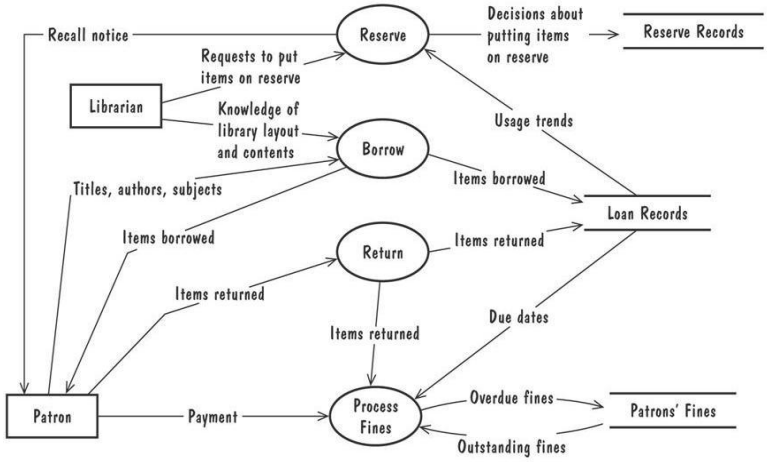


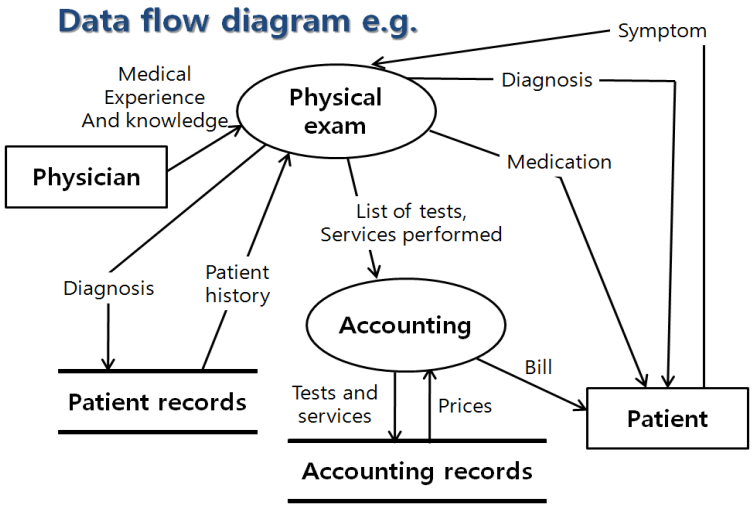


(DFD) Data Flow Diagram

구성요소

- ✓ 프로세스
- ✓ Data flow
- ✓ Data store
- ✓ Actor





Decision Table

	Rule1	Rule2	Rule3	Rule4	Rule5
High standardized exam scores	T	F	F	F	F
High grades	-	T	F	F	F
Outside activities	-	-	T	F	F
Good recommendations	-	-	-	T	F
Send rejection letter			X	X	X
Send admission forms	X	X			

5-2 문제 해결 및 문제의 표현

이전까지의 수업에서 우리는

- ✓ Text, image, audio, numerical data를 computer에 "효율적"으로 저장하는 방법에 대하여 배웠다. 즉, 정해진 bit 수의 범위 내에서 원 데이터를 서로 구별할 수 있도록 저장하고, 또 최대한 많은 것을 표현 할 수 있도록, 그리고 가장 적은 메모리의 양을 차지할 수 있도록 (즉, 가장 적은 양의 bit수를 가지고) 표현하는 방법에 대하여 학습하였다.
- ✓ Digital data로 변환된 데이터에 대하여 어떤 특정한 목적의 computing을 하여 원하는 data를 만들어 내고자 할 때, 똑같은 원 세상데이터라 할지라도 다른 형태의 digital data로 표현될 수 있음을 학습하였다. 이 경우에는 computing에서 필요한 요소 정보가 무엇인지를 먼저 생각을 하고 원 세상 데이터에서 그러한 요소 정보를 끄집어 내어 데이터로 표현하여야 한다.
- ✓ 요소정보를 포함하는 단위정보들을 어떠한 형태로 구조화 함으로써, 의미적 정보/지식을 만들어 낼 수가 있다.
- ✓ 문제 해결 지식은, 단위 지식/정보/task로 잘게 쪼개고 그리고 그들을 조직/구조화 하여 표현한다.

이제 우리는 Data Representation에서 벗어나 Problem Representation에 대하여 학습하여 보자.

"Problem"은 우리가 해결해야 할 세상의 "문제"이다.
What is a problem? Any situation in which a gap is perceived to exists between what is and what should be.

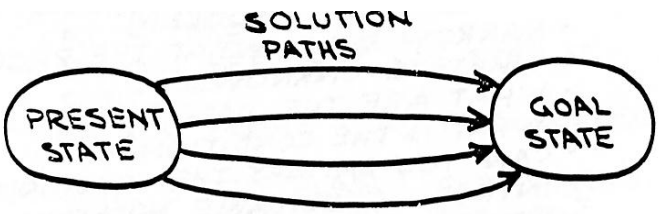
Frequently this GAP is dynamic with either the starting or goal states or both changing.

Problems differ in degree of structure

- ✓ Well-structured
  - ■ All information is available
  - ■ Use a standard procedure to reach a solution (algorithm)
- ✓ Semi-structured
  - ■ Can only partially define the problem due to uncertainty about the starting state, the goal state or the gap
  - ■ Requires some combination of a standard procedure and a creative response (heuristics)
- ✓ Ill-structured
  - ■ Little or no information on best way to develop solution
  - ■ Information to solve problem must be generated during problem solving process
  - ■ Solution is improvised using custom-made approach (creative problem solving techniques)

Our problem can be viewed as a system which we are attempting to analyze and develop an equivalent to

Can also view a problem using a model (an abstract representation of reality)



5-3 문제해결방법

문제 해결에서, 문제의 재정의, 즉, 재 구조화를 통하여 문제 해결을 용이하게 할 수 있다.
알고리즘.

출처 : 중학교 정보, 이원규/김현철 외, 미래엔컬쳐, 2010

문제 해결 방법 찾기

문제 해결 방법 찾기는 주어진 문제를 해결하기 위한 여러 가지 방법을 찾아내는 단계이다. 이 단계에서 다른 사람들과 다양한 문제 해결 방법들을 함께 생각해 보고, 찾아보면 더욱 효율적이다.

주어진 문제를 해결하기 위한 방법을 찾기 위해서 '입력', '처리', '출력'으로 나누어 생각해 보면 보다 편리하다.

입력 문제를 해결하기 위해 가장 먼저 넣어야 하는 자료를 의미한다.

처리 문제를 해결하기 위한 과정을 적어 보는 것을 의미한다.

문제를 해결하는 과정에는 문제의 주어진 조건에 따라 순서대로 문제 해결을 처리하는 과정이 나올 수도 있고, 조건에 따라 여러 가지 경우로 처리되는 과정이 생길 수도 있다. 이때, 다양하게 나올 수 있는 문제 해결 방법의 경우를 모두 다 적어 보는 것이 좋다.

출력 문제가 해결되었을 때, 나올 수 있는 결과를 의미한다.

즉, 문제 해결 방법을 찾기 위해 입력, 처리, 출력값을 '입력-처리-출력(IPO : Input-Process-Output)' 표로 정리할 수 있다.

표 II-3 입력-처리-출력(IPO) 표

입력	처리	출력
입력값 예) 7, 8	다양한 문제 해결 과정 7×8	결과값 56

문제의 표현

문제를 쉽게 이해하거나 분석하기 위해서는 문제에서 주어진 사실을 이용하여 자신이 이해할 수 있는 방법으로 표현해 보는 것이 좋다. 즉, 문제에 포함된 복잡한 개념을 이해하기 쉽고 친숙한 개념으로 바꿀 수 있도록 표현을 다르게 하여 나타내는 것이다.

문제를 표현하는 방법에는 글로 써서 단순화하거나 표나 그림 같은 시각적인 도구를 이용하여 나타낼 수 있다.

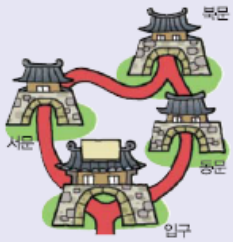
글로 써 보기 문제를 자신이 이해할 수 있는 문장으로 정리해 보면 복잡해 보이는 것이 좀 더 쉽게 이해된다.

표(Table)로 만들어 보기 복잡한 판단을 해야 할 때, 관련된 정보를 일목요연하게 정리하여 표로 만들면 핵심적인 내용을 쉽게 파악할 수 있다.

그림으로 그려 보기 복잡한 문제를 간단하게 표현할 때, 도형이나 그래프 등의 그림으로 그리는 방법이다.

다음은 〈○○산 등산 경로 찾기〉를 '문제 정의 표'를 이용하여 이해하고 분석한 예이다.

예 ○○산의 입구에서 출발하여 '북문'까지 가는 방법에는 두 가지가 있다. 입구에서 '동문'까지는 약 15분이 소요되고, 입구에서 '서문'까지는 약 40분이 걸린다. '동문'에서 '북문'까지는 약 25분, '서문'에서 '북문'까지는 약 20분이 걸린다고 한다.



그렇다면 입구에서 북문까지 가장 빠르게 가는 방법은 무엇일까?

표 II-2 문제 정의 표: ○○산 등산 경로 찾기

아는 사실	요구 사항
입구에서 북문까지 가는 두 가지 방법(시간).	입구에서 북문까지 가장 빠르게 가는 방법.

〈○○산 등산 경로 찾기〉의 예는 입구에서 북문까지 등반할 때, 어떤 등산로를 선택해야 더 빨리 올라갈 수 있는지를 알아보는 것이다.

위의 예를 글, 표, 그림 등을 이용하여 표현하면 다음과 같다.

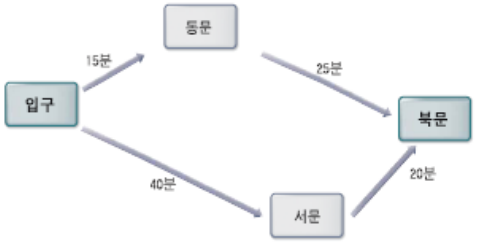
[글로 써 보기]

〈방법 1〉 ○○산 입구 → 동문(15분) → 북문(25분)
〈방법 2〉 ○○산 입구 → 서문(40분) → 북문(20분)

[표(Table)로 만들어 보기]

방법 1			방법 2		
출발지	도착지	시간	출발지	도착지	시간
입 구	동 문	15분	입 구	서 문	40분
동 문	북 문	25분	서 문	북 문	20분

[그림으로 그려 보기]



문제를 정량적으로 표현하는 방법

- ✓ 구조도 (divide and conquer): 문제를 작게 나눈다.
- ✓ Flow-chart, Pseudo code : 문제를 procedural 한 지식으로 해결한다.
- ✓ State-space representation : 문제를 state들의 연결로 정의한다.
- ✓ Problem-reduction representation :문제를 recursive한 abstract모델로 정의한다.

Two views of problem representation

1. State-space representation
   : 초기 상태에서 목표 상태까지 이르는 모든 상태(state)들을 정의하고, 각 state들간을 이동할 수 있게 하는 연산자(operator)를 정의하여 state-space를 만든다. 그 상태공간(state-space)에서 초기상태에서 목표상태를 얻을 때까지 연산자를 거듭 가하는 탐색작업(search process)에 의하여 문제를 해결한다. 다시 말하면, 상태공간에서 초기 상태에서 목표상태에 도달할 수 있는 일련의 연산자를 찾으면 그것이 문제의 해이다.
2. Problem-reduction representation
   주어진 문제를 작고 쉬운 부분문제(sub-problem)들로 나누어 해결한다.

State-space representation

STATE-SPACE REPRESENTATION

(S, I, G, O, R)

S - THE SET OF STATES, SNAPSHOTS OF THE PROBLEM AT VARIOUS STAGES OF SOLUTION

$I \subset S$ - ONE OR MORE INITIAL STATES

$G \subset S$ - ONE OR MORE GOAL OR FINAL STATES

O - SET OF OPERATORS

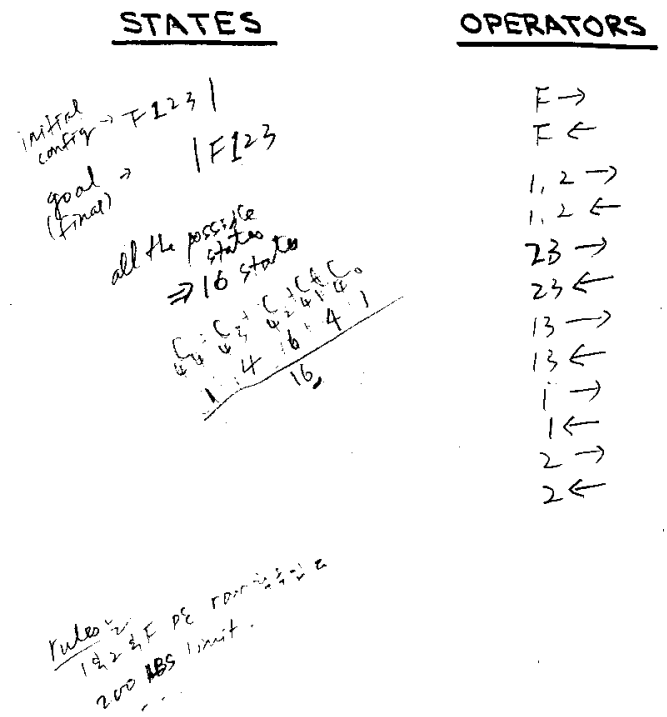
R - SET OF RULES OR CONTROL STRATEGY

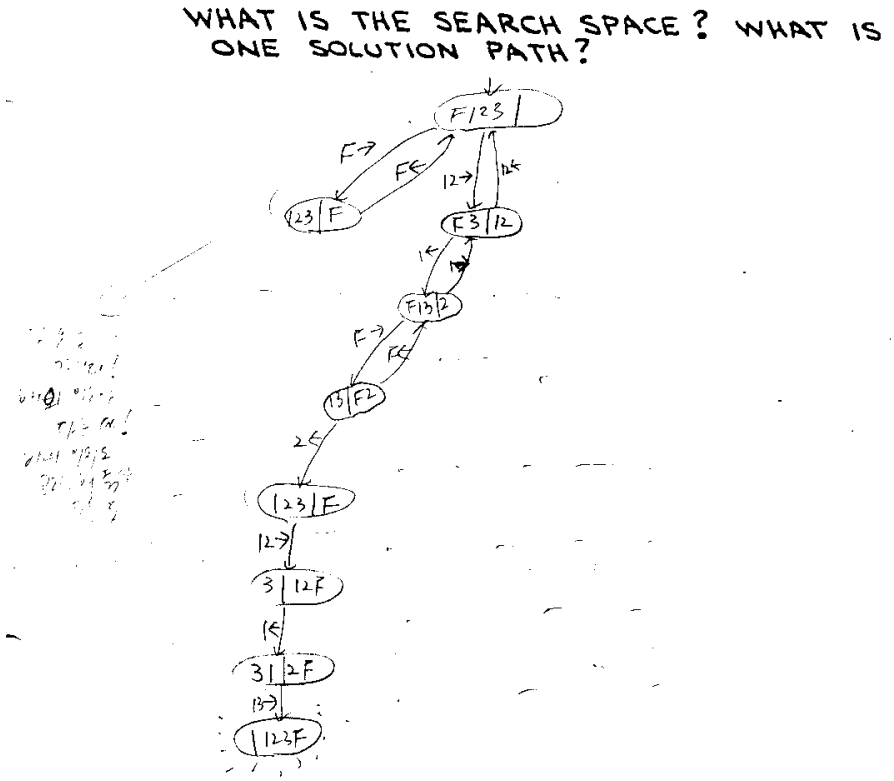
SEARCH SPACE
- TOTAL SPACE WHICH EXISTS
- ALL POSSIBLE STATES
- TYPICAL SPACES:
  CHESS 10^{120}
  CHECKERS 10^{40}

<<예: 아버지와 세 아들 문제, 식인종과 선교사 문제, Tic-Tae-Toe문제, 8-puzzle문제 >>

PROBLEM: THREE SONS AND THEIR FATHER WISH TO CROSS A RIVER. THE FATHER WEIGHS 200 LBS AND EACH SON WEIGHS 100. THEY HAVE A BOAT WHICH CAN CARRY ONLY 200 LBS. ONLY SONS 1 & 2 and Father CAN ROW THE BOAT. WHAT ARE THE STEPS REQUIRED FOR THEM TO CROSS THE RIVER?

STATES OPERATORS


STATES
OPERATORS
initial config → F123|
goal (final) → |F123
all the possible states
⇒ 16 states
F →
F ←
1, 2 →
1, 2 ←
23 →
23 ←
13 →
13 ←
1 →
1 ←
2 →
2 ←
Rules:
200 ABS limit.


WHAT IS THE SEARCH SPACE? WHAT IS ONE SOLUTION PATH?
F123|
F→
F←
12→
12←
123|F
F3|12
1←
1→
F13|2
F→
F←
13|F2
2←
123|F
12→
3|12F
1←
3|2F
13→
|123F

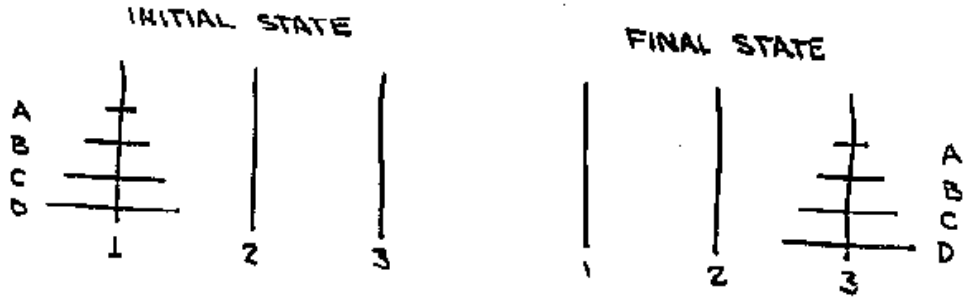
[정리]
State-Space Representation 에서 다음의 단계를 생각해보자.

먼저, 구성요소인 "state"을 어떻게 표현 할 것인가.
그것을 정의하면, => 모든 가능한 state를 포함하고 있는 state space를 구 할 수 있고 그 size도 알 수 있다. 예를 들어 tic-tac-toe게임과 같은 경우는 9칸 각각의 칸에 {empty, x, o}중의 하나가 오므로 3^9가지의 경우가 된다. 8-puzzle의 경우는 9!(=362,880)의 경우. 하지만 이들 중에서 결코 발생하지 않는 state들이 있다. 즉, tic-tac-toe에서는 모든 칸이o으로 채워진 state는 결코 발생하지 않는다. State-space의 size를 모두 다 완벽하게 구할 필요는 없다. Space의 구조와 알고리즘에 따라서 필요 없는 곳으로 가지 않는다는 것만 보장하고, 필요한 곳은 다 거쳐간다는 것만 보장한다면.
두 번째로는, "state space"을 어떻게 표현할 것 인가. 예를 들어, tic-tac-toe에서 "tree"혹은 "directed acyclic graph"을 사용하여 표현하면 최대 9!로 줄 것이다.
State space가 결정되면
세 번째로, "state space search" 전략을 잡아야 한다. 즉, 8-puzzle의 경우 9!=326,880의 가능한 경우가 있고, chess의 경우는 10^120 possible game paths가 있다. 따라서 state space에서 goal state를 효율적으로 찾는 방법이 필요하다.

Problem Reduction Representation

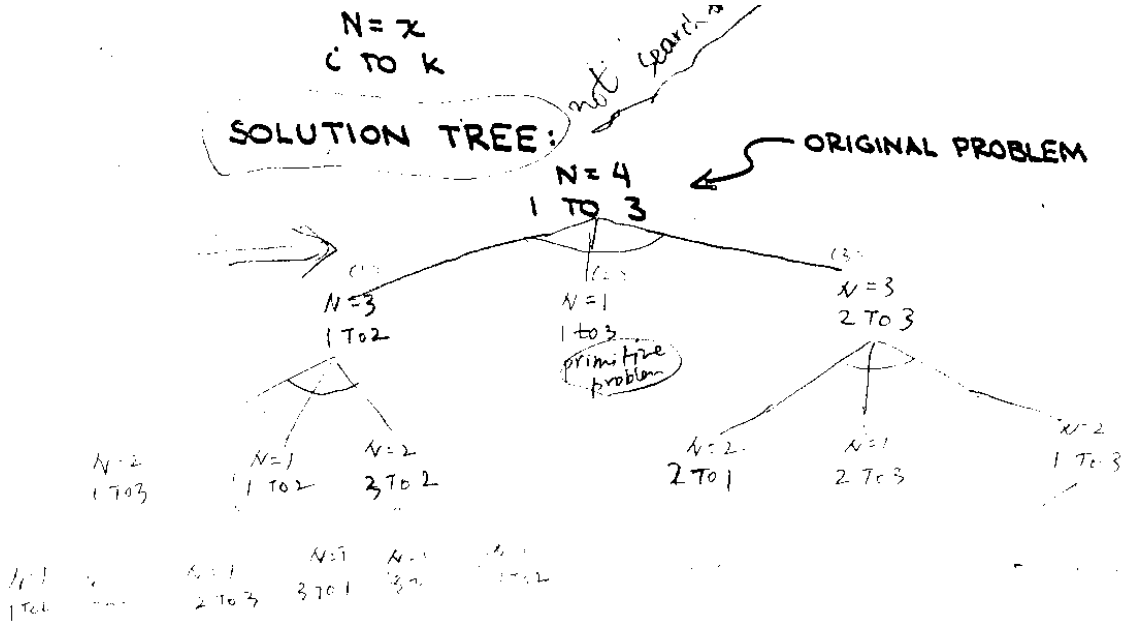
문제를 해결할 수 있는 작은 sub-problems들로 분해하여 해결하려고 하는 방법이다. 문제 자체를 recursive하게 정의할 수 있는 추상적인 형태로 변환하는 것이 관건이다. 다시 설명하면, 문제를 재귀적(recursive) 형태로 변환하면 간단하게 해결할 수 있다.
예를 가지고 설명해보도록 하자

[예: 하노이의 타워]
Hanoi Tower
There are 4 disks(A,B,C,D) of graduated sizes initially stacked on peg1 of three pegs with A, the smallest, on top and D, the largest, at the bottom. The disks are to be transferred to peg 3 observing the following rules:
(1) Only one disk can be moved at a time &
(2) No disk can be placed on top of a smaller disk.



일반화된 방법으로 설명한다면,
Given 3 peg i, j and k, the problem of moving a stack of size n>1 from peg I to peg k can be replaced by the three problems:
(1) move n-1 disks from i to j
(2) move 1 disk from i to k
(3) move n-1 disks from j to k

recursive하게 이 문제를 해결하여 보시오.
Ex) n=4일 때,



3peg이고 disk가 n개일 때 몇 번의 move가 필요한가?

1) 기둥1의 원판A를 기둥3으로 이동한다.
2) 기둥1의 원판B를 기둥2로 이동한다.
3) 기둥3의 원판A를 기둥2로 이동한다.
4) 기둥1의 원판C를 기둥3으로 이동한다.
5) 기둥2의 원판A를 기둥1로 이동한다.
6) 기둥2의 원판B를 기둥3으로 이동한다.
7) 기둥1의 원판A를 기둥3으로 이동한다.

원판 수	2	3	4	5	6	7	8	9	…	30	31	32
이동 회수	3	7	15	31	63	127	255	511	…	1,073,741,823	2,147,483,647	4,294,967,295
시간	3”	7”	15”	31”	1’ 3”	2’ 7”	3’ 15”	6’ 31’	…	약34년	약68년	약136년

※ n개의 원판 수를 이동하는 최소 수 : 2^n-1

.끝.