



1

알고리즘 개념

어떤 작업을 수행하기 위해 입력을 받아 원하는
출력을 만들어내는 과정을 기술한 것

- ◆ 어떤 일을 수행할 수 있는 일련의 명령어 또는 규칙의 집합
- ◆ 알고리즘을 설계하기 위해서는 해야 할 작업을 명확하게 명시해야 함
- ◆ 문제해결이나 처리 과정에서의 순서를 단계적으로 서술

◇ 예) 요리

- 어떤 종류의 음식을 만드는 요리법이 요리 알고리즘
- 레시피가 요리의 질을 좌우하는 결정적 요소
- 순서에 입각하여 단계적으로 서술하는 것이 알고리즘

◇ 예) 게임

- 게임에서 알고리즘의 적용은 매우 중요
- 게임에서 목표지점까지 도달하기 위한 과정을 차례로 표현
- 출발에서 목표까지의 효율적인 접근 방법과 과정

3 생활 속의 알고리즘 예

- ▶ 아침부터 하루 일을 구상하고 스케줄을 짜며 계획하는 일
- ▶ 전자기레인지 등 전자제품의 사용 설명서에 나타난 사용 방법
- ▶ 바둑이나 게임을 잘하기 위한 방법론
- ▶ 어떤 목적지로 이동할 때 효율적인 교통 수단 환승 방법

4 문제해결을 위한 알고리즘

- ▶ 두뇌 또는 컴퓨터로 문제를 해결함에 관계없이 적용
- ▶ 문제의 복잡성에 따라 컴퓨터 프로그램을 통해 문제를 해결
- ▶ 알고리즘은 컴퓨터 프로그램을 작성하는 바탕

4 문제해결을 위한 알고리즘

▶ 알고리즘으로 어떤 문제를 푸는가?

- 자동차 네비게이션

두 지점간의 최단 경로나 최단 시간이 걸리는 경로

→ 최단 경로 알고리즘

- 현금자동인출기(ATM)

여러 지역의 수많은 ATM의 잔고 유지하며 자금 운용

→ 스케줄링 알고리즘 등

- 인간 게놈 프로젝트

방대한 DNA 데이터간의 관계 파악

→ 게놈 서열간의 유사성 파악하는 알고리즘 등

4 문제해결을 위한 알고리즘

▶ 알고리즘으로 어떤 문제를 푸는가?

- 인터넷 검색

인터넷에는 수조 페이지 이상의 데이터 존재

➔ 최대한 빨리 정확한 검색 결과 제공하는 알고리즘

- 반도체 설계

하나의 반도체 칩에는 수천수만 개의 게이트 존재

➔ 게이트의 배치와 논리적 연결 구조 설계 알고리즘

- 제조업

제조 공정에서 작업의 선후 관계가 존재

➔ 위상 정렬 알고리즘

5 문제를 알고리즘으로 작성하는 과정

(1) 문제 분석

- ▶ 주어진 문제에 대한 논리적 분석을 통하여 핵심 사항들을 분석

(2) 데이터 수집과 표현

- ▶ 문제해결과 관련된 정보들을 수집하며 데이터를 적절한 형태로 표현

5 문제를 알고리즘으로 작성하는 과정

(3) 분해

- ▶ 복잡한 문제를 보다 쉽게 다룰 수 있도록 여러 개의 작은 부분들로 쪼개어 분해

(4) 패턴인식

- ▶ 문제 내에서 공통적인 유사성이나 규칙을 찾아냄

5 문제를 알고리즘으로 작성하는 과정

(5) 추상화

- ▶ 문제에서 필요 없는 부분들을 걸러내고 복잡한 문제나 아이디어를 단순화

(6) 알고리즘

- ▶ 문제에 대한 단계적인 해결책, 설명, 지시 사항들을 설계

5 문제를 알고리즘으로 작성하는 과정

(7) 평가

- ▶ 알고리즘의 정확성, 해답의 적절성, 효율성 등을 최종 점검
- ▶ 평가 완료 후에는 알고리즘을 기반으로 코딩함

5 문제를 알고리즘으로 작성하는 과정

▶ 예) 모바일 앱 만들기

- 복잡한 앱을 만들려면 여러 가지 상황의 고려가 필요
- 여러 개의 작은 일들로 분해하여 명확하고 효율적으로 해결
- 여러 사람들이 분해된 일을 나누어 병렬로 수행 가능

※ 고려사항

만들려는 앱의 종류, 핵심 기능, 앱의 유형,
사용할 대상, 필요한 SW, 테스트 방법 등

5 문제를 알고리즘으로 작성하는 과정

- ▶ 예) 학생 10명의 시험점수를 입력받아 최고점을 출력하는 알고리즘을 작성하시오.
- 입력 : 10개의 점수
 - 출력 : 입력된 10개의 점수 중 최댓값

5 문제를 알고리즘으로 작성하는 과정

▶ 말로 표현된 알고리즘

- ① 첫번째 점수의 숫자를 읽고 머릿속에 기억해 둔다.
- ② 다음 점수의 숫자를 읽고 그 숫자를 머릿속의 숫자와 비교한다.
- ③ 비교 후 큰 숫자를 머릿속에 기억해 둔다.
- ④ 다음에 읽을 점수가 남아있으면 ②로 간다.
- ⑤ 머릿속에 기억된 숫자가 최대 점수이다.

5 문제를 알고리즘으로 작성하는 과정

- ◆ 의사 코드로 표현된 알고리즘
배열 A에 학생 점수가 10개 있다고 가정

```
max = A[0];
```

```
for i = 1 to 9
```

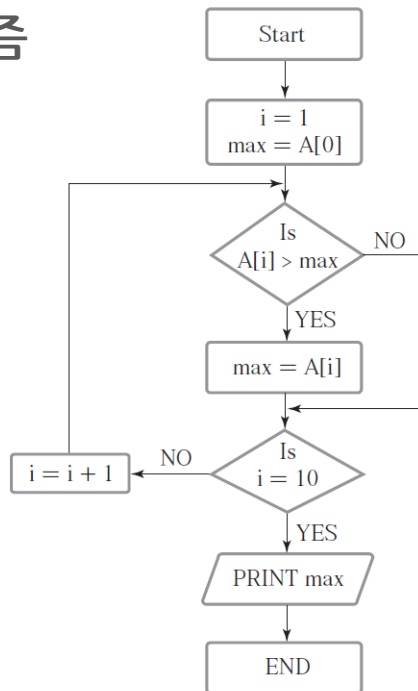
```
    if (A[i] > max)
```

```
        max = A[i];
```

```
return max;
```


5 문제를 알고리즘으로 작성하는 과정

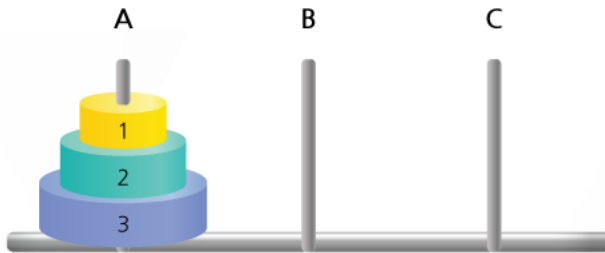
▶ 순서도로 표현된 알고리즘



5 문제를 알고리즘으로 작성하는 과정

예) 하노이 탑(Tower of Hanoi) 문제

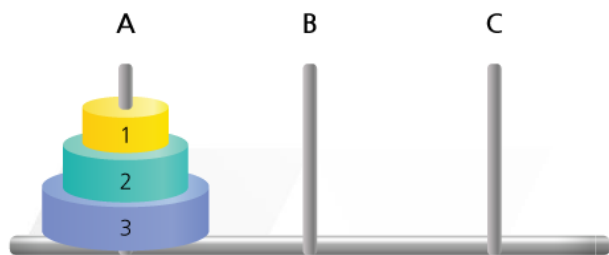
- 크기가 서로 다른 n 개의 원반이 크기 순서대로 막대 A에 끼워져 있음
- 다른 2개의 막대기(B, C)를 이용하여 이 n 개의 원판을 다른 막대기 C로 옮기는 문제



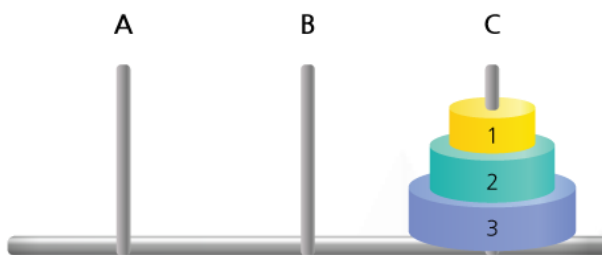
5 문제를 알고리즘으로 작성하는 과정

하노이 탑 문제의 규칙

▶ A 막대에 쌓여있는 3개의 원반을 C 막대로 옮기는 문제



시작 상태의 하노이 탑



이동 완료

5 문제를 알고리즘으로 작성하는 과정

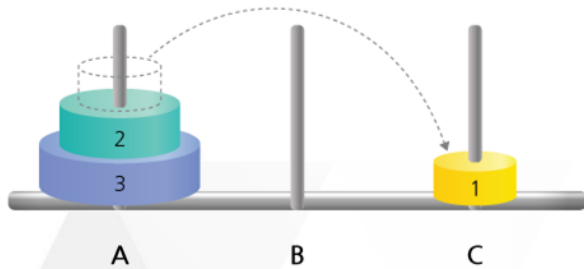
하노이 탑 문제의 규칙

- ▶ 원반이 한 막대에서 다른 막대로 한 번에 하나씩만 이동
- ▶ 작은 원반 위에 큰 원반이 절대 놓일 수 없음
- ▶ 중간 막대를 임시적으로 이용 가능하나 위의 규칙 지켜야 함

5 문제를 알고리즘으로 작성하는 과정

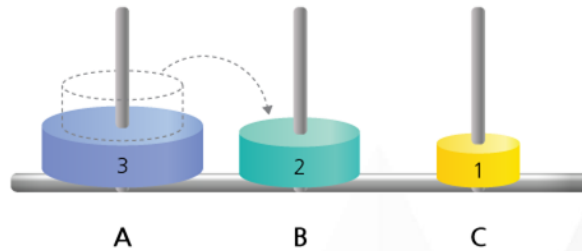
원반 3개의 경우

[1단계]



[원반 1을 A에서 C로 이동]

[2단계]

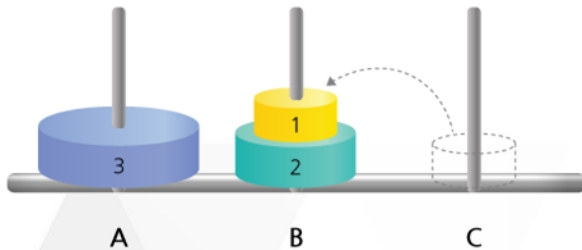


[원반 2를 A에서 B로 이동]

5 문제를 알고리즘으로 작성하는 과정

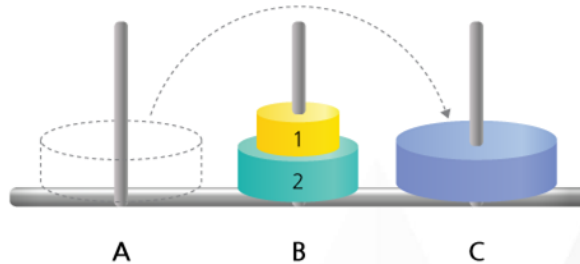
원반 3개의 경우

[3단계]



[원반 1을 C에서 B로 이동]

[4단계]

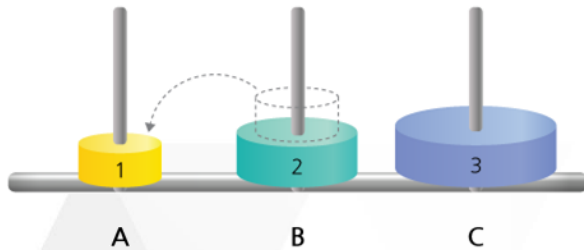


[원반 3을 A에서 C로 이동]

5 문제를 알고리즘으로 작성하는 과정

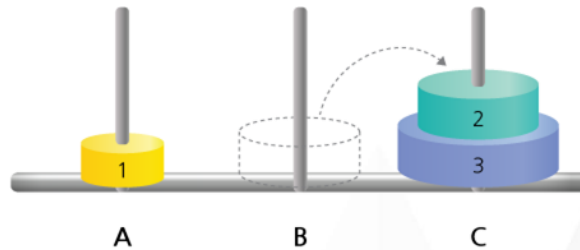
원반 3개의 경우

[5단계]



[원반 1을 B에서 A로 이동]

[6단계]

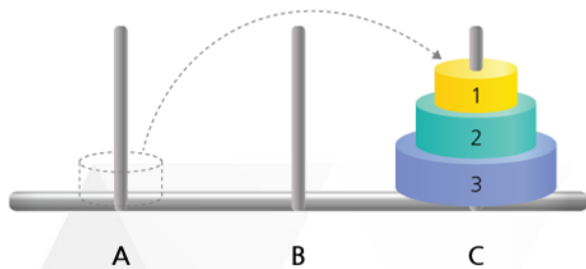


[원반 2을 B에서 C로 이동]

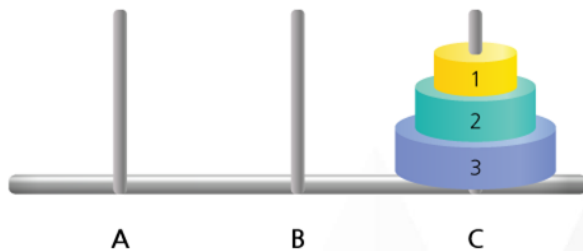
5 문제를 알고리즘으로 작성하는 과정

원반 3개의 경우

[7단계]



[원반 1을 A에서 C로 이동]



[이동완료]

5 문제를 알고리즘으로 작성하는 과정

하노이 탑 점화식

명백하게 $a_1 = 1$ 이다. $n = 2$ 라면 위의 작은 원판을 B로 옮긴 다음 남은 큰 원판을 C로 옮기고 다시 B에 있는 원판을 C에 있는 원판 위에 놓으면 되므로 $a_2 = 3$ 이다. 일반적으로 다음과 같이 3단계로 나누어 생각할 수 있다.

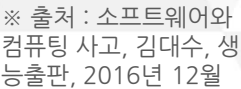
- ① A에 있는 가장 큰 원판만을 제외하고 나머지 $n - 1$ 개의 원판을 B로 옮긴다. 이때 필요한 최소 이동 횟수는 a_{n-1} 이다.
- ② A에 남은 가장 큰 원판을 C로 옮긴다. 물론 필요한 이동 횟수는 1이다.
- ③ B에 있는 $n - 1$ 개의 원판을 C로 옮긴다. 이때 최소 이동 횟수는 a_{n-1} 이다.

그러므로 n 개의 원판을 다른 막대기로 옮길 때 필요한 최소 이동 횟수 a_n 은 $a_1 = 1$, $a_n = 2a_{n-1} + 1$ 이다. ➔ 이 점화식을 풀면 일반항 a_n 을 구할 수 있음 $a_n = 2^n - 1$

5 문제를 알고리즘으로 작성하는 과정

하노이 탑 문제의 결과

- ▶ 만약 원반이 64개이고 1개 옮기는데 1초가 걸린다고 가정하면 5,845억년의 엄청난 시간이 걸림
($a_n = 2^n - 1$ 이므로 $a_{64} = 2^{64} - 1 = \text{약 } 5,845 \text{년}$)



2 알고리즘의 특성

2

알고리즘의 특성

1

알고리즘의 특성

입력(Input)

- ▶ 문제를 풀기 위한 입력이 반드시 필요함

출력(Output)

- ▶ 문제를 해결했을 때 그 결과인 출력이 반드시 존재해야 함

유한성(Finiteness)

- ▶ 알고리즘은 일정한 시간내에 반드시 종료되어야 함
- ▶ 알고리즘 수행이 끝나지 않거나 매우 오래 걸리면 해를 얻을 수 없음

2

알고리즘의 특성

1

알고리즘의 특성

정확성(Correctness)

- ▶ 주어진 문제에 대한 정확한 출력값을 만들어야 함

일반성(Generality)

- ▶ 같은 유형의 문제에 모두 적용 가능

2 알고리즘은 생각하는 방법을 훈련하는 것

- ▶ 문제 자체를 해결하는 알고리즘을 배우
- ▶ 그 과정에 깃든 ‘생각하는 방법’을 배우는 것이 더 중요
(체계적으로 생각하는 훈련)
- ▶ 미래에 다른 문제를 해결하는 생각의 빌딩블록을 제공

3 알고리즘의 표현 방법

순서도(Flow Chart)

- ▶ 문제 해결 과정을 기호와 도형을 사용하여 표현하는 방식

의사코드(Pseudocode)

- ▶ 프로그램 명령문 형식을 취하고 각 명령을 사람이 이해하기 쉽게 적당한 뜻을 가진 단어로 나타냄

3

알고리즘의 표현 방법

1

알고리즘의 표현 방법

일반적인 언어

▶ 사람이 사용하는 문장으로 설명

알고리즘은 여러 가지 방법으로 표현 가능

- ▶ 알고리즘은 문제를 해결하기 위한 체계적이고 순서적인 절차
- ▶ 논리적 표현으로 변환하면 순서도나 의사코드로 변환 가능
- ▶ 알고리즘을 프로그래밍 언어로 변환하면 컴퓨터 프로그램
- ▶ 최종적으로 컴퓨터 프로그램을 실행하여 결과를 얻음

3

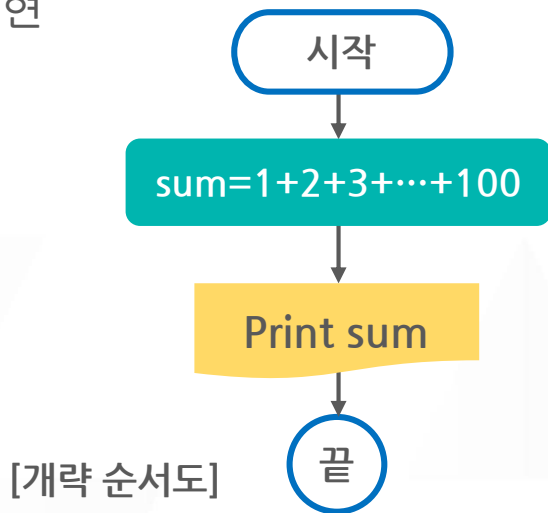
알고리즘의 표현 방법

2

알고리즘의 구현 과정

- ▶ 예) 1부터 100까지의 자연수 덧셈 알고리즘을
순서도와 의사코드로 표현하시오.

풀이) 순서도로 표현



3

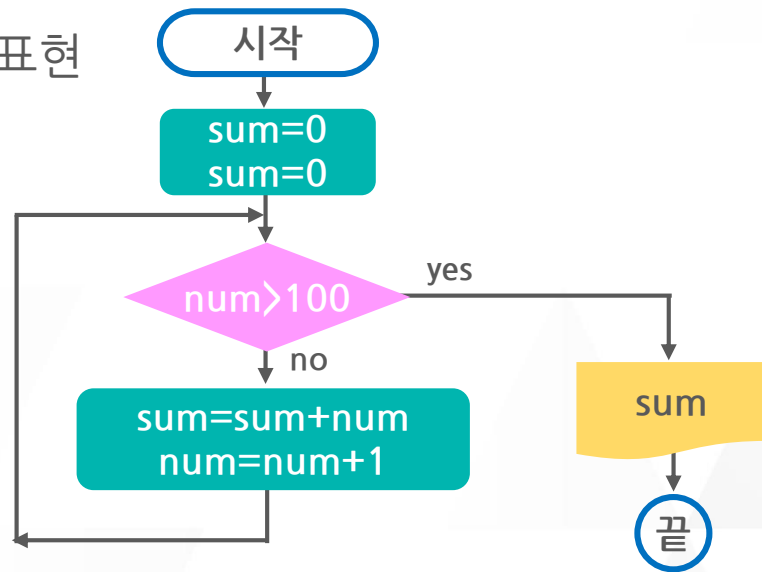
알고리즘의 표현 방법

2

알고리즘의 구현 과정

- ▶ 예) 1부터 100까지의 자연수 덧셈 알고리즘을
순서도와 의사코드로 표현하시오.

풀이) 순서도로 표현



[상세 순서도]

2 알고리즘의 구현 과정

- ▶ 예) 1부터 100까지의 자연수 덧셈 알고리즘을
순서도와 의사코드로 표현하시오.

풀이) 의사코드로 표현

```
algorithm total(){  
    sum = 0;  
    num = 1;  
    while(num <=100)  
    {  
        sum = sum+num;  
        num = num+1;  
    }  
    print sum;  
}
```

- ① 변수 sum에 0, num에 1을 저장
- ② num > 100이면 ⑤로 이동
- ③ sum에 num을 더하여 sum의 값에 저장
- ④ num의 값을 1 증가시키고 ②로 이동
- ⑤ sum의 값을 출력하고 종료