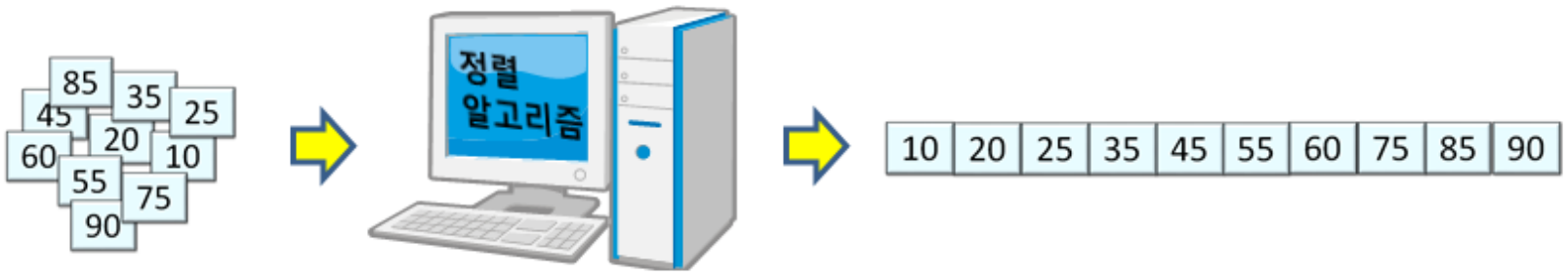
2. 알고리즘을 배우기 위한 준비

목차

2.1 알고리즘이란

- 알고리즘
  - 문제를 해결하는 단계적 절차 또는 방법
  - 여기서 주어지는 문제는 컴퓨터를 이용하여 해결할 수 있어야 한다.
  - 알고리즘에는 입력이 주어지고, 알고리즘은 수행한 결과인 해 (또는 답)를 출력



알고리즘의 일반적 특성

- 정확성 (Correctness)
  - 알고리즘은 주어진 입력에 대해 올바른 해를 주어야 한다.
- 수행성 (Runability)
  - 알고리즘의 각 단계는 컴퓨터에서 수행 가능해야 한다.
- 유한성 (Finiteness)
  - 알고리즘은 일정한 시간 내에 종료되어야 한다.
- 효율성 (Efficiency)
  - 알고리즘은 효율적일수록 그 가치가 높아진다.

2.2 최초의 알고리즘

- 유클리드(Euclid)의 최대공약수 알고리즘
  - 기원전 300년경에 만들어진 가장 오래된 알고리즘
  - 최대공약수는 2개 이상의 자연수의 공약수들 중에서 가장 큰 수

2.2 최초의 알고리즘

- 유클리드(Euclid)의 최대공약수 알고리즘
  - 2개의 자연수의 최대공약수는 큰 수에서 작은 수를 뺀 수와 작은 수와의 최대공약수와 같다는 성질을 이용하여 최대공약수를 찾음

유클리드의 최대공약수 알고리즘

최대공약수(24, 14)

= 최대공약수(24-14, 14)
= 최대공약수(14-10, 10)
= 최대공약수(10-4, 4)
= 최대공약수(6-4, 4)
= 최대공약수(4-2, 2)
= 2

= 최대공약수(10, 14)
= 최대공약수(4, 10)
= 최대공약수(6, 4)
= 최대공약수(2, 4)
= 최대공약수(2, 2)

Pseudo Code of Euclid Algorithm

- N = arg1, M=arg2
- if (N==M)

  then return N
- Else if (N=1 or M=1)

  then return 1
- Else
  - if N>M, then N=N-M
  - Else M = M-N

Euclid(a, b)

입력: 정수 a, b; 단, a≥b≥0

출력: 최대공약수(a, b)

1. If (b=0) return a
2. return Euclid(b, a mod b)

GCD(24, 14)

- Line 1: b=14이므로 if-조건이 '거짓'
- Line 2: Euclid(14, 24 mod 14) = Euclid(14, 10) 호출
- Line 1: b=10이므로 if-조건이 '거짓'
- Line 2: Euclid(10, 14 mod 10) = Euclid(10, 4) 호출
- Line 1: b=4이므로 if-조건이 '거짓'
- Line 2: Euclid(4, 10 mod 4) = Euclid(4, 2) 호출
- Line 1: b=2이므로 if-조건이 '거짓'
- Line 2: Euclid(2, 4 mod 2) = Euclid(2, 0) 호출
- Line 1: b=0이므로 if-조건이 '참 ' 이 되어 a=2를 최종 리턴

2.3 알고리즘의 표현 방법

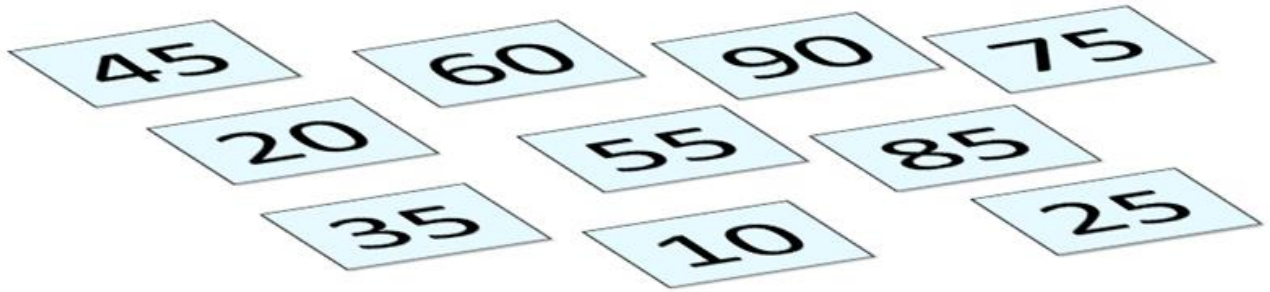
- 알고리즘의 형태는 단계별 절차이므로, 마치 요리책의 요리를 만드는 절차와 유사
- 알고리즘의 각 단계는 보통 말로 서술할 수 있으며, 컴퓨터 프로그래밍 언어로만 표현할 필요는 없음

알고리즘의 표현 방법 4가지

1. Natural Language
2. Flow Chart
3. Pseudo Code
4. Programming Code

최대 숫자 찾기 문제를 위한 알고리즘

- 최대 숫자 찾기



  - 카드의 숫자를 하나씩 비교하면서 본 숫자들 중에서 가장 큰 숫자를 기억해가며 진행하는 방법
  - 마지막 카드의 숫자를 본 후에, 머릿속에 기억된 가장 큰 숫자가 적힌 카드를 바닥에서 집어 든다.

최대 숫자 찾기 문제를 위한 알고리즘

- 보통 말로 표현된 알고리즘
  1. 첫 카드의 숫자를 읽고 머릿속에 기억해 둔다.
  2. 다음 카드의 숫자를 읽고, 그 숫자를 머릿속의 숫자와 비교한다.
  3. 비교 후 큰 숫자를 머릿속에 기억해 둔다.
  4. 다음에 읽을 카드가 남아있으면 line 2로 간다.
  5. 머릿속에 기억된 숫자가 최대 숫자이다.

최대 숫자 찾기 문제를 위한 알고리즘

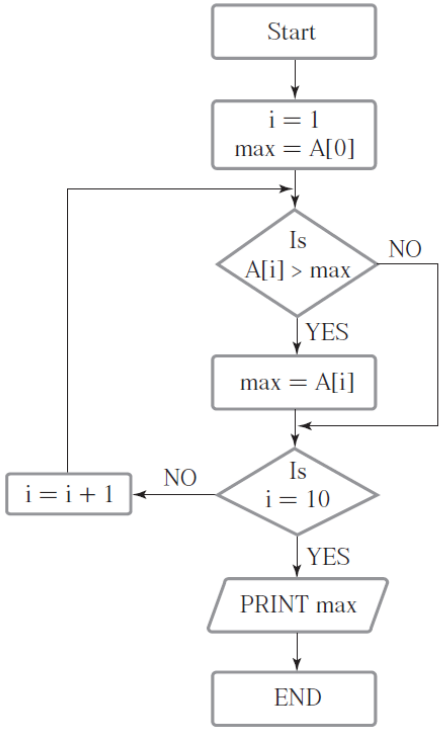
- 의사 코드로 표현된 알고리즘

  배열 A에 10개의 숫자가 있다고 가정

  1. max = A[0]
  2. for i = 1 to 9
  3. if (A[i] > max) max = A[i]
  4. return max

최대 숫자 찾기 문제를 위한 알고리즘

- 플로 차트 (flow chart) 형태



2.4 알고리즘의 분류

- 문제의 해결 방식에 따른 분류(By Strategy)
  - 분할 정복 (Divide-and-Conquer) 알고리즘
  - 그리디 (Greedy) 알고리즘
  - 동적 계획 (Dynamic Programming) 알고리즘
  - 근사 (Approximation) 알고리즘
  - Search
    - 백트래킹 (Backtracking) 기법
    - 분기 한정 (Branch-and-Bound) 기법

알고리즘의 분류

- 문제에 기반한 분류 (By Problem)
  - 정렬 알고리즘
  - 그래프 알고리즘
  - 기하 알고리즘
- 특정 환경에 따른 분류 (by Computing Environment)
  - 병렬 (Parallel) 알고리즘
  - 분산 (Distributed) 알고리즘
  - 양자 (Quantum) 알고리즘

알고리즘의 설계과정

- 문제
- 설계
- 알고리즘 제안
- 분석
- 끝? (4 properties)

알고리즘의 평가

- 하나의 문제를 다양한 알고리즘으로 해결 가능
  - 가장 적합한 알고리즘을 찾는 것이 중요
- 좋은 알고리즘의 조건

1. 신뢰성이 높을 것: 정확도가 높고 올바른 결과를 만들어 냄
2. 처리 효율이 좋을 것: 계산 횟수가 적고 처리 속도가 빨라야 함
3. 일반적일 것: 특정 상황에서만 사용되는 알고리즘이 아니라 일반적 상황에서 사용 가능
4. 확장성이 있을 것: 변경 사항을 반영하여 간단하게 수정 가능
5. 알아 보기 쉬울 것: 알기 쉬워야 함, 이해하기 어려운 알고리즘은 유지 보수에 방해
6. 이식성이 높을 것: 유용한 프로그램은 타 기종에서도 사용할 가능성이 높음

알고리즘의 분석

- Space Complexity
- Time complexity