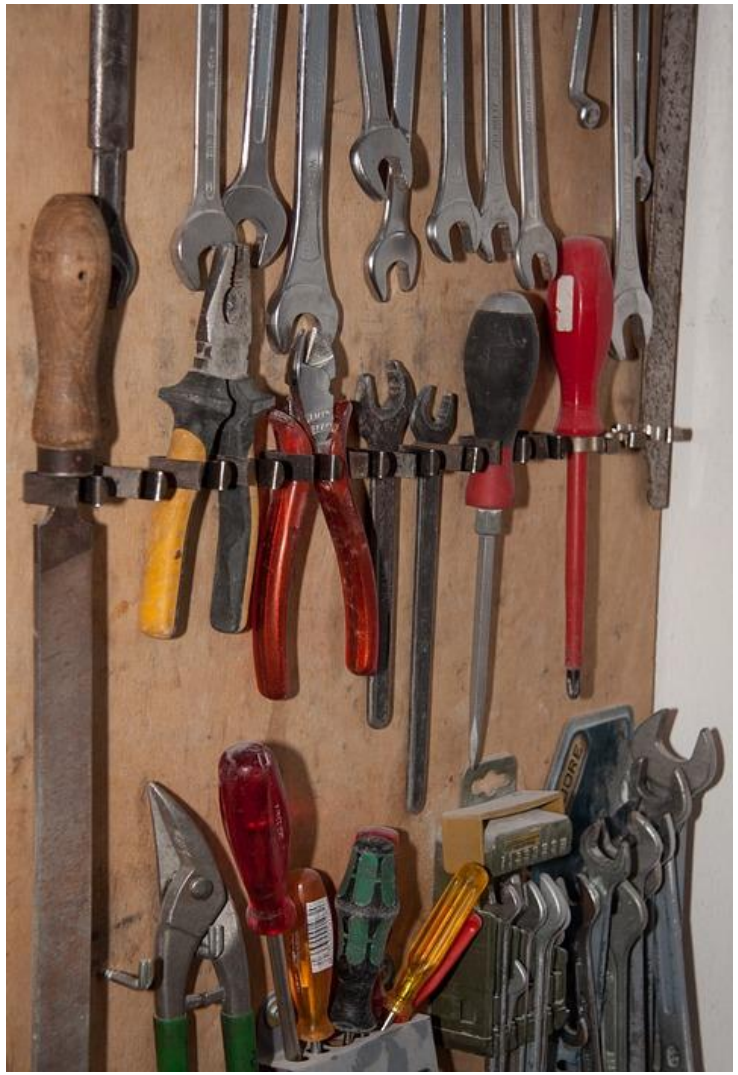
10주차

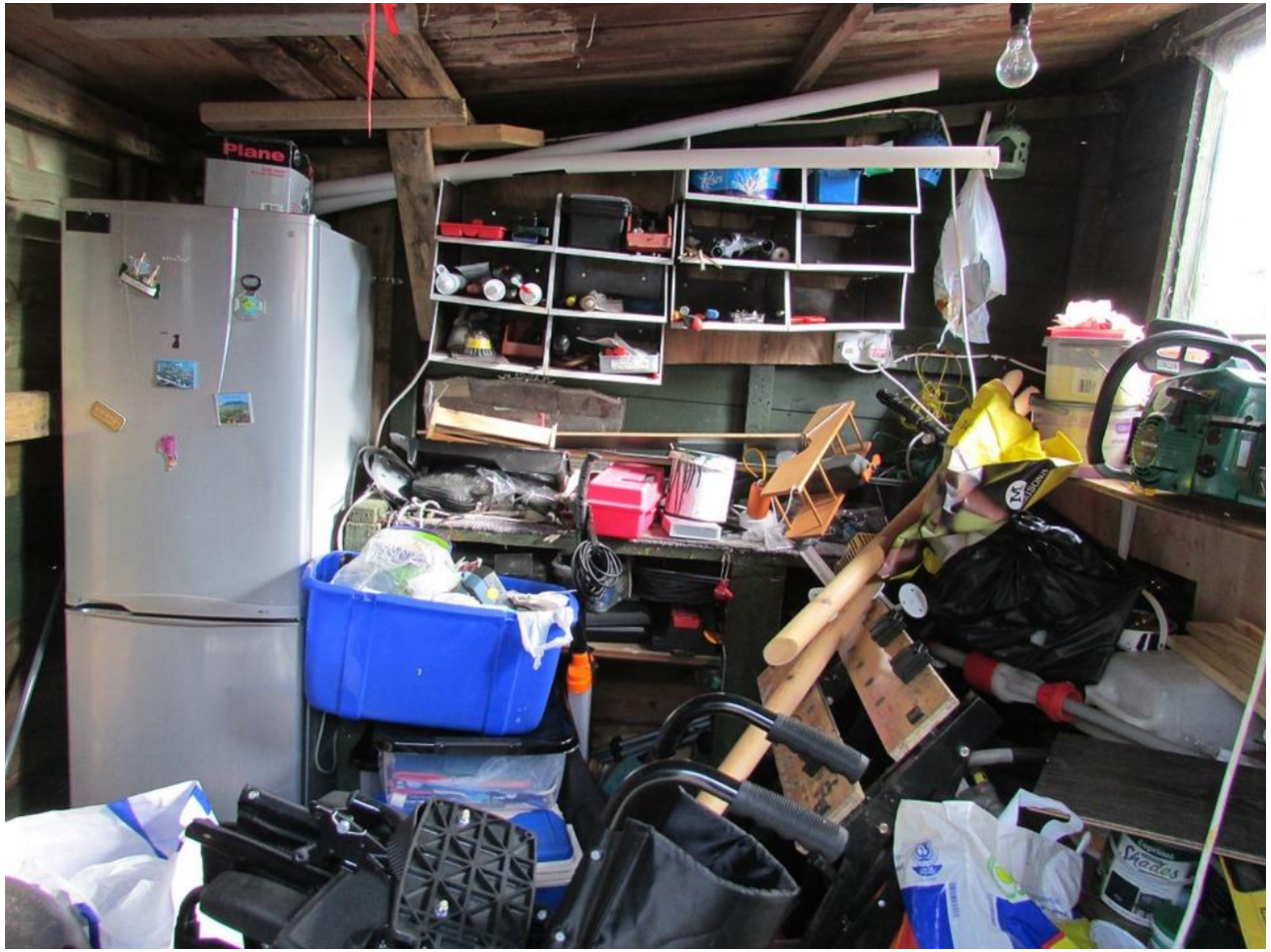
데이터의 저장과 검색

- 배열 및 연결 리스트
- 선형검색
- 이분검색
- 색인순차검색
- 해싱

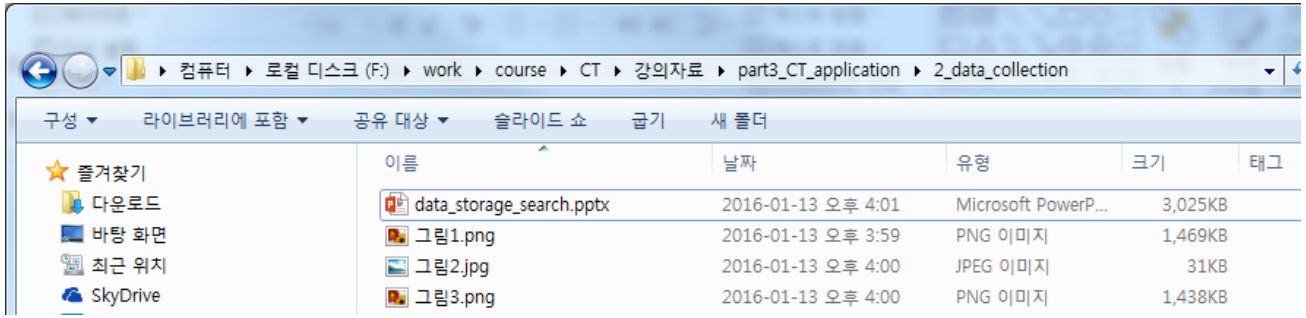
		강의 주제	주차
1부 컴퓨팅 사고력	컴퓨팅 사고력의 소개	IT 사회, 소프트웨어 세상, 컴퓨팅 사고력의 소개, 컴퓨팅 사고력의 개념	1
		컴퓨팅 사고력의 개념, 주위에서 볼 수 있는 컴퓨팅 사고력, 문제해결 방법	2
	문제해결 방법, 컴퓨터	문제해결 방법, 문제해결 과정 예, 문제해결을 위한 소프트웨어 설계 사상, 컴퓨터의 특징, 소프트웨어, 유한상태기계	3
2부 소프트웨어	알고리즘	알고리즘 소개, 알고리즘의 표현 방법, 의사코드, 흐름도	4
	프로그램	프로그램의 기능, 함수, 컴파일러	5
	파이썬	파이썬 소개 및 설치, 변수에 값 저장, 입력, 출력, 조건부 수행	6
		반복, 리스트, 함수, 출력 형식	7
3부 컴퓨팅 사고력 활용하기	데이터의 표현	이진수, 아스키코드, 오디오 데이터, 이미지 데이터, 자료구조	8
		인코딩 및 압축, 오류확인	9
	데이터의 저장과 검색	배열 및 연결 리스트, 선형검색, 이분검색, 색인순차검색, 해싱	10
		이진검색트리, 최대값 및 최소값 검색	11
	알고리즘설계	정렬, 분할정복 알고리즘, 탐욕적 알고리즘	12

- 일상 생활에서 물건을 모아서 잘 정리할 필요가 있다.




Plane

색인(인덱스)


컴퓨터 ▸ 로컬 디스크 (F:) ▸ work ▸ course ▸ CT ▸ 강의자료 ▸ part3_CT_application ▸ 2_data_collection
구성
라이브러리에 포함
공유 대상
슬라이드 쇼
굽기
새 폴더
즐겨찾기
다운로드
바탕 화면
최근 위치
SkyDrive
이름
날짜
유형
크기
태그
data_storage_search.pptx 2016-01-13 오후 4:01 Microsoft PowerP... 3,025KB
그림1.png 2016-01-13 오후 3:59 PNG 이미지 1,469KB
그림2.jpg 2016-01-13 오후 4:00 JPEG 이미지 31KB
그림3.png 2016-01-13 오후 4:00 PNG 이미지 1,438KB

윈도우 탐색기

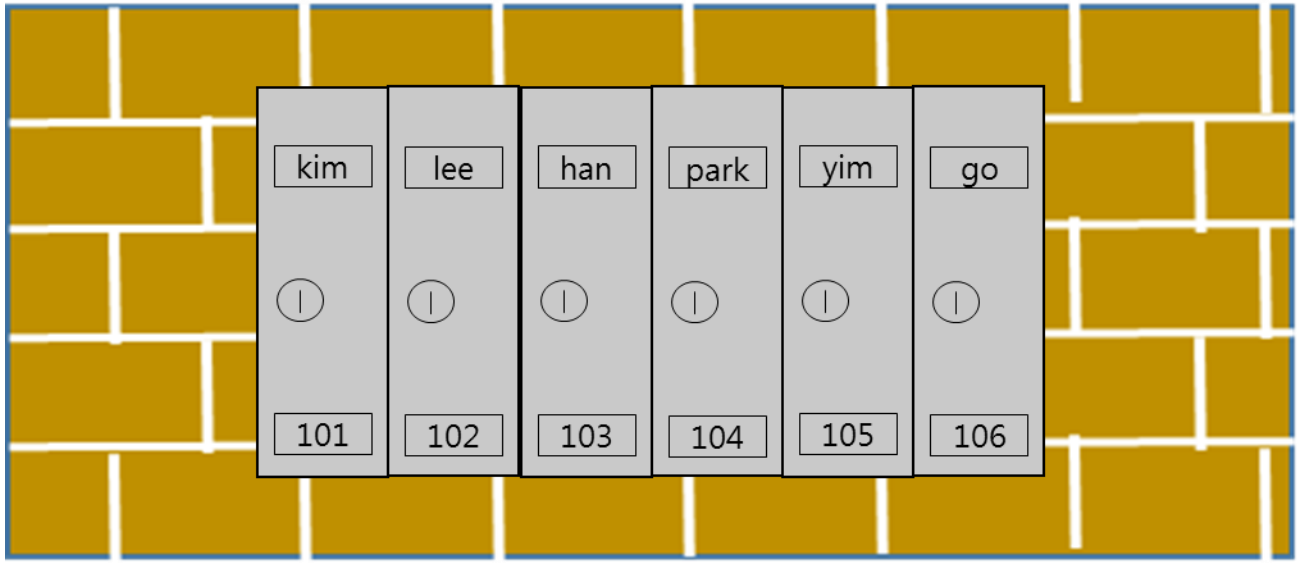
데이터의 저장과 검색 (Data Storage and Search)

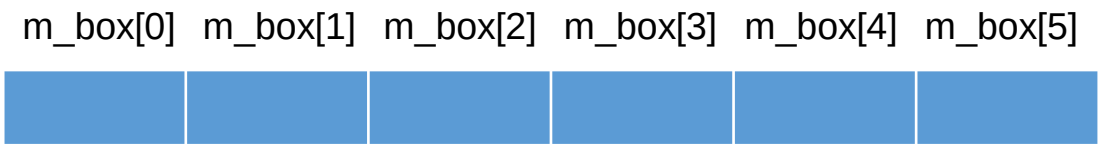
- 데이터의 추후 활용을 위해 데이터를 저장한다.
- 저장할 때 가능한 한 저장 공간을 줄인다.
- 활용할 때를 위해 쉽게 데이터를 찾을 수 있도록 저장한다.
- 데이터를 찾는 방법과 저장 방법과는 직접적인 관계가 있다.
- 대규모의 데이터를 저장, 검색하므로 저장, 검색 방법의 중요성 증대

배열과 단순 연결 리스트

배열

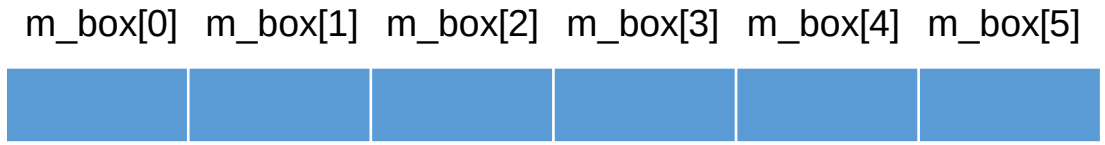
- 변수명[ ] 의 형태로 프로그래밍 언어에서 사용
- a[0], b[3], mail_box[5], sale_2015[7] 등
- C++, Java, Python에서 array index는 0 에서 시작





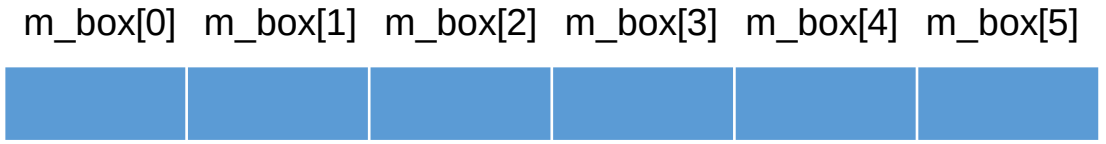
2015 sale		
Jan	150	sale_2015[0]
Feb	200	sale_2015[1]
Mar	170	sale_2015[2]
April	188	sale_2015[3]
May	175	sale_2015[4]
June	180	sale_2015[5]
July	182	sale_2015[6]
Aug	155	sale_2015[7]
Sept	160	sale_2015[8]
Oct	199	sale_2015[9]
Nov	193	sale_2015[10]
Dec	184	sale_2015[11]

m_box[6]



sale_2015[12]

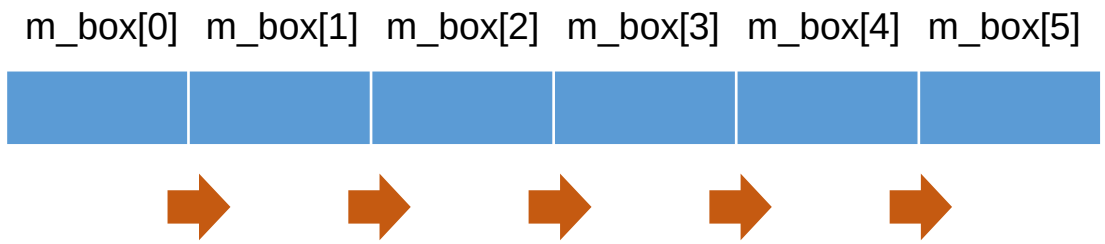
대표의 이름을 사용하면서, 데이터가 저장된 위치 첨자(인덱스) 사용



- 사용할 공간을 초기에 설정한 후, 변경 불가능
- 초기에 설정한 데이터의 개수보다 더 많은 데이터가 입력될 경우 저장이 불가능
- 크기가 고정된 데이터 저장에 사용(예, 월별 판매액)
- 데이터가 저장된 위치(첨자, 인덱스)를 아는 경우 데이터 값을 쉽게 확인 (예, 2015년 5월 판매액)

```
value = sale_2015[4]
```

- 배열에 저장되어 있는 특정 데이터의 저장 위치(첨자,인덱스)를 모르는 상태에서 특정 데이터를 찾기 위해서는 배열의 제일 처음부터 위치를 변경하면서 하나씩 찾아 나가야 한다.



파이썬에서의 배열(리스트)

```
a=[5,”abc”,2]
print(a)
print(a[0])
print(a[1])
print(a[2])
```

```
[5, 'abc', 2]
5
abc
2
```

```
for i in range(0,len(a)):
    print(a[i])
```

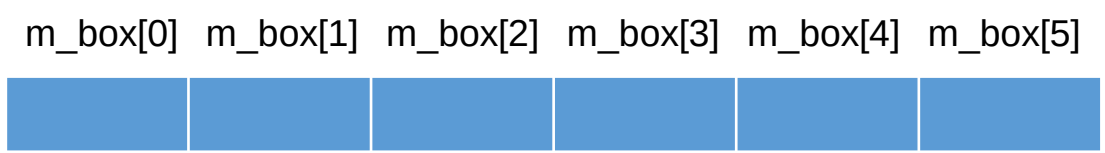
```
5
abc
2
```

```
for k in a:
    print(k)
```

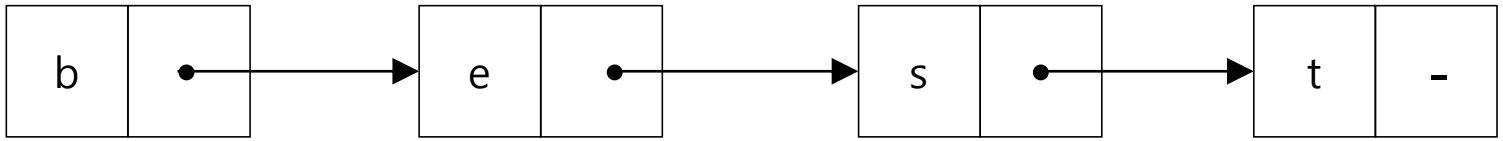
```
5
abc
2
```

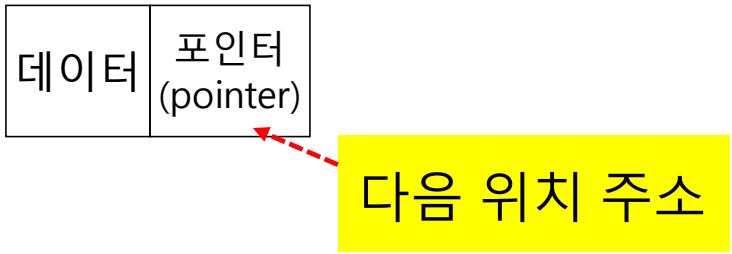
배열을 사용할 경우의 장점

- 하나의 변수 명을 이용하여 많은 데이터를 저장
- 데이터의 위치(인덱스)를 이용하여 데이터 접근 가능
- 반복적인 작업을 수행할 때 효과적



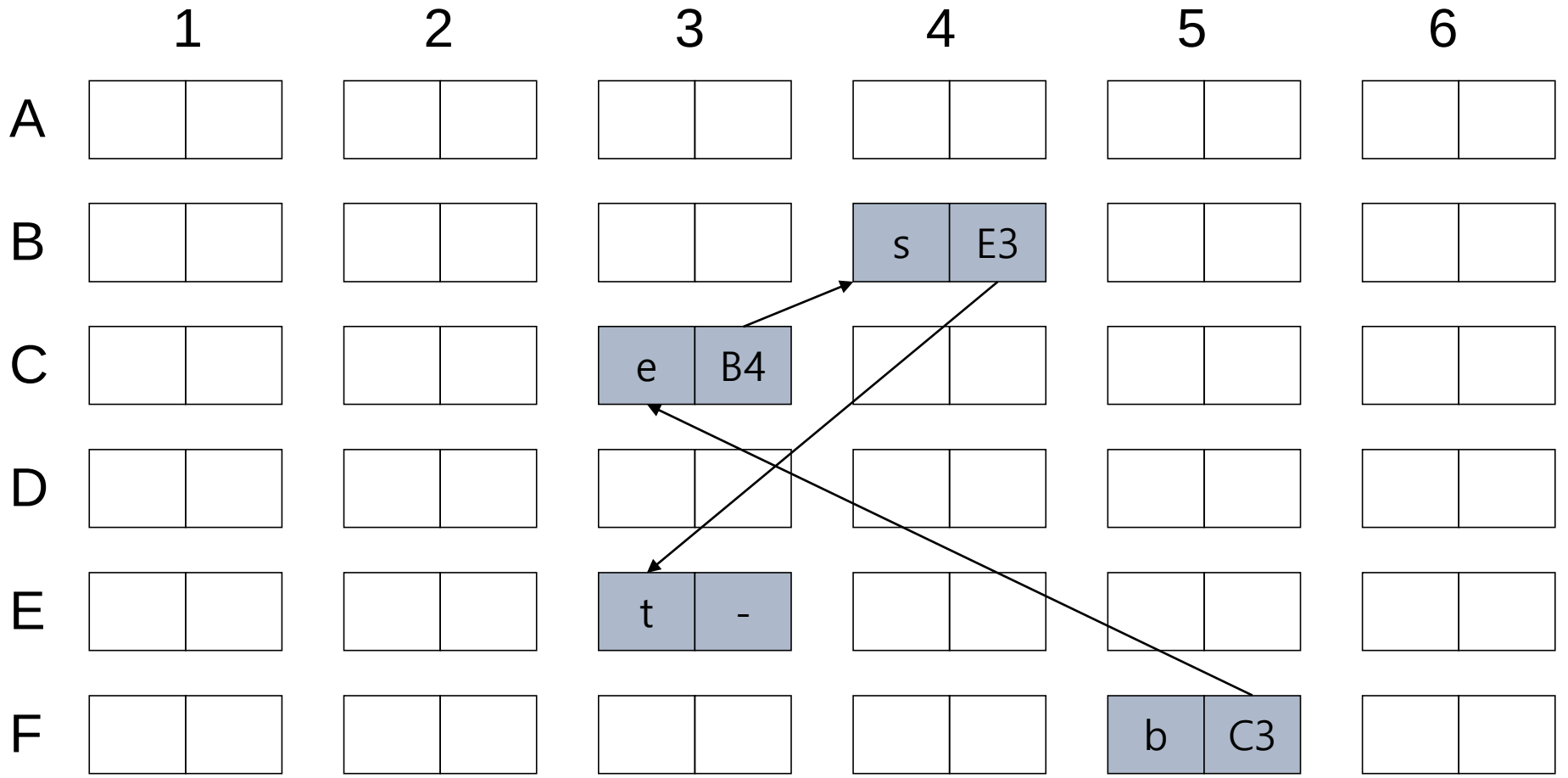
단순 연결 리스트





- 한 방향으로만 이동 가능

컴퓨터 메모리 공간에서

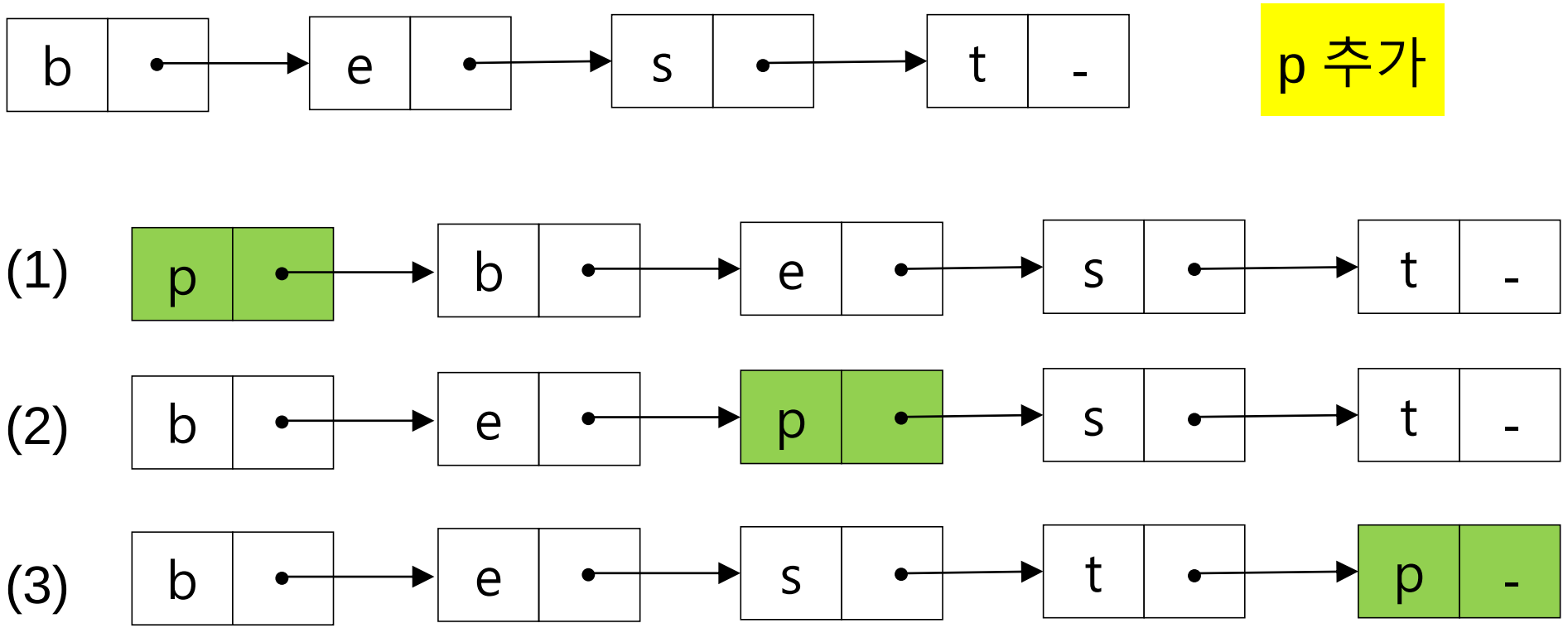


data	다음 데이터 번지수

✓ 읽히는 데이터는

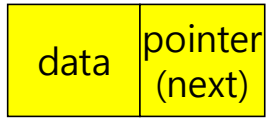
b	e	s	t

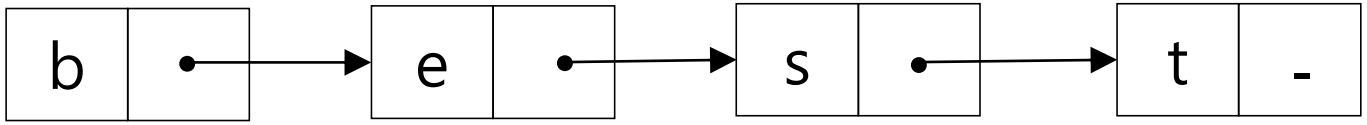
- 데이터의 개수가 가변적일 때 사용
- 메모리 크기의 한도에 도달하지 않는 한, 데이터의 추가 가능



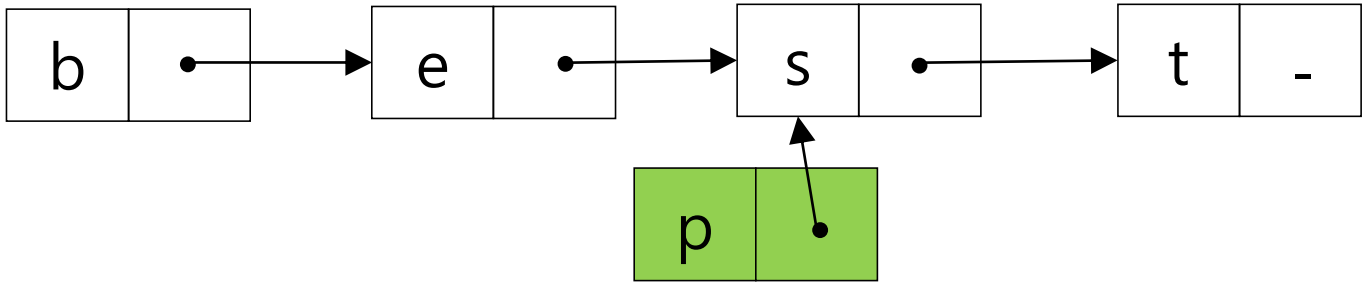
- 어느 위치에도 추가 가능

- e 다음에 p를 삽입하는 방법

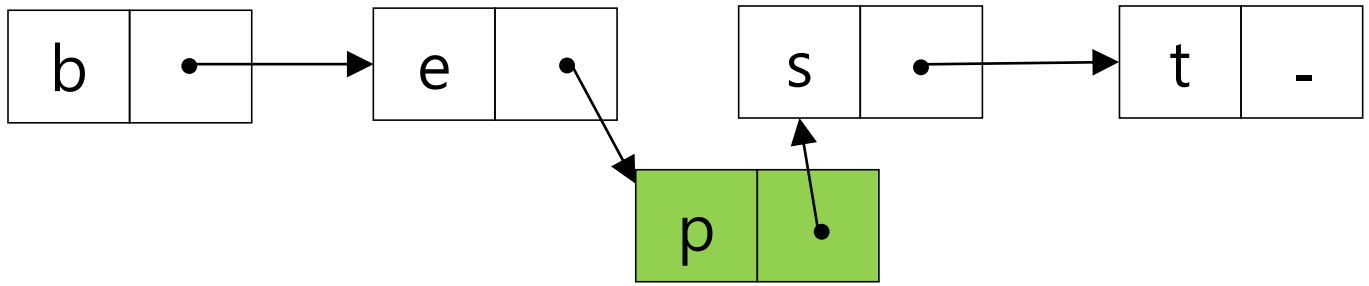


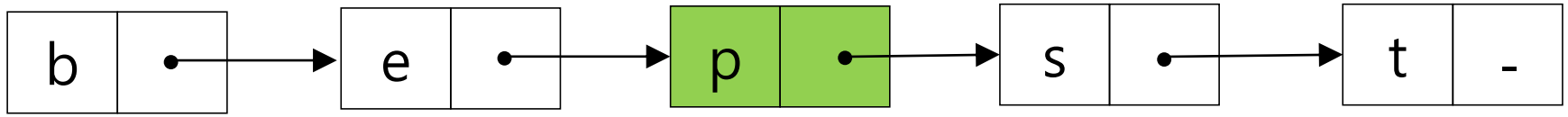


(1단계)

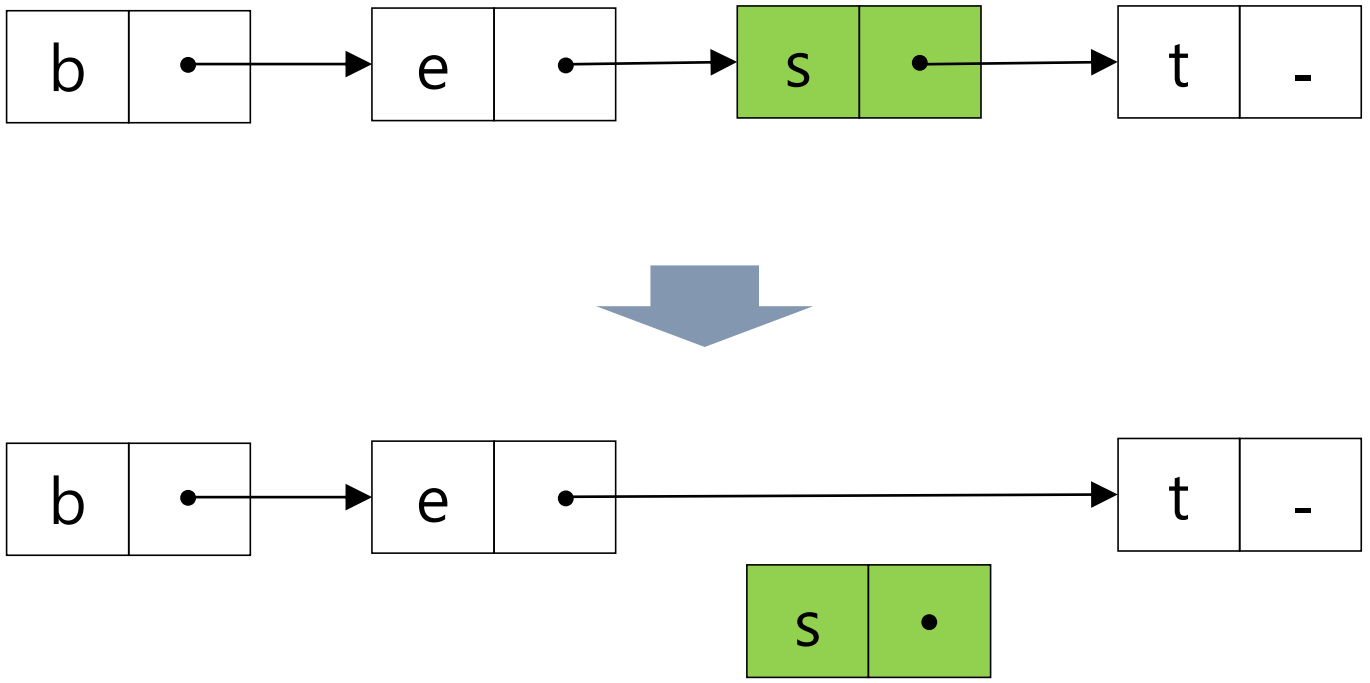


(2단계)

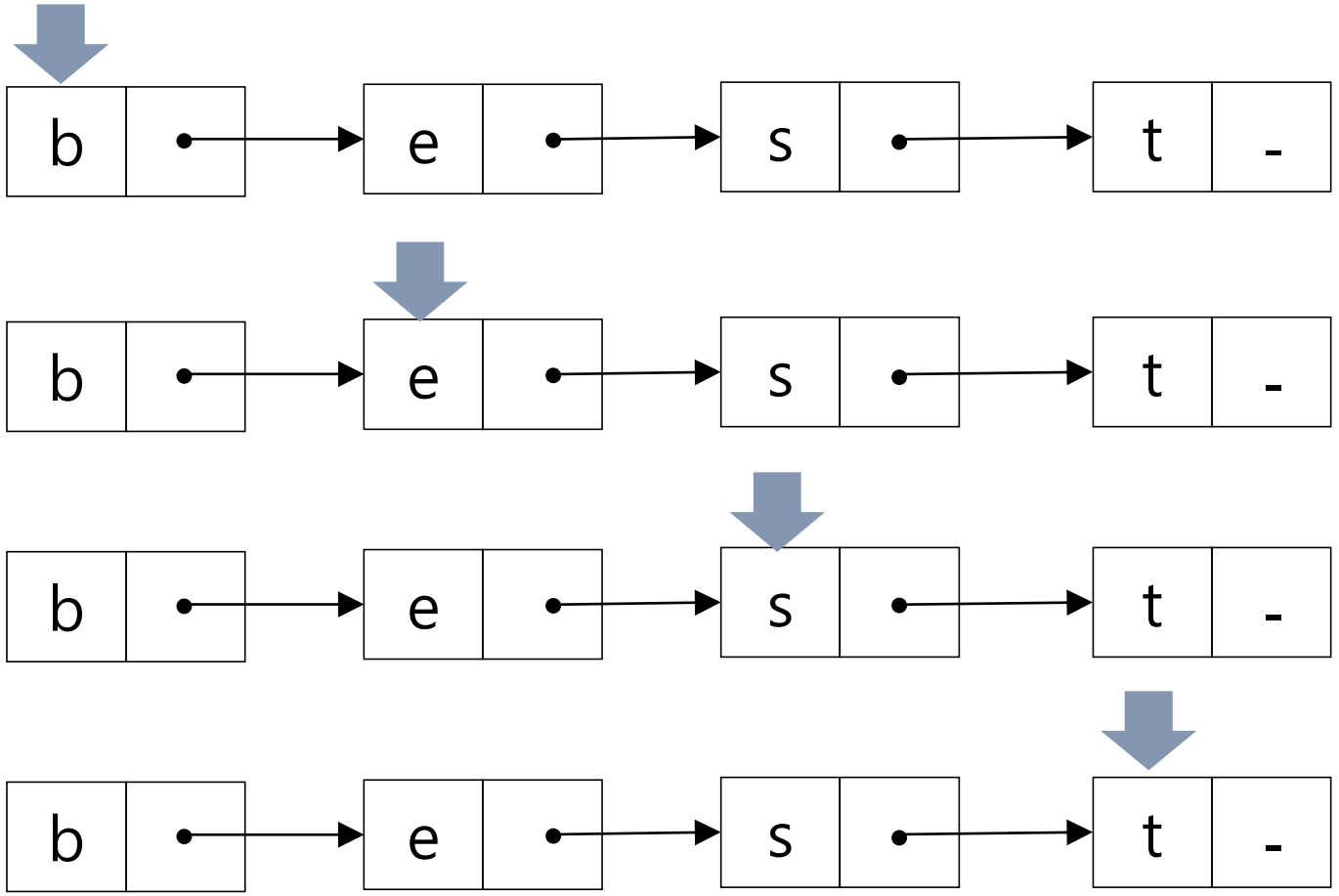




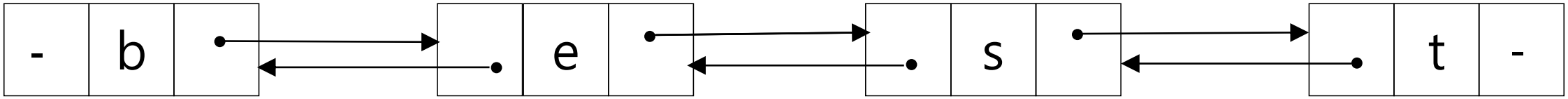
- 데이터의 삭제가 쉽다
  - ✓ s의 삭제



- 특정 데이터를 찾기 위해서는 처음 데이터부터 확인하고, 링크를 따라 이동하면서 확인 해야 한다.
- 데이터 t의 검색



이중 연결 리스트 표현

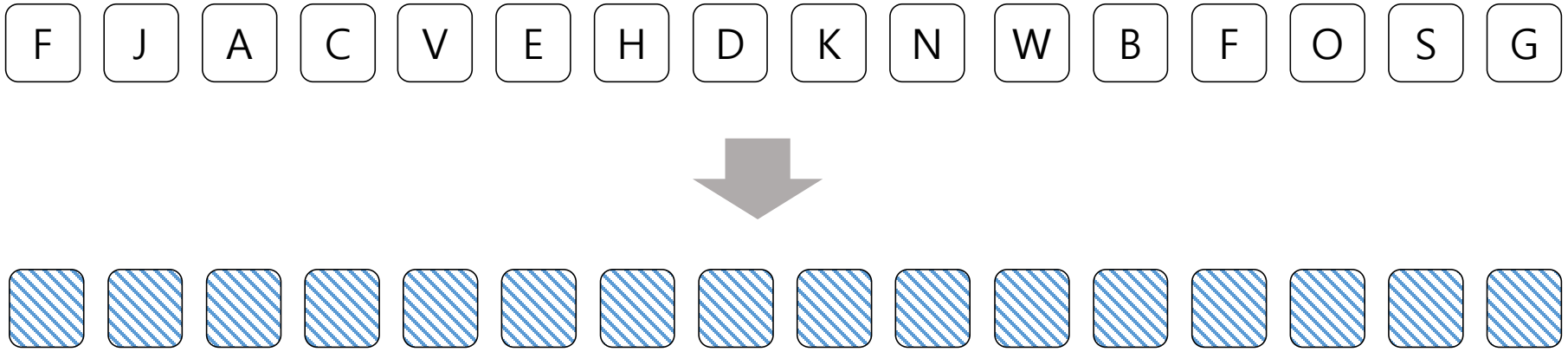


선형검색

Linear Search

[problem]

무작위로 문자가 표시되어 있는 카드 16장을 뒤집어 놓는다. 이 중 특정 카드(예, K)가 표시된 카드를 찾는다



문제 해결 과정

1. 문제를 이해한다
2. 계획을 세운다
3. 계획을 실행한다
4. 풀이과정을 재점검한다

Computational Thinking

- 분해
- 패턴 확인
- 추상화
- 알고리즘 설계
- 데이터 모음
- 데이터 표현
- 데이터 분석
- 모의실험
- 자동화
- 병렬화
- 패턴 일반화

문제 해결 전략

- 추정과 확인
- 순서 리스트 작성
- 가능성들을 제거
- 대칭 성질을 이용
- 특별한 경우를 고려
- 직접 추론 사용
- 방정식을 푼다
- 패턴을 찾는다
- 그림을 그린다
- 단순한 문제 해결 시도
- 모델 사용
- 역방향으로 시도
- 공식 사용
- 창조적 생각
- 잔꾀를 사용

- 알고 있는 데이터나 정보는 무엇인가?

  카드의 장수, 찾으려는 카드

- 모르고 있는 것은 무엇인가?

  찾으려고 하는 카드의 위치

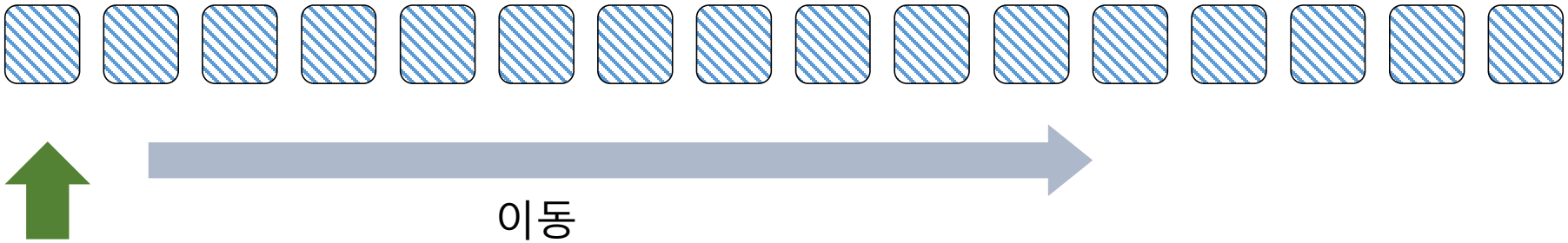
- 조건들은 무엇인가?

  ✓ 한 번에 한 장의 카드의 문자가 보이게 하여 찾는 카드가 맞는지 확인한다.

  ✓ 찾는 카드가 전체 카드 내에 반드시 있는지 불분명하다. 있을 경우와 없을 경우로 나누어서 생각해야 한다.

[해결 방법]

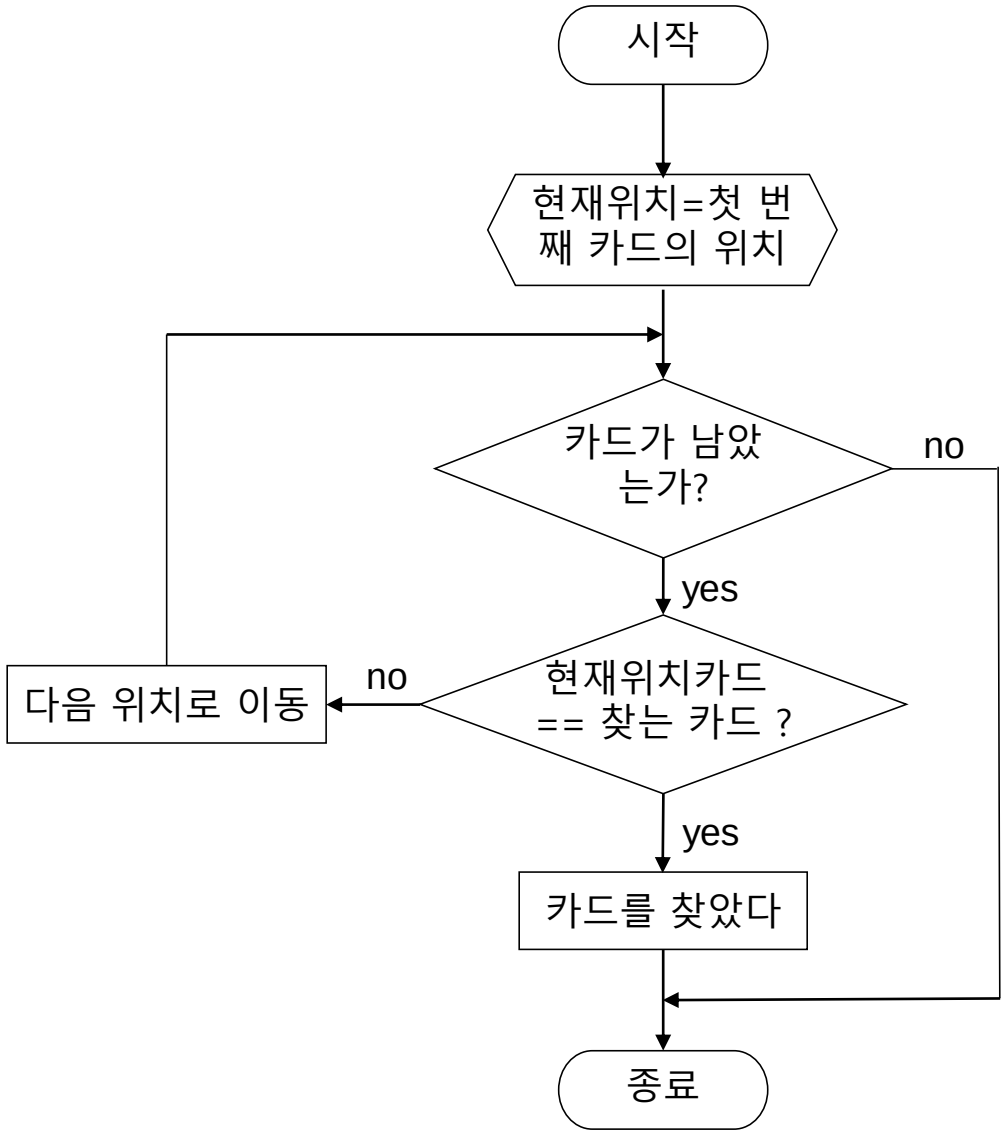
- 카드의 가장 왼쪽(처음)부터 시작하여 한 장씩 뒤집으면서, 찾는 카드가 맞는지 확인한다.
- 찾는 카드가 맞으면 종료. 아니면 오른쪽에 있는 다음의 카드로 이동



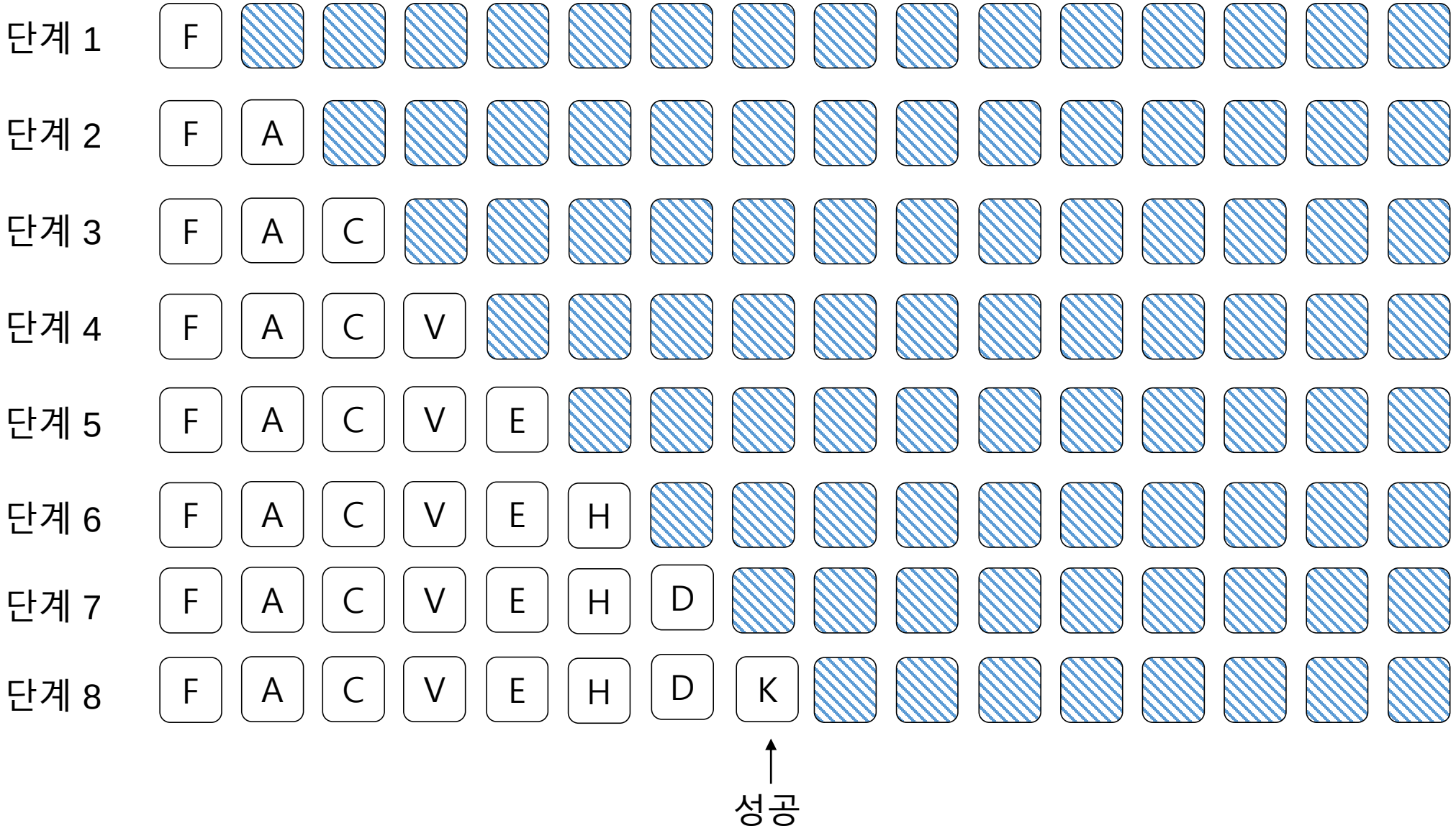
의사코드

```
현재 위치 = 첫 번째 카드의 위치
while (카드열에 확인할 카드가 있고 아직 카드를 못찾은 상태)
        if (현재 위치의 카드 == 찾는 카드)
                특정 카드를 찾았다.
        else
                다음 위치로 이동
```

흐름도

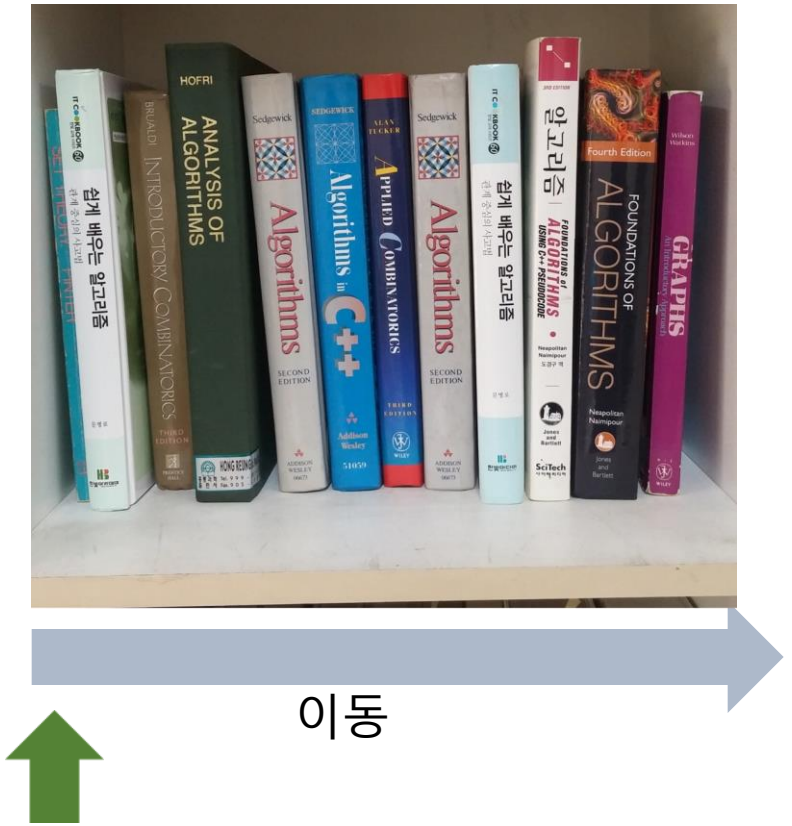

시작
현재위치=첫 번째 카드의 위치
카드가 남았는가?
no
yes
현재위치카드 == 찾는 카드 ?
no
다음 위치로 이동
yes
카드를 찾았다
종료

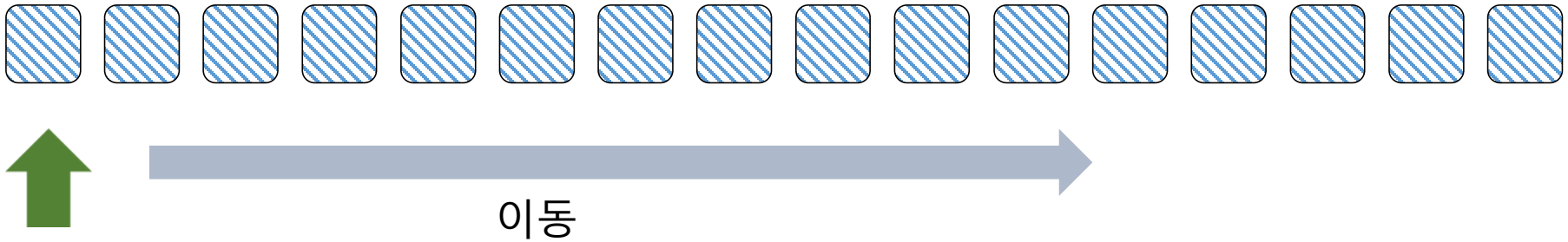
문제: K를 찾는다


단계 1
F
단계 2
F A
단계 3
F A C
단계 4
F A C V
단계 5
F A C V E
단계 6
F A C V E H
단계 7
F A C V E H D
단계 8
F A C V E H D K
성공

- 한 장의 카드를 뒤집어서 찾는 카드가 맞는지 확인하는 작업이 필요하다.
- K를 찾기 위해서는 몇 번의 확인 작업이 필요했나? 8회
- 이러한 방식으로 진행하는 데이터 검색 방법을 선형 검색(linear search) 또는 순차 검색(sequential search)라고 한다.
- 데이터가 선형적으로 정리되어 있을 때 사용 가능 – 배열, 연결 리스트에 저장
- 알고리즘이 단순하고 구현이 쉽다.

책꽂이에 있는 책을 찾는 방식



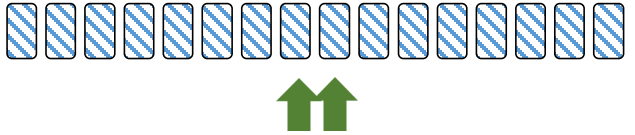


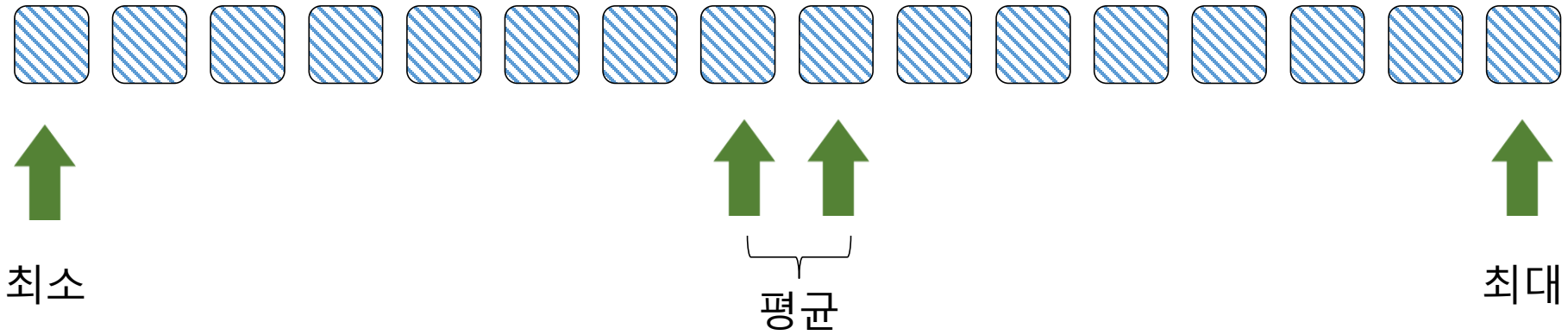
- 이 예에서 카드 한 장을 찾는데 필요한
  - ✓ 최소 확인 횟수는?
  - ✓ 최대 확인 횟수는?
  - ✓ 평균 확인 횟수는?
  - ❖ 카드열 내에 찾는 카드가 있다고 가정한다.

- 최소
  - ✓ 제일 처음(제일 왼쪽)에 있는 카드를 찾을 경우
    - 1회의 확인 작업 필요

- 최대
  - ✓ 제일 끝(제일 오른쪽)에 있는 카드를 찾을 경우
    - 16회의 확인 작업 필요

- 평균
  - ✓ 각 카드를 선택할 확률이 같다고 가정
  - ✓ 전체 카드 개수의 반 ½ 에 해당하는 횟수 필요

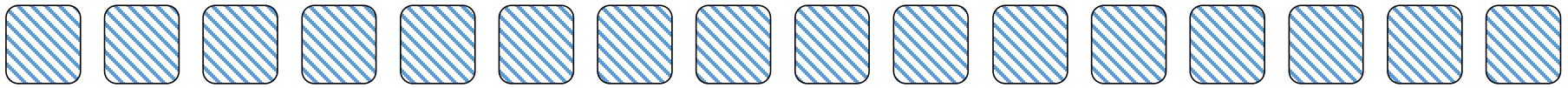



최소
평균
최대

- 찾는 카드가 주어진 카드들에 없을 경우

  ✓ 항상 제일 끝에 있는 카드까지 확인해야 찾는 카드가 없다는 것을 확인할 수 있다.

  ✓ 이 경우 16회의 확인 작업 필요



최소, 평균, 최대

Computational thinking

- 데이터가 일직선으로 저장되어 있는 것처럼 생각할 수 있다.
- 데이터가 배열 또는 연결 리스트에 저장되어 있을 경우 사용 가능
- 이 방법을 사용하기 위해서는 데이터가 정렬(sort)되어 있어야 하나?
- 문자를 선택할 확률이 서로 다르다면 어떻게 될까?
- 데이터의 개수가 n개 일 때 결과는 어떻게 표시할 수 있나?
- 이 방법은 데이터의 개수가 많은 경우 비효율적이다. 왜 그런가?

- 이 문제에서 저장되어 있는 데이터는 선형적(linear)으로 저장되었다고 한다.
- 데이터를 찾는 방법은 선형 검색(linear search)이라고 한다.
- [찾는 데이터가 존재할 경우]

  n개의 데이터에서 특정 데이터를 하나를 찾는데 걸리는 비교횟수는 최소 1회, 최대 n회, 평균은

  (n+1)/2, n이 홀수일 때,

  n/2 또는 n/2+1, n이 짝수 일 때
- [찾는 데이터가 존재하지 않는 경우]

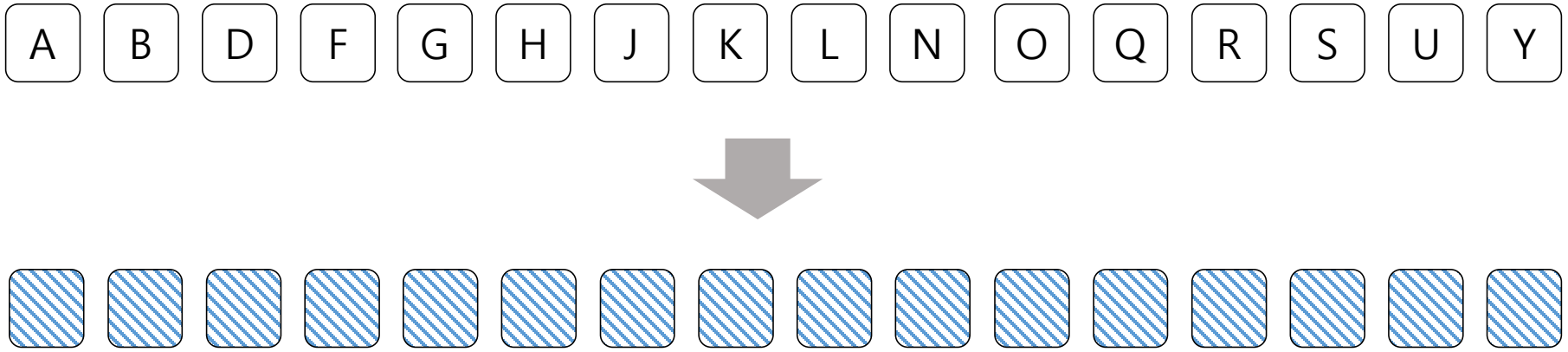
  데이터의 끝까지 가야 없다는 것을 확인할 수 있다. n번의 데이터 비교가 필요하다.

참고: [정렬(sort)]

- 데이터는 숫자, 문자로 구성되어 있고, 각 구성 요소는 값의 대소 순서가 미리 정해져 있다. – 알파벳 순서 또는 0,1,2…
- 데이터 값이 증가 또는 감소하는 순서로 데이터를 정리하는 작업
- 예 – 증가하는 순서(오름차순)
  - ✓ 101, 129, 250, 295, 307, 408
  - ✓ base, cartoon, fire, moon, park, young
  - ✓ 김, 바둑, 사회, 우주, 카누
- 감소하는 순서(내림차순)

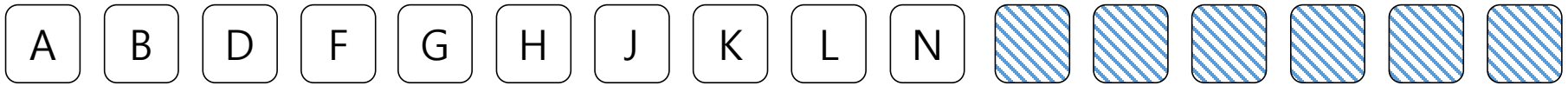
[problem] 정렬된 데이터에 대한 선형검색

문자가 표시되어 있는 카드 16장이 정렬되어 뒤집어져 있다. 이 중 특정 카드(예, K)가 표시된 카드를 찾는다.



- 찾는 데이터가 존재할 경우에는 정렬 안된 데이터와 동일한 결과
- 찾는 데이터가 존재하지 않을 경우에는
  - ✓ 데이터가 없다는 사실을 빨리 알 수 있다.

- [예] M을 찾을 경우: N을 확인하는 순간 M이 없다는 것을 알 수 있다.



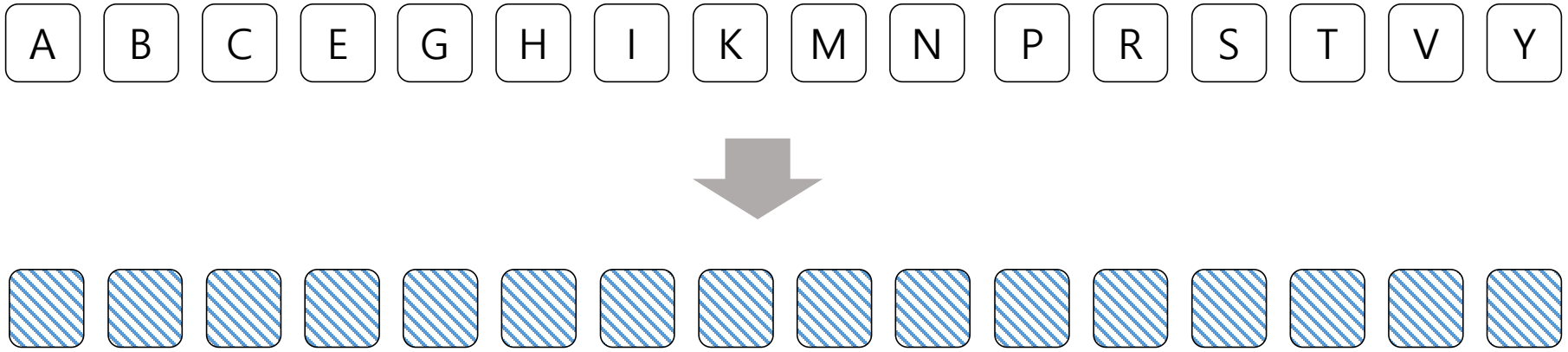
- 정렬된 데이터에 대해서는 선형검색이 아닌 다른 검색 방법 사용
- 이분검색 사용 가능

이분검색

Binary Search

[problem]

- 문자가 표시되어 있는 카드 16장을 뒤집어 놓는다. 이 중 특정 카드(예, G)가 표시된 카드를 찾는다
- 이 때 카드는 오름차순으로 정렬되어 있다.



- Computational Thinking
  - 분해
  - 패턴 확인
  - 추상화
  - 알고리즘 설계
  - 데이터 모음
  - 데이터 표현
  - 데이터 분석
  - 모의실험
  - 자동화
  - 병렬화
  - 패턴 일반화

- 문제 해결 전략
  - 추정과 확인
  - 순서 리스트 작성
  - 가능성들을 제거
  - 대칭 성질을 이용
  - 특별한 경우를 고려
  - 직접 추론 사용
  - 방정식을 푼다
  - 패턴을 찾는다
  - 그림을 그린다
  - 단순한 문제 해결 시도
  - 모델 사용
  - 역방향으로 시도
  - 공식 사용
  - 창조적 생각
  - 잔꾀를 사용

[참고]

- 카드에 표시된 문자를 비교할 때 알파벳 순서에 따라 카드 문자의 대소를 구분한다.

- 예를 들어 문자 A, E는 알파벳 순서에서 A는 E보다 먼저 오게 되므로, A의 값이 작다고 표현한다. 반대로 E는 A보다 값이 크다고 표현한다.

- 카드의 장수= n
  카드열의 중앙은 짝수일 때는 n/2, 홀수일 때 (n+1)/2

[이분검색(binary search)]

단계 1: 카드열의 중앙에 있는 카드를 뒤집어 카드의 문자를 확인한다. 다음의 3가지 경우가 발생한다.

(경우 1) 찾는 문자가 지금 보고 있는 문자보다 작으면 현 카드부터 오른쪽에 있는 카드를 없앤다.

(경우 2) 찾는 문자가 맞다.

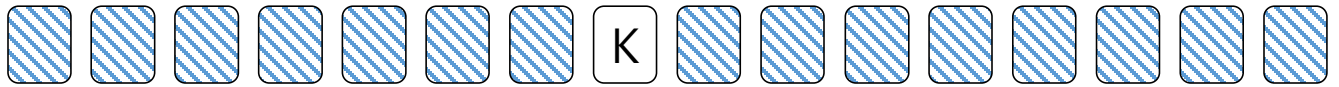
(경우 3) 찾는 문자가 지금 보고 있는 문자보다 크면 현 카드부터 왼쪽에 있는 카드를 없앤다.

단계 2: 남아 있는 카드들에서 단계 1을 반복하여 문자를 찾는다.

문제: G를 찾는다

4장 확인 필요

1. 중앙에 위치한 카드의 문자를 확인한다. 중앙의 위치가 두 개일 때 앞의 것을 선택



2. 찾는 카드가 K보다 작으므로, K가 있는 카드부터 오른쪽에 있는 모든 카드를 제거한다.

3. 중앙에 위치한 카드의 문자를 확인한다.

4. 찾는 카드가 E보다 크므로, E가 있는 카드부터 왼쪽에 있는 모든 카드를 제거한다.

5. 중앙에 위치한 카드의 문자를 확인한다.

6. 찾는 카드가 H보다 작으므로, H가 있는 카드부터 오른쪽에 있는 모든 카드를 제거한다.

7. 남은 한 장 카드의 문자를 확인한다.

문제: Y를 찾는다

5장 확인 필요

1. 중앙에 위치한 카드의 문자를 확인한다. 중앙의 위치가 두 개일 때 앞의 것을 선택

2. 찾는 카드가 K보다 크므로, K가 있는 카드 부터 왼쪽에 있는 모든 카드를 제거한다.

3. 중앙에 위치한 카드의 문자를 확인한다.

4. 찾는 카드가 R보다 크므로, R이 있는 카드 부터 왼쪽에 있는 모든 카드를 제거한다.

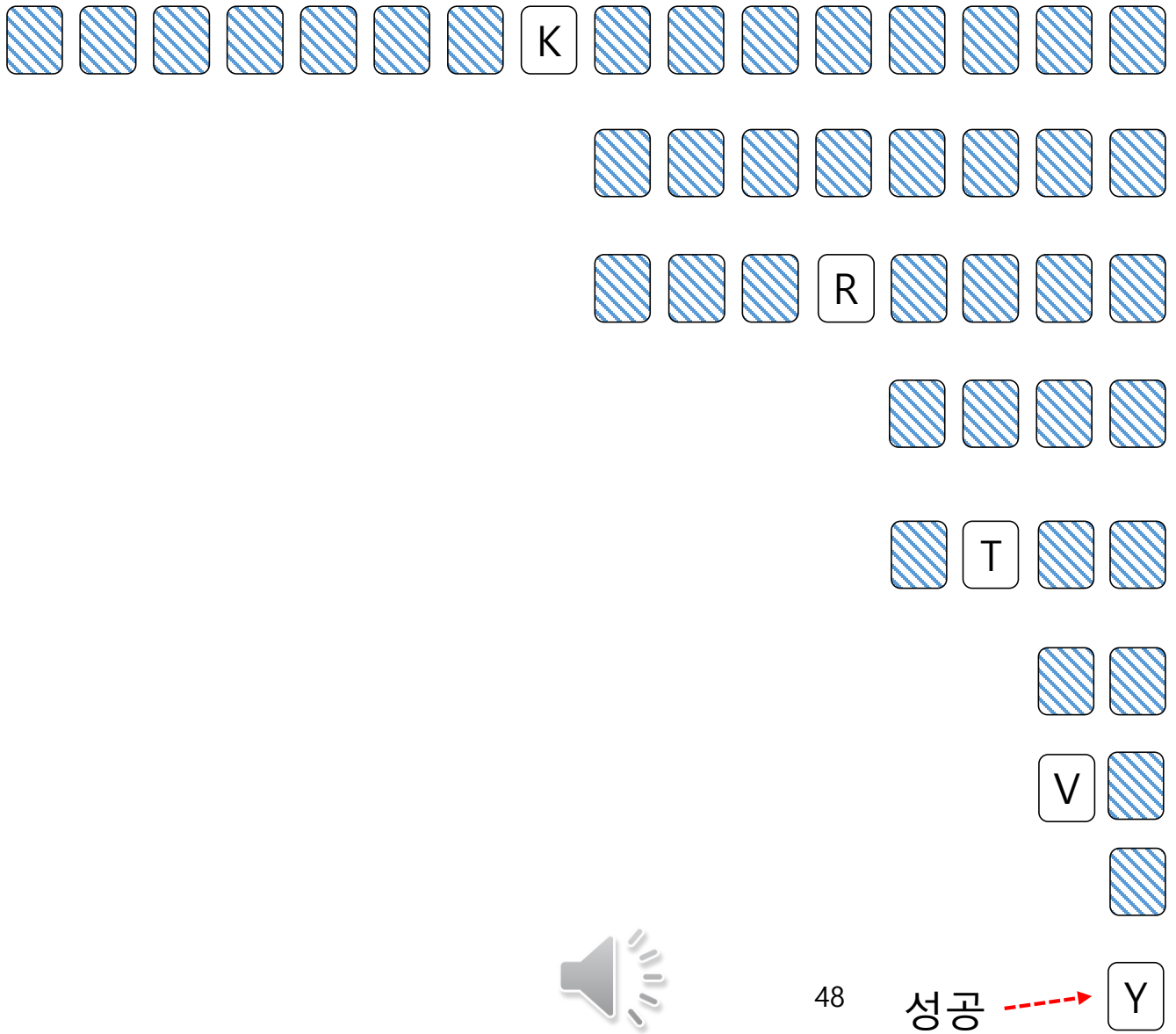
5. 중앙에 위치한 카드의 문자를 확인한다.

6. 찾는 카드가 T보다 크므로 T가 있는 카드 부터 왼쪽에 있는 모든 카드를 제거한다.

7. 중앙에 위치한 카드의 문자를 확인한다.

8. 찾는 카드가 V보다 크므로, V카드 제거한다.

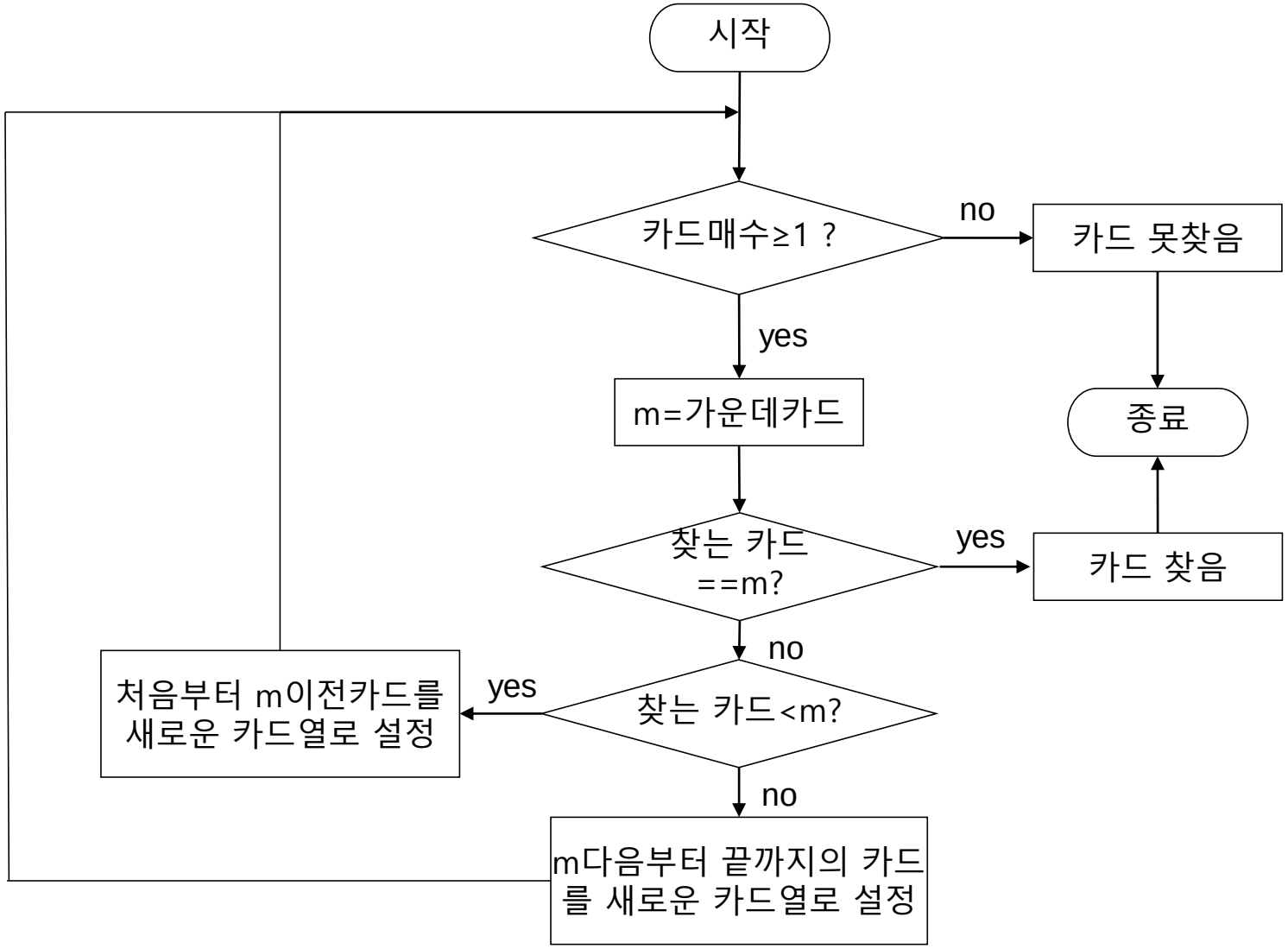
9. 남은 한 장 카드의 문자를 확인한다.



카드열에서 특정 카드를 찾는 이분검색 알고리즘

```
while (카드열의 개수가 1개 이상이고 아직 카드를 못 찾았을 때)
    m=가운데 위치한 카드
    if (찾는 카드 == m)
        카드를 찾았다.
    else if (찾는 카드 < m)
        처음부터 m 이전의 카드들을 새로운 카드열로 설정
    else
        m 다음 카드부터 끝까지의 카드들을 새로운 카드열로 설정
```

카드열에서 특정 카드를 찾는 이분검색 알고리즘

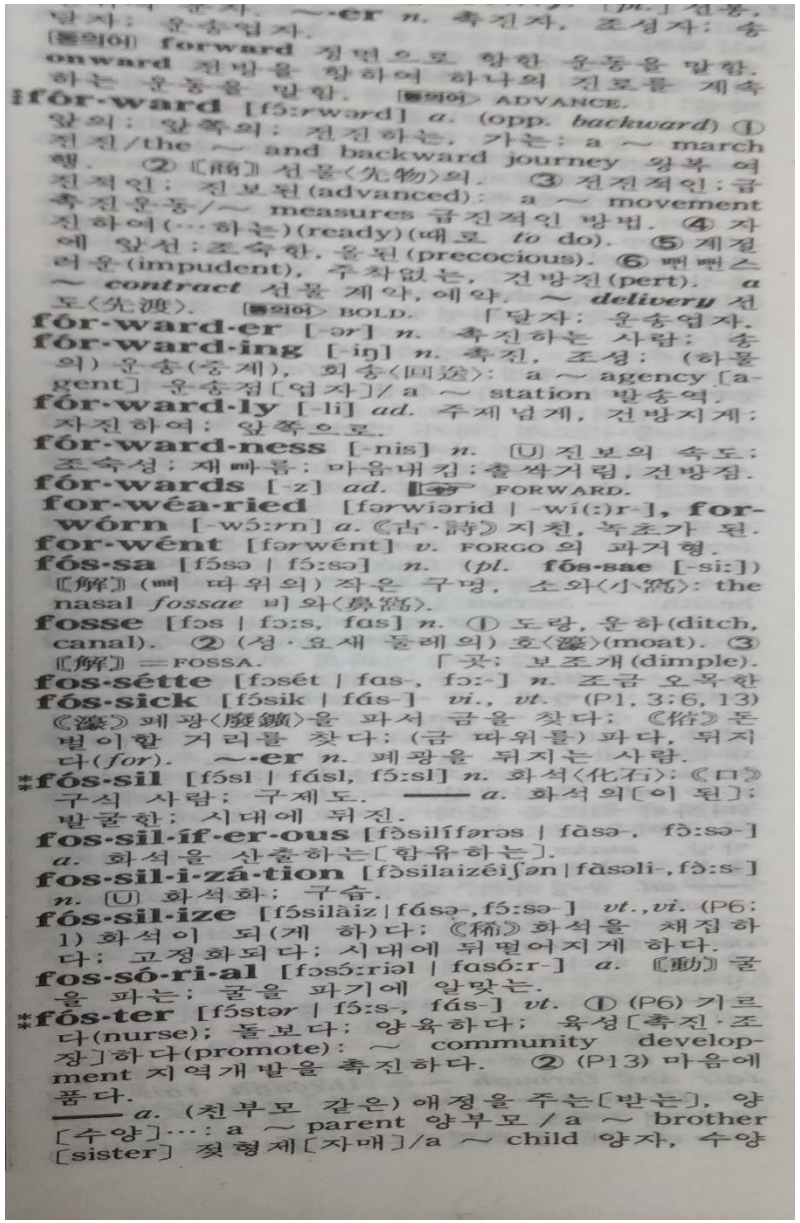


- 매 단계에서 남아 있는 카드열의 장수가 이전 단계의 카드열 장수에 비해 반(1/2)으로 줄어 든다.
- 따라서, 이 방법을 이분검색이라고 한다.
- 찾으려는 카드를 변경하면서 동일한 작업을 수행해 본다.
- 어느 경우에 카드 비교 횟수가 제일 작았나?
- 어느 경우에 카드 비교 횟수가 제일 컸나? 그 때의 비교 횟수는?
- 찾으려는 카드가 주어진 카드 열에 없을 때는 어떻게 될까?

Computational thinking

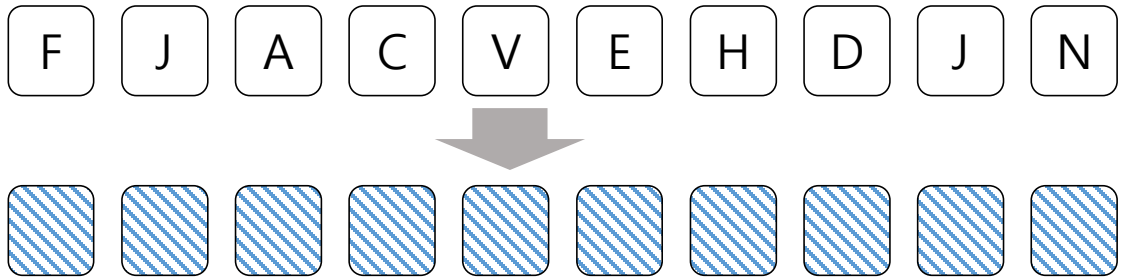
- 찾는 데이터가 있을 영역을 확인하고, 가능성이 없는 영역을 제외시킨다
- 같은 방식을 작은 문제에 동일하게 적용한다.
- 이와 같은 방식을 분할정복(divide and conquer) 알고리즘이라고 한다.(이후 설명)
- 데이터가 A B C … 순서가 아니어도, 즉 정렬되어 있지 않아도 이 방법을 적용할 수 있나?

정렬된 단어들이 나오는 사전에서 단어 찾을 때 사용 가능

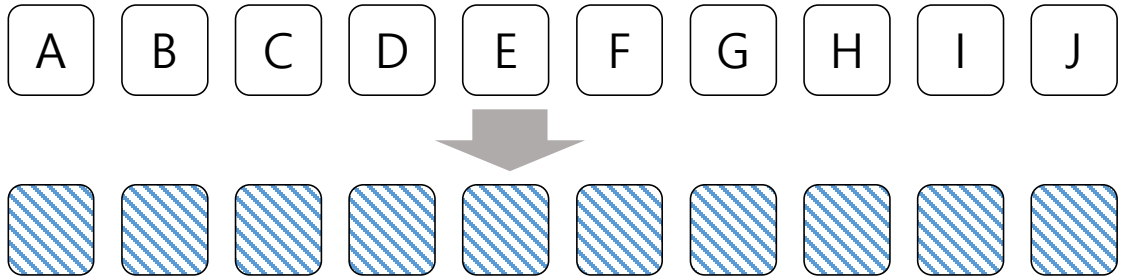


- 데이터가 정렬되어 있지 않다면 이분검색은 정상적으로 동작하지 못한다.

- 정렬되지 않은 경우



- 정렬된 경우



컴퓨터 과학

- n개의 데이터가 존재하는 자료에서 이분 검색 (binary search) 방법을 사용하여 하나의 데이터를 찾는데 필요한 데이터 비교 횟수는 최소 1 회, 최대 $\lfloor \log_2 n \rfloor + 1$
- n=16 일 때, 최대 5장 확인 필요. $\lfloor \log_2 16 \rfloor + 1$ = 4+1=5.

- $\lfloor \log_2 n \rfloor$는 마루함수(floor function)이라고 한다. $\log_2 n$을 넘지 않는 최대 정수
  (예) $\lfloor 3.5 \rfloor$ = 3

- $n=2^{10}$ 이면 최대 11번의 비교 필요
- n=대한민국 인구 대략 5,100만명. 이 데이터에서 특정 인물 자료를 찾는데 필요한 비교 횟수는 27회

색인순차검색(Indexed Sequential Search)

- 색인(인덱스, index): 사전, 전화번호부에서 특정 알파벳이 시작하는 위치를 쉽게 찾을 수 있도록 인덱스를 만들어 놓는다

색인

거리의 주소 표시

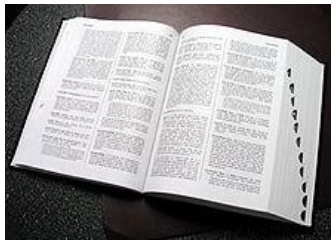
색인순차검색

- 특정 데이터를 검색할 때 먼저 색인을 확인하고, 해당 위치로 이동한 후 순차적으로 검색하는 방법

- 컴퓨터에 저장된 데이터도 색인순차검색으로 찾을 수 있다.

색인들 내에서 검색

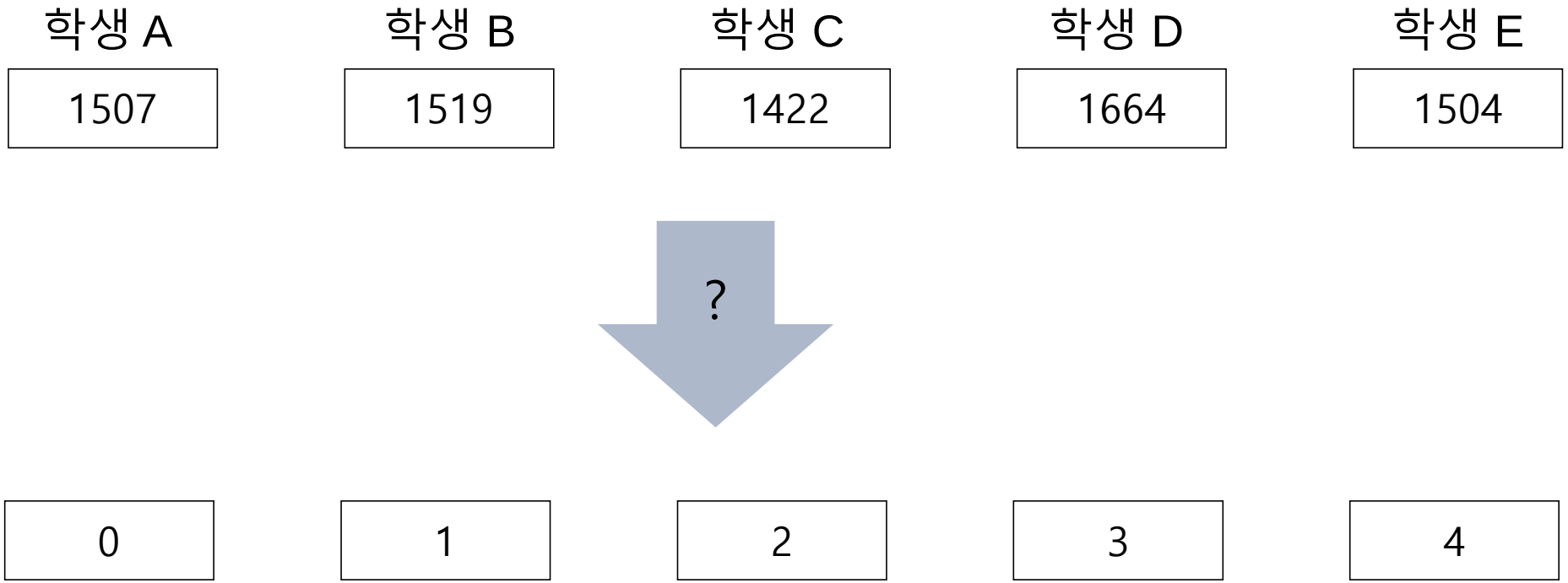
하나의 색인 내의 데이터들에 대해서 검색



해싱(Hashing)

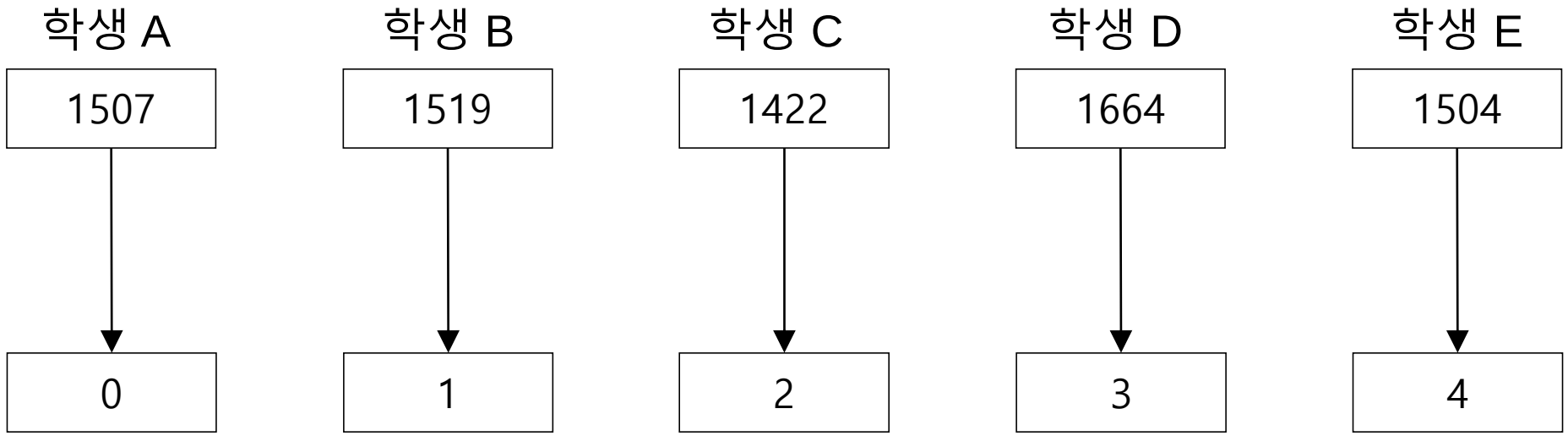
[problem]

- 5명의 학생에게 4자리의 수를 이용해서 학번을 부여했다.
- 학생을 4자리 학번으로 구분하려니, 매번 4자리 수를 표시해야 하므로 이 일을 단순화 하기로 한다
- 5명의 학생들을 0,1,2,3,4 숫자를 이용하여 학생들을 나타낼 수 있는 방법을 설계하시오.



- 학생들에게 새로운 번호를 부여하는 것은 특정 공간에 정보를 저장하는 것과 동일

단순한 방법



- 만일 현재의 순서대로 학생들을 0, 1, 2, 3, 4번 공간에 배정한다.
- 저장에 규칙이 없다
- 추후에 어떤 학생이 어느 장소에 저장되어 있는지 확인하는데 어려움이 있다.

- 알고 있는 데이터나 정보는 무엇인가?
  - ✓ 5명의 학생, 4자리 수의 학번
  - ✓ 학생들을 0, 1, 2, 3, 4 번호를 부여

- 모르고 있는 것은 무엇인가?
  - ✓ 각 학생에게 부여될 단순한 한자리 수 번호

- 조건들은 무엇인가?
  - ✓ 학생의 학번이 있으면, 그 에 해당하는 새로운 한 자리수 번호를 쉽게 알 수 있어야 한다.
  - ✓ 두 학생이 같은 새로운 번호를 갖지 않아야 한다.

- 문제 해결 과정
  1.문제를 이해한다
  2.계획을 세운다
  3.계획을 실행한다
  4.풀이과정을 재점검한다

- Computational Thinking
  - 분해
  - 패턴 확인
  - 추상화
  - 알고리즘 설계
  - 데이터 모음
  - 데이터 표현
  - 데이터 분석
  - 모의실험
  - 자동화
  - 병렬화
  - 패턴 일반화

- 문제 해결 전략
  - 추정과 확인
  - 순서 리스트 작성
  - 가능성들을 제거
  - 대칭 성질을 이용
  - 특별한 경우를 고려
  - 직접 추론 사용
  - 방정식을 푼다
  - 패턴을 찾는다
  - 그림을 그린다
  - 단순한 문제 해결 시도
  - 모델 사용
  - 역방향으로 시도
  - 공식 사용
  - 창조적 생각
  - 잔꾀를 사용

[해결 방안]

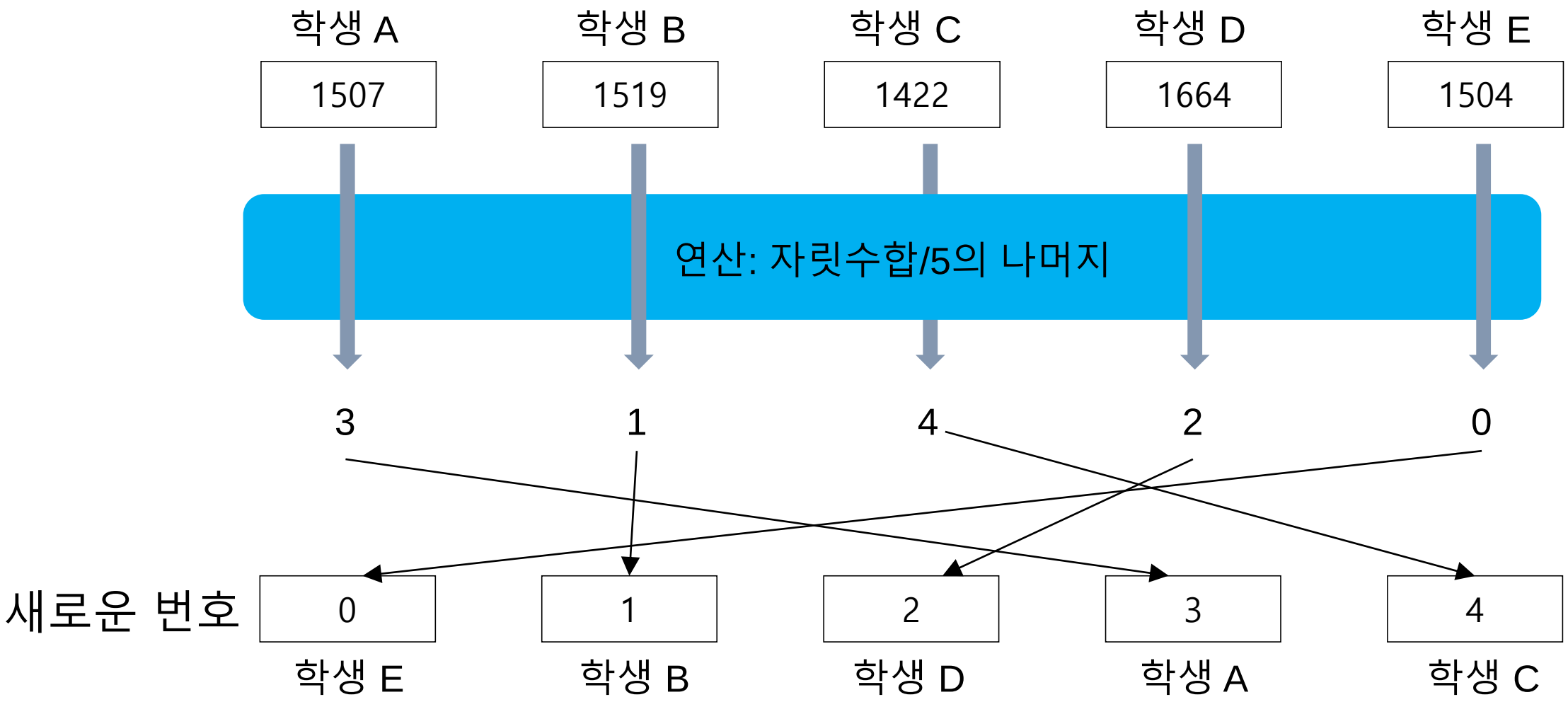
- 각 학생의 학번의 각 자릿수를 합친 후 5로 나눈다. 이 때 나머지를 학생의 새로운 구분번호로 한다.

(예) 1507 → 1+5+0+7 = 13 13/5=2와 나머지 3

새로운 번호는 3

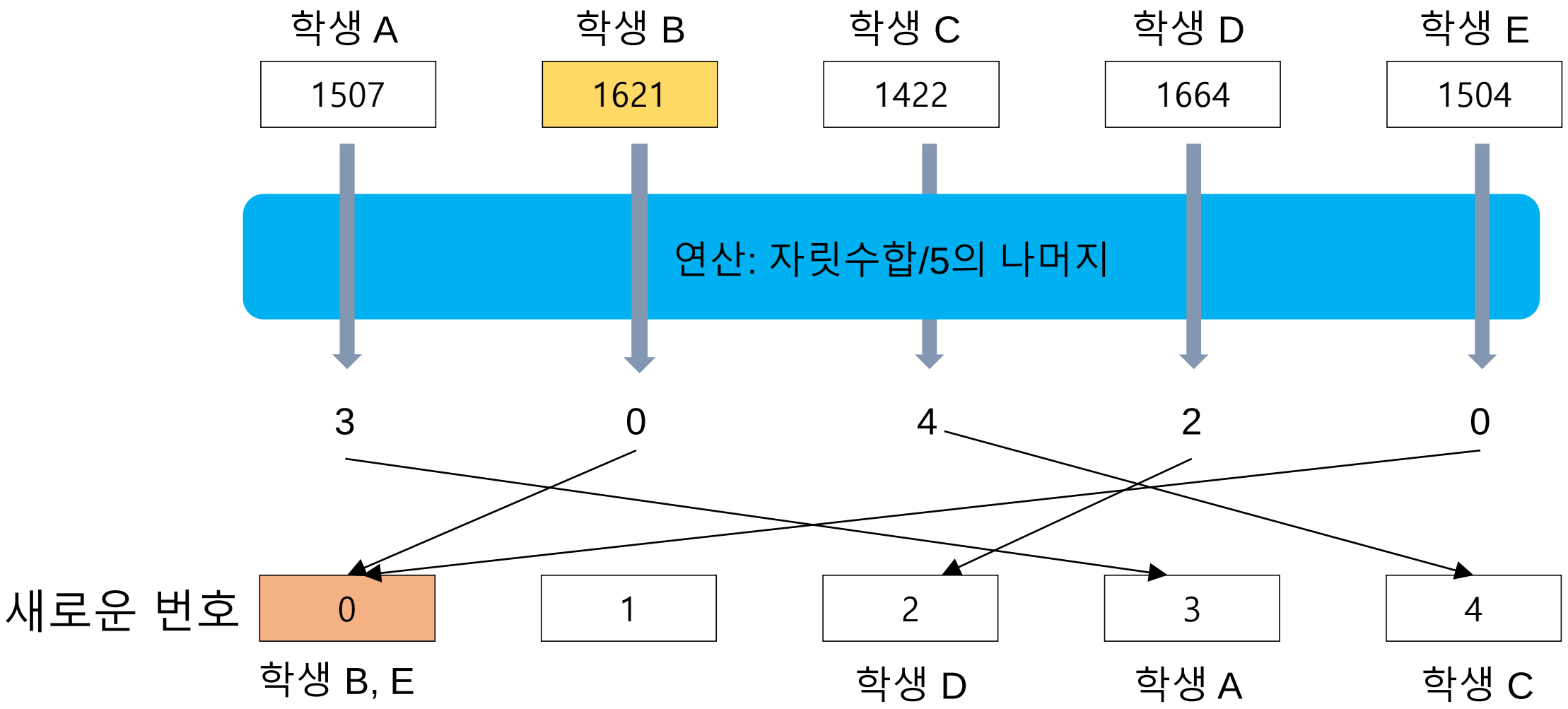
3번 칸에 저장

- 체크섬(checksum)의 일종


학생 A
학생 B
학생 C
학생 D
학생 E
1507
1519
1422
1664
1504
연산: 자릿수합/5의 나머지
3
1
4
2
0
새로운 번호
0
1
2
3
4
학생 E
학생 B
학생 D
학생 A
학생 C

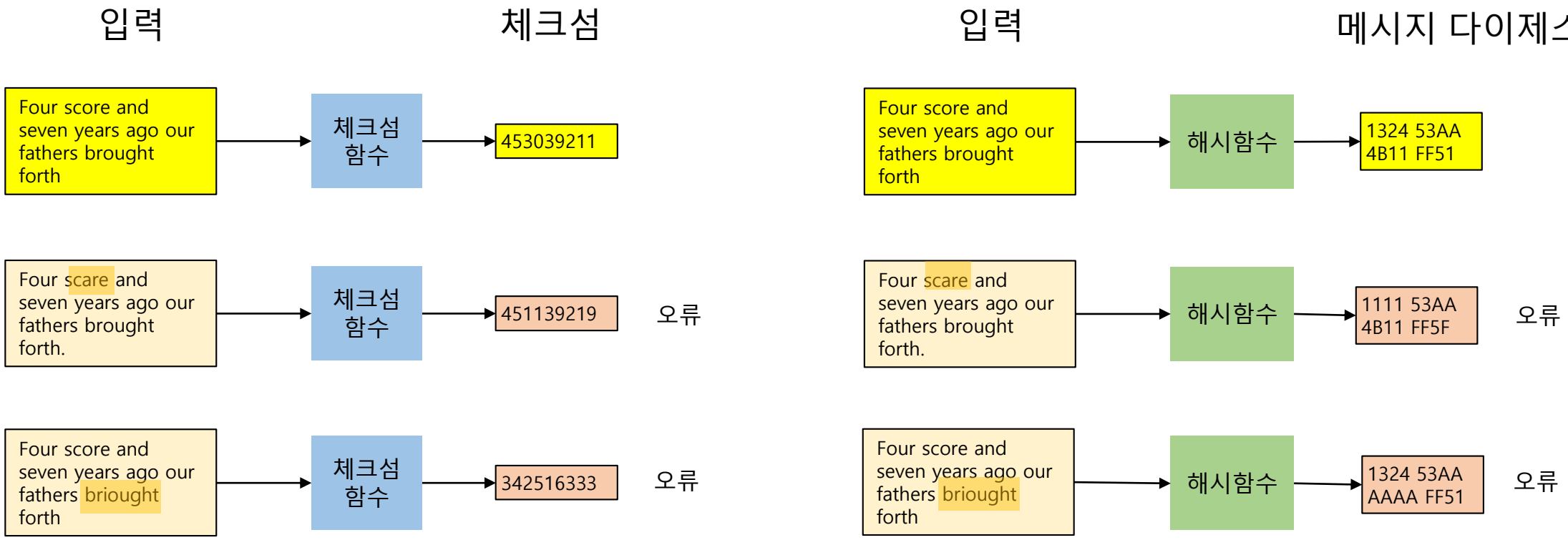
- 자릿수합/5의 나머지를 구하는 연산을 해시(hash)함수라 한다.
- 해시함수를 이용해 새로운 값을 부여하는 것을 해싱(hashing)이라고 한다.
- 다양한 해시함수가 가능하다.
- 학생의 학번만 알게 되면, 새로 부여된 번호를 쉽게 계산할 수 있다.
- 데이터의 검색이 쉽게 이루어 진다.

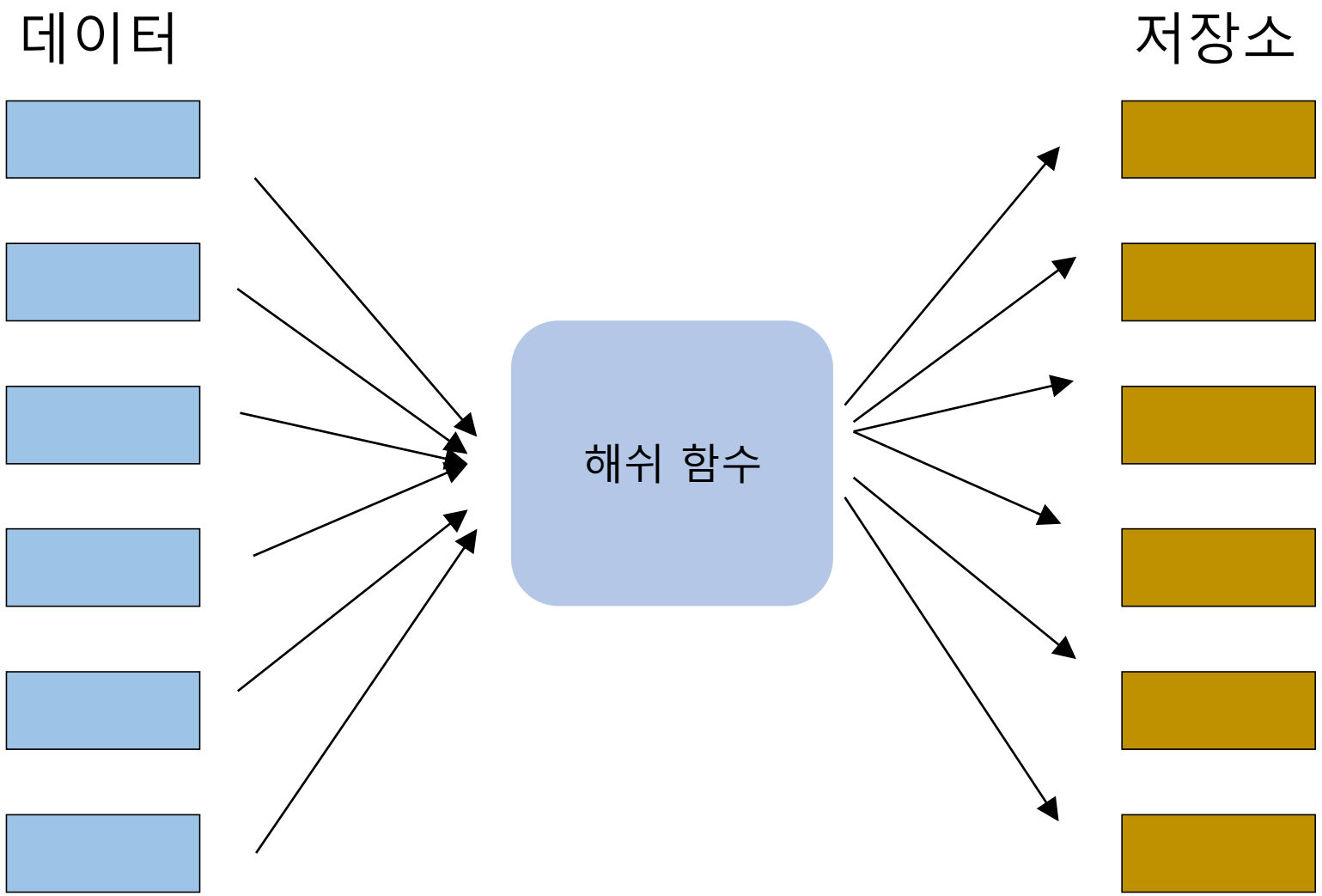
만일 학생 B의 학번이 1621 이라면,



- 2명의 학생이 동일한 새로운 번호를 갖게 된다.
- 데이터에 새로운 번호를 부여하는 체계가 유효하지 않게 된다.
- 이 경우 해시 함수가 적절하지 않으므로 새로운 해시함수를 고안해야 한다.
- 또는 충돌해결(conflict resolution) 방법이 필요

체크섬과 메시지 다이제스트는 일종의 해쉬함수를 이용한 방법





- 데이터의 검색, 추가, 삭제가 용이
- 같은 저장소에 데이터가 중복 저장되는 경우의 처리 방법 필요

Computational thinking

[풀이과정 재점검]

- 데이터를 어떻게 표현하고, 변환했는가?
- 데이터의 변환 공식을 어떻게 작성했는가?
- 5명의 학생이 있을 때 미리 10개의 새로운 번호를 준비해서, 5명의 학생에게 새로운 해시함수를 이용해서 부여하면 어떻게 될까?
- 이 때의 해시함수를 설계해 보시오.
- 10명의 사람이 있다. 이 사람들에게 0…9 까지의 새로운 번호를 부여할 수 있는 해시함수를 설계 하시오.
- 설계한 해시함수가 적절한 지 10개의 데이터를 이용해서 검증하시오.

10주차 강의 요약

- 배열 및 연결 리스트
- 선형검색
- 이분검색
- 색인순차검색
- 해싱


10주차 끝
수고하셨습니다.