

Exhaustive Search

탐색

2020/07/19

CONTENS

1. Brute-force

- Sequential Search (Linear Search)

2. Backtracking

- Maze
- N-Queen

3. Optimization Problems

- TSP

4. Divide & Conquer

- Merge Sort - Binary Search
- Power Function

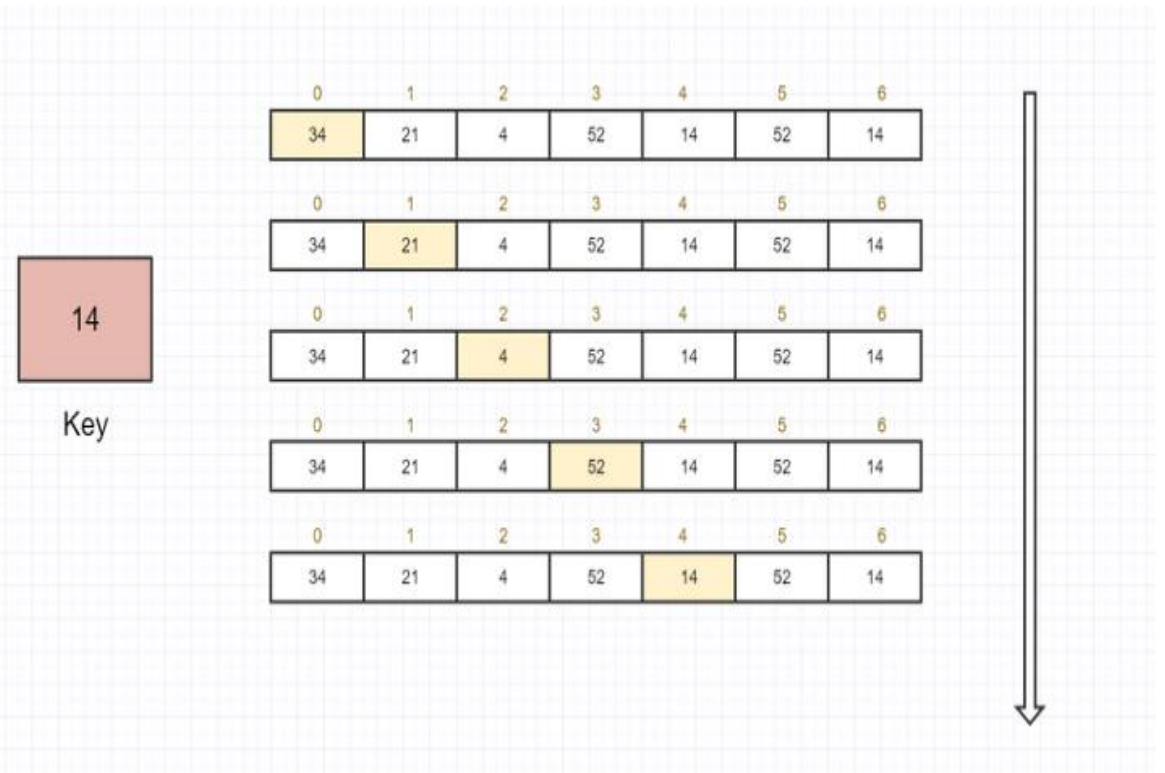
Brute-Force

■ Sequential Search

- 한 데이터에서 처음부터 끝까지 순차적으로 탐색하는 알고리즘

■ Time Complexity

- 시간 복잡도 $O(n)$
- 최악 복잡도 : 찾는 데이터가 마지막에 있거나 없을 때
- 최선 복잡도 : 제일 첫번째 있을 때
 1. 전진이동법
 2. 전위법
 3. 반도계수법

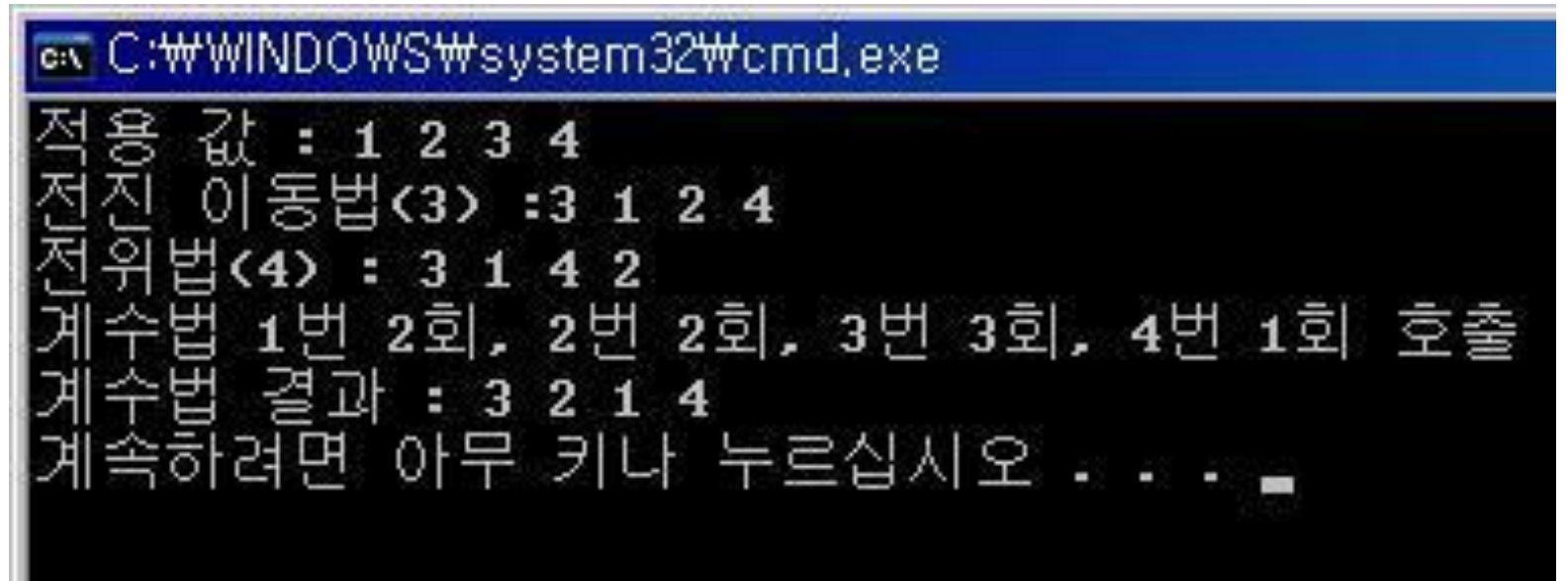


brute: 무식한, force: 힘 [완전탐색 알고리즘]

즉, 가능한 모든 경우의 수를 모두 탐색하면서 요구조건에 충족되는 결과만을 가져온다.
대표적으로 순차탐색(선형탐색), 깊이 우선 탐색, 너비 우선 탐색이 있다.

Brute-Force

1. 전진이동법 : 데이터를 맨 앞으로 이동
ex) 최근 문서, 다시 탐색될 가능성이 큰 데이터에 유용
2. 전위법 : 데이터를 한칸씩 이동
ex) 추천 메뉴, 자주 탐색되는 데이터에 유용
3. 빈도계수법 : 각 데이터가 탐색빈도로 정렬하는 것



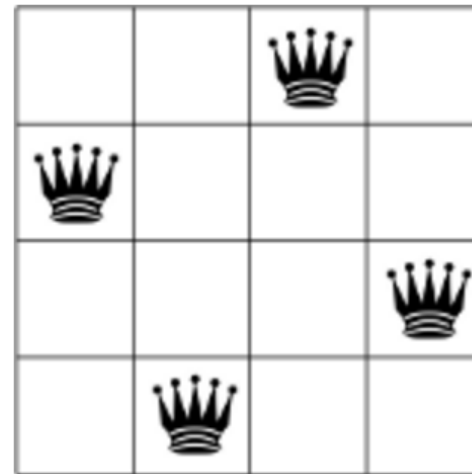
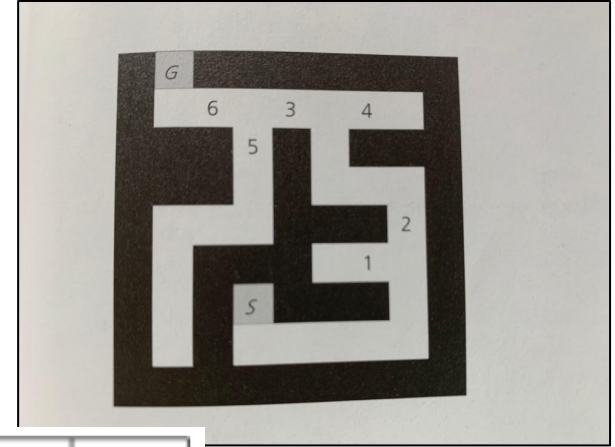
```
C:\WINDOWS\system32\cmd.exe
적용 값 : 1 2 3 4
전진 이동법<3> : 3 1 2 4
전위법<4> : 3 1 4 2
계수법 1번 2회, 2번 2회, 3번 3회, 4번 1회 호출
계수법 결과 : 3 2 1 4
계속하려면 아무 키나 누르십시오 . . .
```

Backtracking

■ Backtracking Algorithm

모든 경우의 수에서 해가 되는 경우를 찾는 알고리즘

1. 후보 해 : 모든 경우의 수, 해가 될 수 있는 후보들
ex. 미로의 모든 경로의 수 (탈출하는 경로 x)
2. 부분 해 : 후보 해의 부분 경우의 수
ex. 미로의 방향(왼쪽, 오른쪽)
3. 모든 해&최종 해 : 후보 해 중 원하는 경우들
ex. 탈출되는 경로



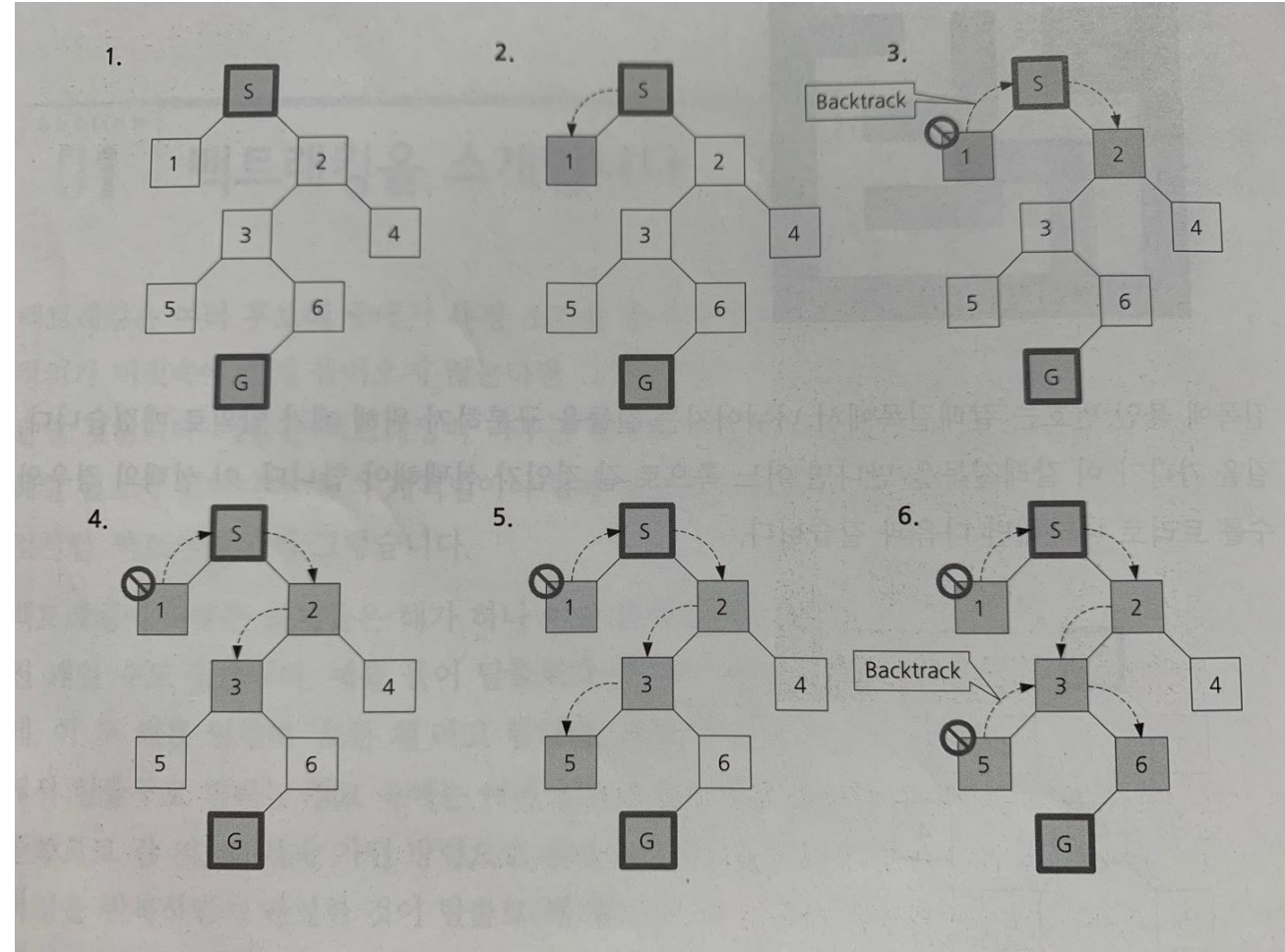
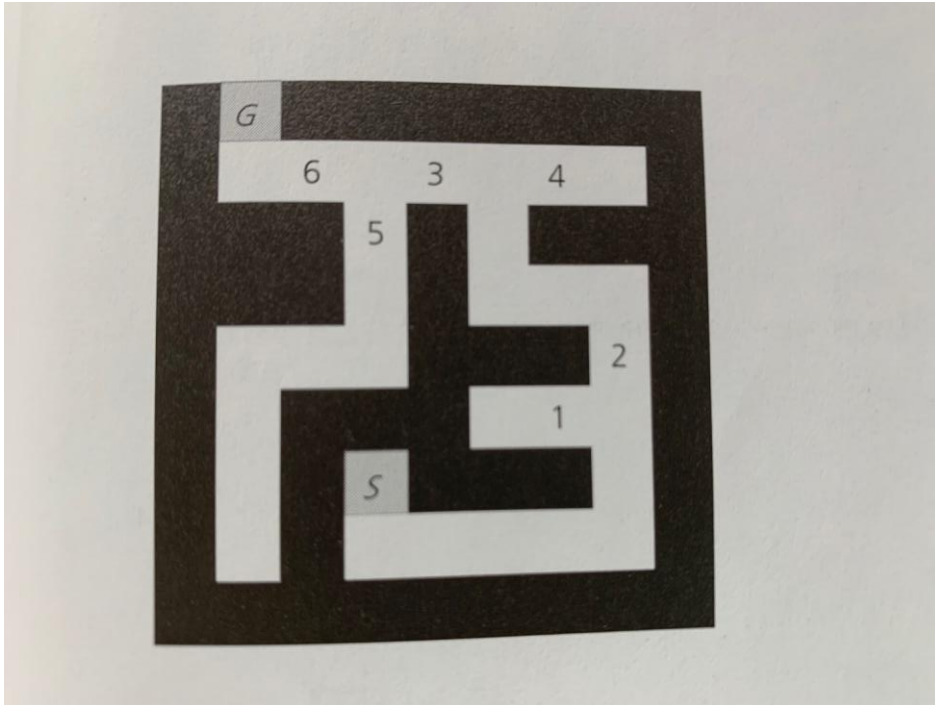
Backtrack : 되짚어 가다

후보 해를 만들어지는 과정에서 해가 될 수 없는 후보 해일 경우 전 부분 해로 돌아가서 해를 찾는 방식

Backtracking

■ 초기 설정

왼쪽부터 후보 해를 만들어간다.



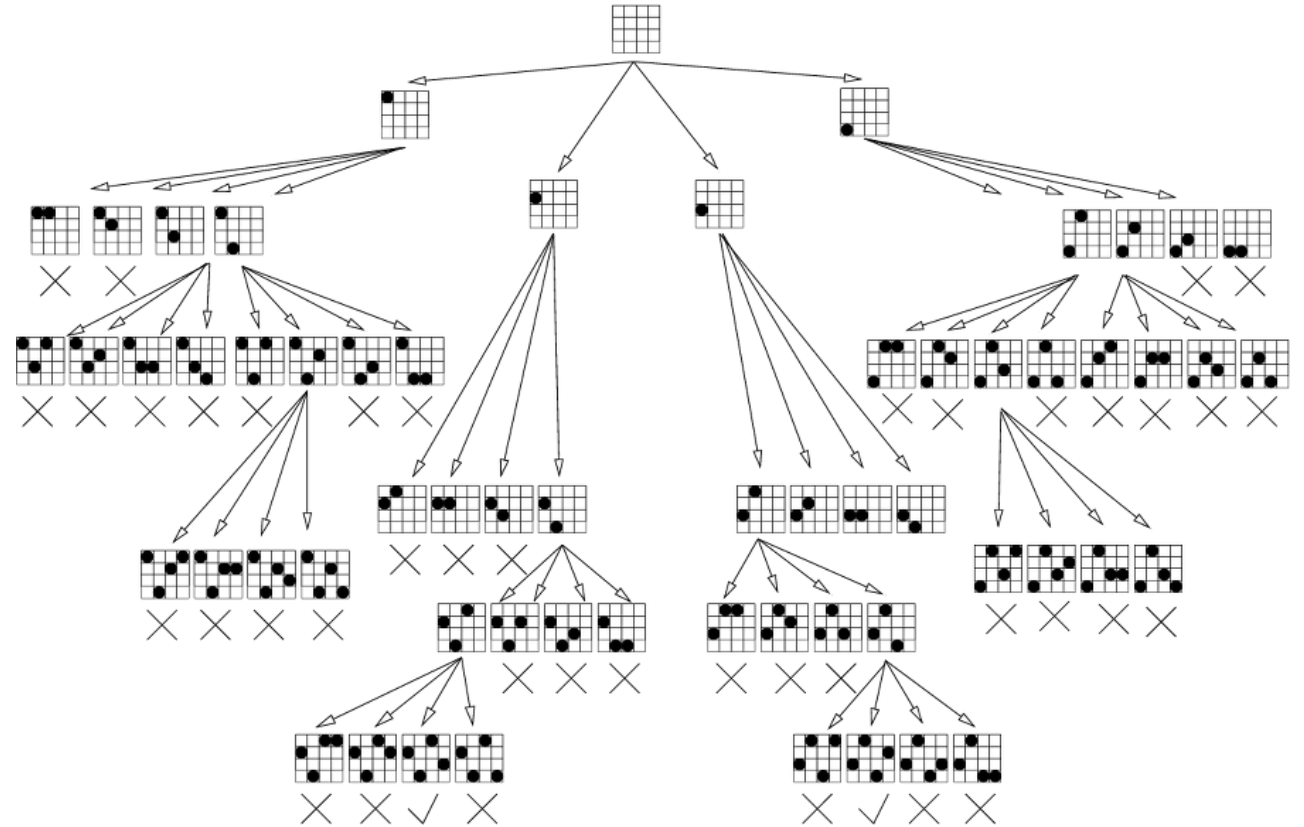
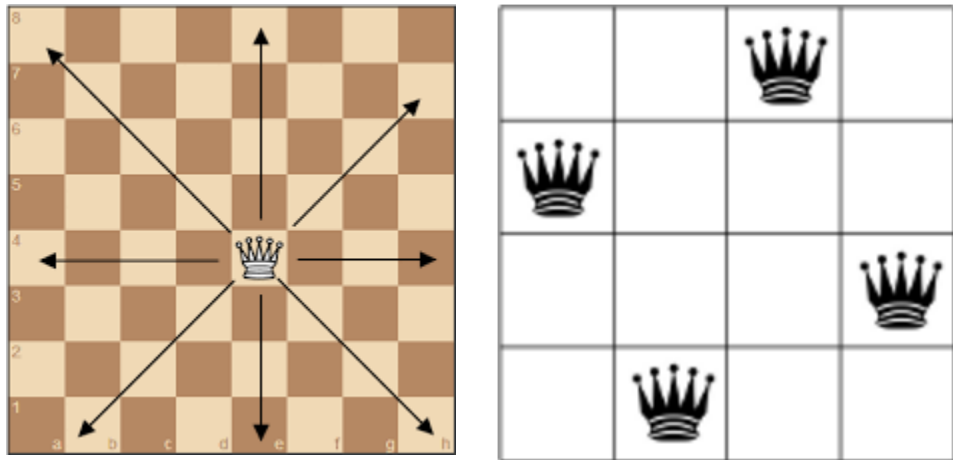
Backtrack : 되짚어 가다

후보 해를 만들어지는 과정에서 해가 될 수 없는 후보 해일 경우 다시 그 전의 부분 해로 돌아가서 해를 찾는 방식

Backtracking

■ N-Queen

N개의 퀸을 $N \times N$ 칸에서 퀸들끼리 공격을 받지 않게 배치하는 문제



Backtrack : 되짚어 가다

후보 해를 만들어지는 과정에서 해가 될 수 없는 후보 해일 경우 다시 그 전의 부분 해로 돌아가서 해를 찾는 방식

Optimization Problem

- TSP (Traveling Salesman Problem)

최단 경로 찾는 문

제

영업사원이 각 도시를 방문할 때 효율적으로 최적, 최단의 경로를 찾기 위해서 만들어진 문제

[최단 경로 알고리즘]

- 다익스트라 알고리즘



Optimization Problem : 최적화 문제

실생활에서 최단, 최적화 문제를 해결하기 위한 알고리즘

Optimization Problem

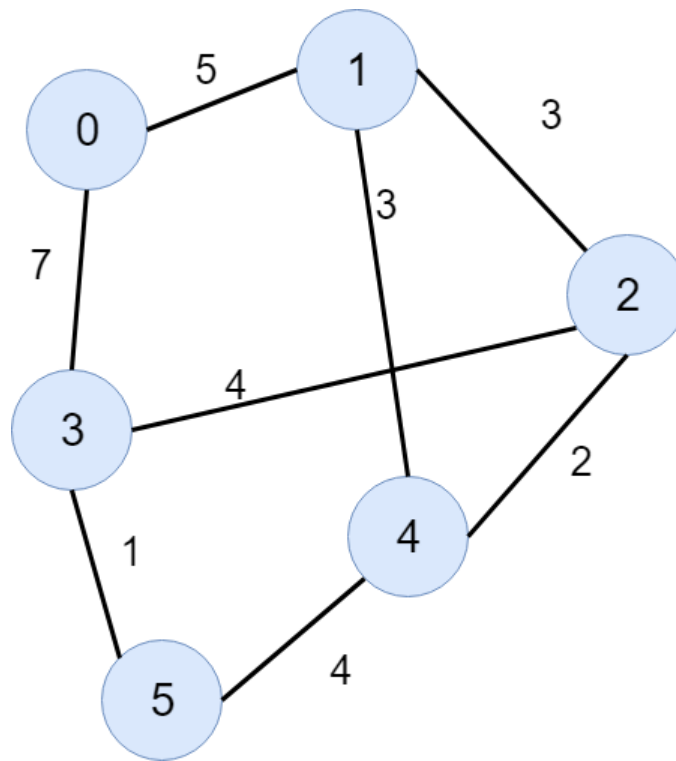
■ 다익스트라 알고리즘

정점 u 와 정점 v 를 연결하는 경로 중 간선들의 가중치 합이 최소가 되는 경로를 찾는 문제다.

간선의 가중치는 경우에 따라 비용, 거리, 시간 등으로 해석될 수 있다.

■ 1에서 5로 가는 최단 경로

경로	가중치
1 - 4 - 5	7
1 - 0 - 3 - 5	13
1 - 2 - 4 - 5	9
1 - 2 - 3 - 5	8
1 - 0 - 3 - 2 - 5	13



Optimization Problem : 최적화 문제

실생활에서 최단, 최적화 문제를 해결하기 위한 알고리즘

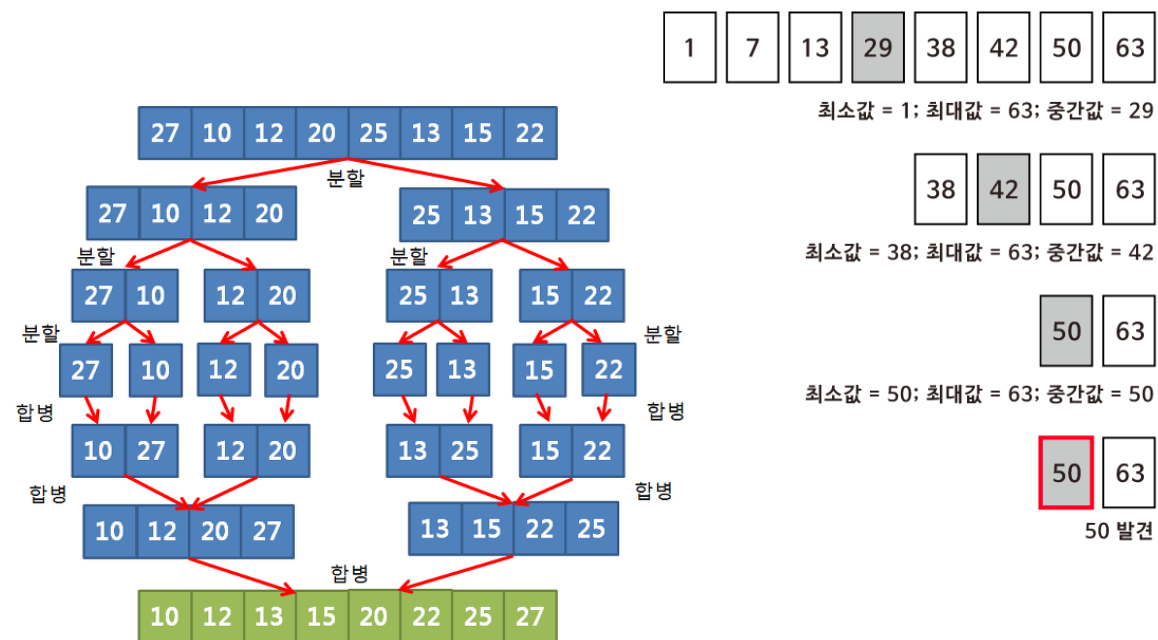
Divide & Conquer

■ 분할 & 정복의 패턴

1. 분할 : 어떤 문제의 반으로 반복하여 쪼개어 가는 것
2. 정복 : 더 이상 쪼개지지 않는 상태를 만든다.
3. 합병 : 정복된 문제들을 통합하여 문제의 답을 얻는 것

■ 대표적인 쓰임

- ✓ 이진 탐색
- ✓ 합병 정렬



$$a^n = a \times \cdots \times a$$

n

ex) $2^4 = 2 \times 2 \times 2 \times 2 = 8$

Divide : 나누다, **Conquer** : 정복하다

어떤 문제를 쪼개어지지 않을 때까지 쪼개어 전체의 답을 얻는 알고리즘

Divide & Conquer

- 이진탐색

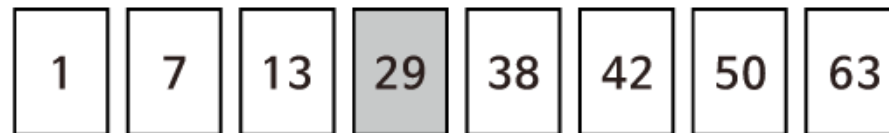
자료를 절반으로 나눈 후 찾는 값이 어느 쪽에 있는지 파악해 탐색범위를 반으로 줄여 나가는 탐색 알고리즘

- 이진탐색의 조건

데이터가 정렬된 상태에서 이진탐색 알고리즘을 사용할 수 있다.

- 성능

시간 복잡도 : $O(\log n)$



최소값 = 1; 최대값 = 63; 중간값 = 29



최소값 = 38; 최대값 = 63; 중간값 = 42



최소값 = 50; 최대값 = 63; 중간값 = 50



50 발견

Divide & Conquer

- 합병정렬

전체 원소를 하나의 단위로 **분할**한 후 분할한 원소를 다시 **병합**하는 정렬 알고리즘

- 성능

시간 복잡도 : $O(n \log n)$

