# Chapter 09.
# 그래프 탐색 알고리즘
## 핵심 유형 문제풀이

핵심 유형 문제풀이 | 다양한 문제를 접하며 코딩 테스트에 익숙해지기

강사 나동빈

# Chapter 09.
# 그래프 탐색 알고리즘

## 핵심 유형 문제풀이

A → B: https://www.acmicpc.net/problem/16953



# 혼자 힘으로 풀어보기



**문제 제목**: A → B

**문제 난이도**: ★★☆☆☆

**문제 유형**: BFS, 최단 거리

**추천 풀이 시간**: 40분

A → B: https://www.acmicpc.net/problem/16953

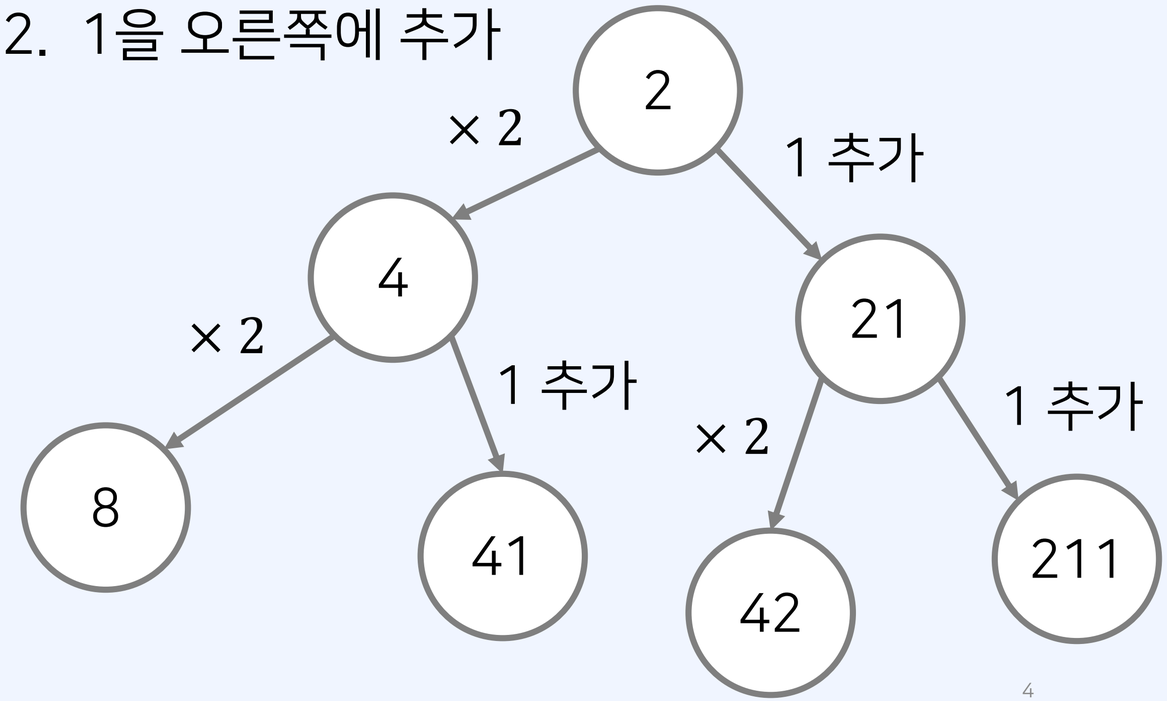


# 문제 풀이 핵심 아이디어



- 정수 $A$를 $B$로 바꾸려고 한다.
- 사용할 수 있는 연산은 **두 가지 종류**가 있으며, "**최소 연산 횟수**"를 구해야 한다.

1. 2를 곱하기
2. 1을 오른쪽에 추가



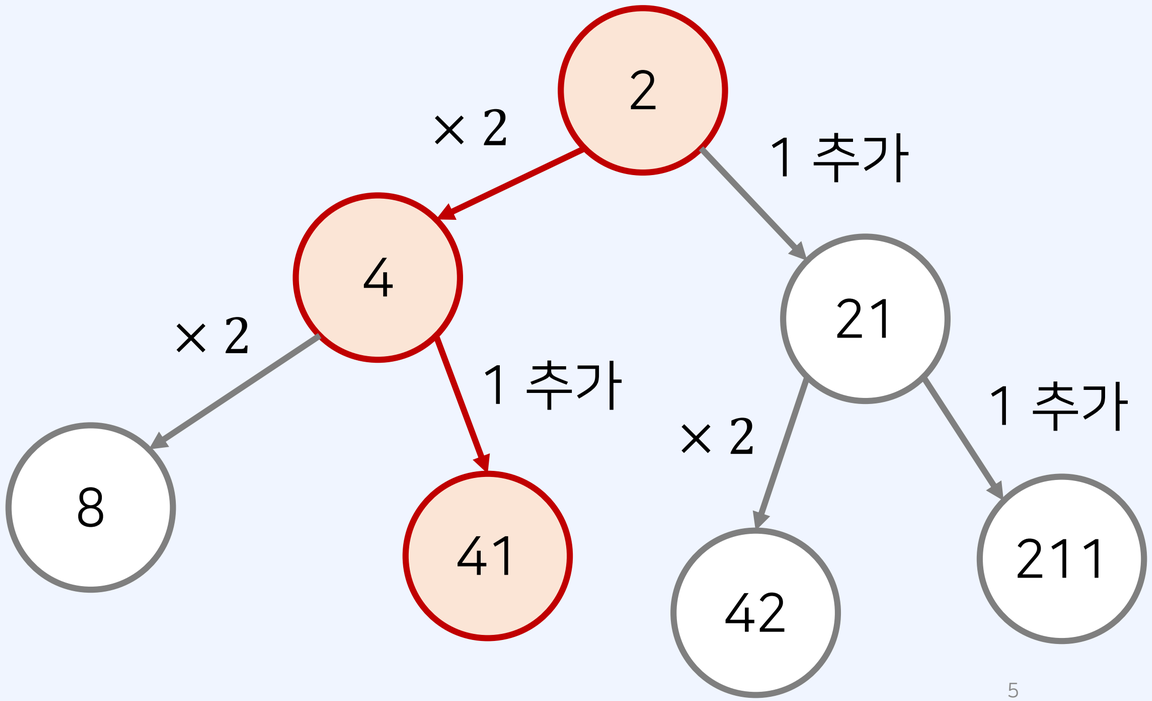
A → B: https://www.acmicpc.net/problem/16953



# 문제 풀이 핵심 아이디어



- 본 문제에서는 "필요한 연산의 최솟값"을 구해야 한다.
- 이는 다시 말하면, "최소 거리"를 구하는 문제와 동일하다.
- $A = 2, B = 41$일 때, 최소 거리는 2이다.



A → B: https://www.acmicpc.net/problem/16953



# 문제 풀이 핵심 아이디어



```python
import sys
# 빠른 입력 함수 사용
input = sys.stdin.readline
from collections import deque

# 시작(s)과 목표(t)를 입력받기
s, t = map(int, input().split())

# 너비 우선 탐색(BFS) 수행
queue = deque([])
# (값, 최소 연산 횟수) 삽입
queue.append((s, 0))
visited = set() # 방문 처리 집합(set)
found = False
# 큐가 빌 때까지 반복하기
while len(queue) != 0:
    value, dist = queue.popleft()
    if value > int(1e9): # 범위를 벗어나는 경우
        continue
    if value == t: # 목표 값에 도달한 경우
        print(dist + 1) # 최소 연산 횟수 + 1 출력
        found = True
        break
    for oper in ['*', '+']:
        next_value = value
        if oper == '*': # 2를 곱하기
            next_value *= 2
        if oper == '+': # 1을 오른쪽에 추가
            next_value *= 10
            next_value += 1
        if next_value not in visited:
            queue.append((next_value, dist + 1))
            visited.add(next_value)
# 바꿀 수 없는 경우
if not found:
    print(-1)
```

**4연산**: https://www.acmicpc.net/problem/14395



# 혼자 힘으로 풀어보기



**문제 제목**: 4연산

**문제 난이도**: ★★★☆☆

**문제 유형**: BFS, 최단 거리

**추천 풀이 시간**: 40분

**4연산**: https://www.acmicpc.net/problem/14395



# 문제 풀이 핵심 아이디어



- BFS 문제는 **최단 거리**를 찾거나 **최소 횟수**를 찾는 문제에서 많이 사용된다.
  만약에 그래프 형태로 문제가 표현된다면 **BFS**를 이용해 볼 수 있다.
- 이 문제에서는 주어진 연산이 +, −, *, /이다.
  특정한 정수 s에서 시작해서 "탐색"을 하는 형태로 간주할 수 있다.
- 따라서 값이 $t$인 노드를 만날 때까지 BFS를 수행합니다.

**4연산**: https://www.acmicpc.net/problem/14395

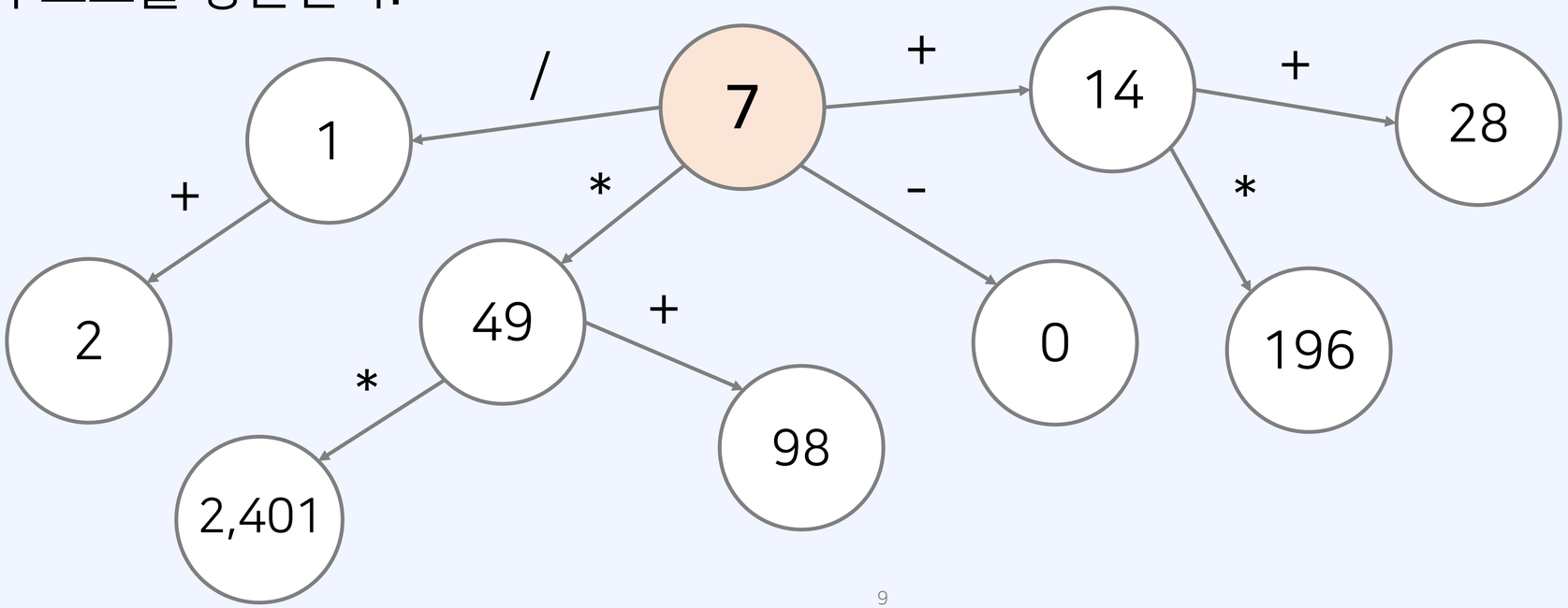


# 문제 풀이 핵심 아이디어



## [문제 해결 아이디어]

- 최소 연산 횟수를 구하는 문제이므로, **너비 우선 탐색(BFS)**을 사용한다.
- 예를 들어 $s = 7$일 때, **연산 횟수(간선 개수)**를 2까지 고려할 때 다음과 같은 형태로 각 노드를 방문한다.



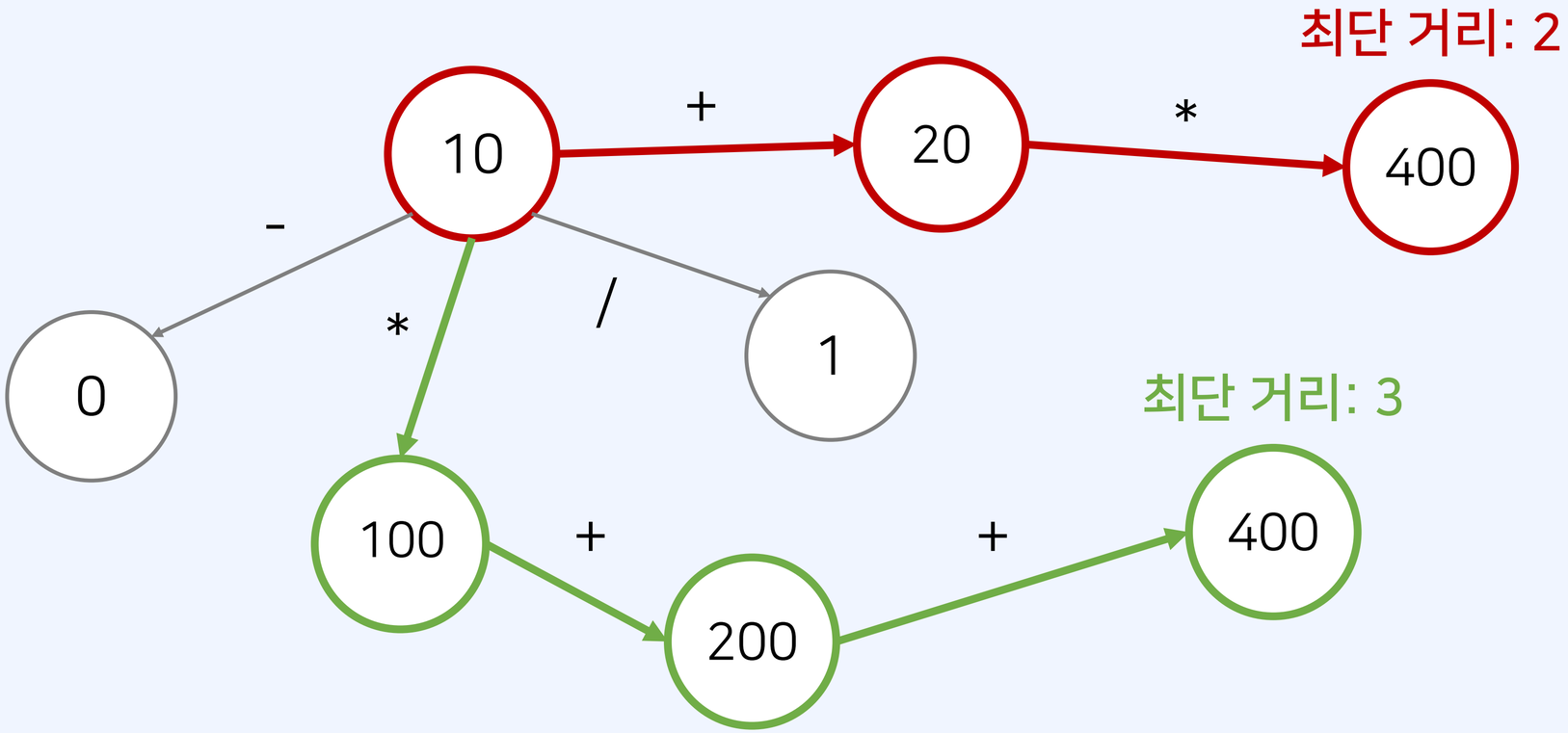
4연산: https://www.acmicpc.net/problem/14395



# 문제 풀이 핵심 아이디어



- $s = 10$에서 $t = 400$으로 가는 예시는 다음과 같다.



4연산: https://www.acmicpc.net/problem/14395



# 문제 풀이 핵심 아이디어



```python
import sys
# 빠른 입력 함수 사용
input = sys.stdin.readline
from collections import deque

# 시작(s)과 목표(t)를 입력받기
s, t = map(int, input().split())

# 너비 우선 탐색(BFS) 수행
q = deque([(0, s, '')]) # (거리, 값, 연산자) 삽입
visited = set([s]) # 방문한 값 기록
found = False

if s == t: # 시작 값과 목표 값이 같은 경우
    print(0)
    sys.exit(0)
```

```python
while q:
    dist, value, opers = q.popleft()
    if value > int(1e9): # 값의 범위를 벗어나는 경우
        continue
    if value == t: # 목표 값에 도착한 경우
        print(opers) # 전체 연산자들을 출력
        found = True
        break
    for oper in ('*', '+', '-', '/'): # 각 연산자로 BFS 수행
        if oper == '*':
            next_value = value * value
        elif oper == '+':
            next_value = value + value
        elif oper == '-':
            next_value = value - value
        elif oper == '/':
            next_value = 1
        if next_value not in visited:
            q.append((dist + 1, next_value, opers + oper))
            visited.add(next_value)

# 바꿀 수 없는 경우
if not found:
    print(-1)
```

**물통**: https://www.acmicpc.net/problem/14867



# 문제 풀이 핵심 아이디어



**문제 제목**: 물통

**문제 난이도**: ★★★☆☆

**문제 유형**: BFS, 최단 거리

**추천 풀이 시간**: 50분

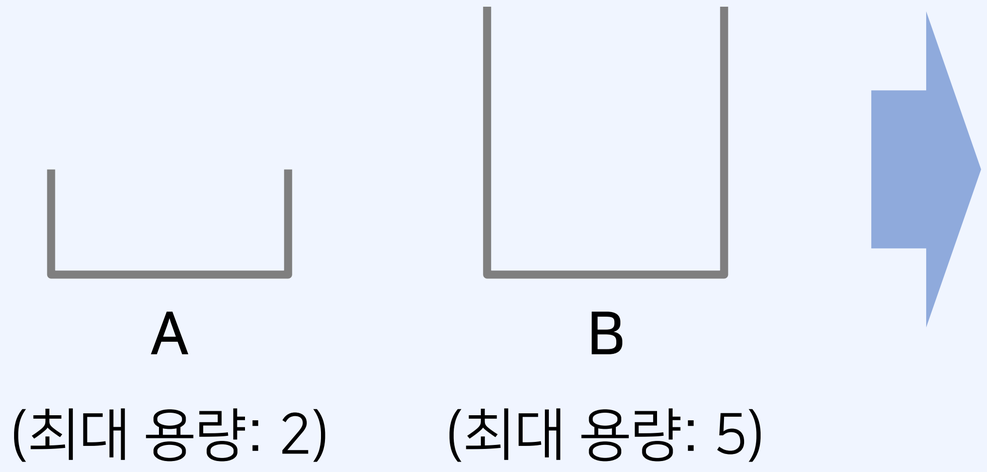
**물통**: https://www.acmicpc.net/problem/14867



# 문제 풀이 핵심 아이디어



- 문제에서의 예시를 확인하면 다음과 같다.
- **목표**: $A = 2, B = 4$




1. F(A)
2. M(A, B)
3. F(A)
4. M(A, B)
5. F(A)

**최소 작업**: 5

**물통**: https://www.acmicpc.net/problem/14867



# 문제 풀이 핵심 아이디어



- 시작 시점에서 원하는 시점까지의 최소 연산 횟수를 구하는 문제다.
- 따라서 BFS를 이용해 "최소 연산" 횟수를 계산하면 된다.
- 이 문제는 4연산 문제와 굉장히 유사한 문제다.

물통: https://www.acmicpc.net/problem/14867



# 문제 풀이 핵심 아이디어



- 특정한 **상태(state)**에서 사용할 수 있는 연산으로는 다음의 6가지 연산이 존재한다.

1. 물통 A를 비우기 (Empty A)
2. 물통 B를 비우기 (Empty B)
3. 물통 A를 채우기 (Fill A)
4. 물통 B를 채우기 (Fill B)
5. A에서 B로 물 옮기기 (A → B)
6. B에서 A로 물 옮기기 (B → A)

- 각 연산의 비용은 모두 동일하므로 **너비 우선 탐색(BFS)**으로 최소 작업 수를 계산할 수 있다.

물통: https://www.acmicpc.net/problem/14867



# 문제 풀이 핵심 아이디어



```python
from collections import deque

# (A의 최대, B의 최대, A의 목표, B의 목표)
a, b, c, d = map(int, input().split())
# (최소 연산 횟수, A 상태, B 상태) 입력
queue = deque([(0, 0, 0)])
visited = set()
found = False

# 현재의 (A 상태, B 상태)를 방문 처리
def visit(dist, nx, ny):
    if (nx, ny) not in visited:
        queue.append((dist, nx, ny))
        visited.add((nx, ny))
```

```python
while queue:
    dist, x, y = queue.popleft()
    if x == c and y == d: # 목표 상태에 도달한 경우
        print(dist)
        found = True
        break
    nx, ny = x, 0 # Empty B
    visit(dist + 1, nx, ny)
    nx, ny = 0, y # Empty A
    visit(dist + 1, nx, ny)
    nx, ny = x, b # Fill B
    visit(dist + 1, nx, ny)
    nx, ny = a, y # Fill A
    visit(dist + 1, nx, ny)
    # B에서 A로 물 옮기기
    if x + y < a: nx, ny = x + y, 0
    else: nx, ny = a, x + y - a
    visit(dist + 1, nx, ny)
    # A에서 B로 물 옮기기
    if x + y < b: nx, ny = 0, x + y
    else: nx, ny = x + y - b, b
    visit(dist + 1, nx, ny)

if not found:
    print(-1)
```