Chapter. 09

그래프 기본 탐색 알고리즘

기초 문제풀이

FAST CAMPUS
ONLINE
유형별 문제풀이

강사. 나동빈

Chapter. 09

그래프 기본 탐색 알고리즘(기초 문제풀이)

FAST CAMPUS
ONLINE
나동빈 강사.

I 혼자 힘으로 풀어 보기

DFS와 BFS: https://www.acmicpc.net/problem/1260

문제 제목: DFS와 BFS

문제 난이도: 하(Easy)

문제 유형: DFS, BFS

추천 풀이 시간: 30분

DFS와 BFS: https://www.acmicpc.net/problem/1260

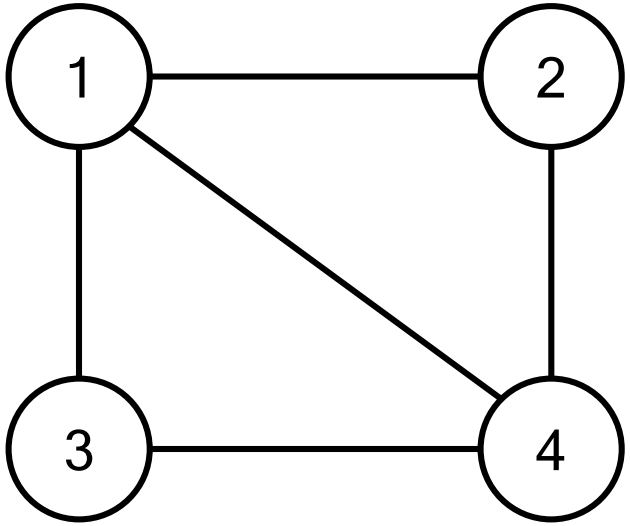
I 문제 풀이 핵심 아이디어

- 기본적인 형태의 그래프를 단순히 DFS와 BFS로 탐색합니다.
- 이 문제에서는 '정점 번호가 작은 것'을 먼저 방문해야 합니다.
- 모든 노드와 간선을 차례대로 조회하여 **O(N + M)**의 시간 복잡도로 문제를 해결해야 합니다.
- 국내 코딩 테스트 합격을 위해서는 이 문제를 매우 빠르게 풀 수 있도록 숙달할 필요가 있습니다.
- 큐(Queue) 구현을 위해 collections 라이브러리의 **deque**를 사용합니다.

DFS와 BFS: https://www.acmicpc.net/problem/1260

I 문제 풀이 핵심 아이디어

- 예시) 정점 개수 = 4, 간선 개수 = 5, 시작 정점 번호 = 1



- DFS: 1 → 2 → 4 → 3
- BFS: 1 → 2 → 3 → 4

DFS와 BFS: https://www.acmicpc.net/problem/1260

| 소스코드

```python
from collections import deque

def dfs(v):
    print(v, end=' ')
    visited[v] = True
    for e in adj[v]:
        if not(visited[e]):
            dfs(e)

def bfs(v):
    q = deque([v])
    while q:
        v = q.popleft()
        if not(visited[v]):
            visited[v] = True
            print(v, end=' ')
            for e in adj[v]:
                if not visited[e]:
                    q.append(e)
```

```python
n, m, v = map(int, input().split())
adj = [[] for _ in range(n + 1)]

for _ in range(m):
    x, y = map(int, input().split())
    adj[x].append(y)
    adj[y].append(x)

for e in adj:
    e.sort()

visited = [False] * (n + 1)
dfs(v)
print()
visited = [False] * (n + 1)
bfs(v)
```

숨바꼭질: https://www.acmicpc.net/problem/1697

혼자 힘으로 풀어 보기

문제 제목: 숨바꼭질

문제 난이도: 하(Easy)

문제 유형: BFS

추천 풀이 시간: 30분

숨바꼭질: https://www.acmicpc.net/problem/1697

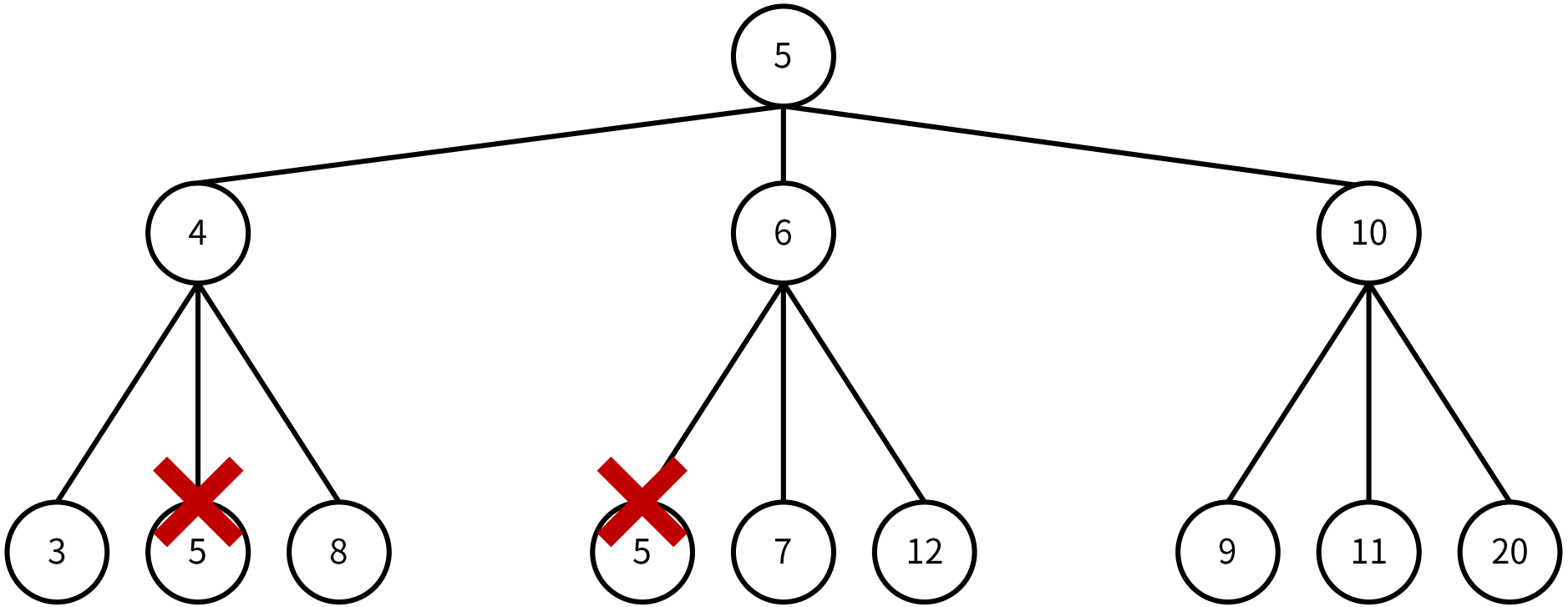
문제 풀이 핵심 아이디어

- 특정 위치까지 이동하는 최단 시간을 계산해야 하는 문제입니다.
- 이동 시간이 모두 1초로 동일하므로, 간단히 BFS를 이용하여 해결할 수 있습니다.
- 큐(Queue) 구현을 위해 collections 라이브러리의 **deque**를 사용합니다.

숨바꼭질: https://www.acmicpc.net/problem/1697

문제 풀이 핵심 아이디어

- 5에서 12로 가는 최단 시간은 다음과 같이 계산할 수 있습니다.



- BFS: 5 → 6 → 12

숨바꼭질: https://www.acmicpc.net/problem/1697

소스코드

```python
from collections import deque

MAX = 100001
n, k = map(int, input().split())
array = [0] * MAX

def bfs():
    q = deque([n])
    while q:
        now_pos = q.popleft()
        if now_pos == k:
            return array[now_pos]
        for next_pos in (now_pos - 1, now_pos + 1, now_pos * 2):
            if 0 <= next_pos < MAX and not array[next_pos]:
                array[next_pos] = array[now_pos] + 1
                q.append(next_pos)

print(bfs())
```