Chapter. 12

백트래킹

핵심 유형 문제풀이

FAST CAMPUS
ONLINE
유형별 문제풀이

강사. 나동빈

Chapter. 12

백트래킹(핵심 유형 문제풀이)

FAST CAMPUS
ONLINE
나동빈 강사.

I 혼자 힘으로 풀어 보기

N-Queen: https://www.acmicpc.net/problem/9663

문제 제목: N-Queen

문제 난이도: 중(Medium)

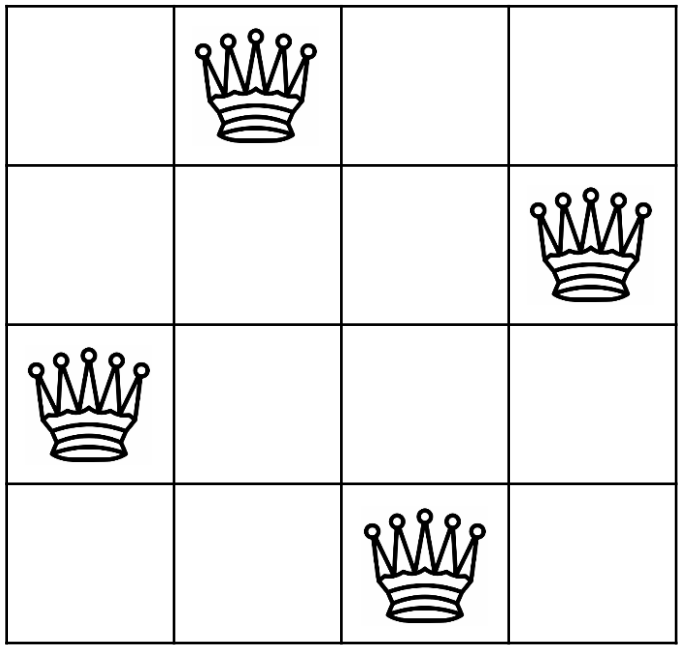
문제 유형: 백트래킹

추천 풀이 시간: 40분

N-Queen: https://www.acmicpc.net/problem/9663

I 문제 풀이 핵심 아이디어

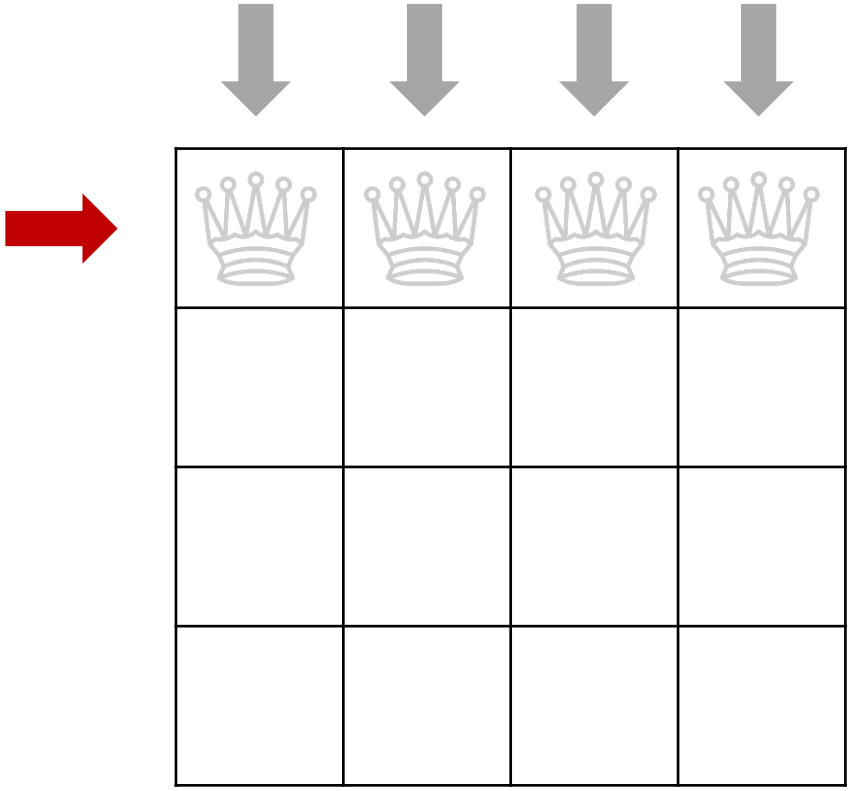
- N X N 크기의 체스 보드 위에 퀸(Queen) N개를 서로 공격할 수 없게 배치시켜야 합니다.
- 대표적인 백트래킹(Backtracking) 문제입니다.
- N = 4일 때는, 다음과 같은 경우가 존재합니다.
- DFS를 이용하여 간단히 백트래킹 알고리즘을 구현할 수 있습니다.



N-Queen: https://www.acmicpc.net/problem/9663

문제 풀이 핵심 아이디어

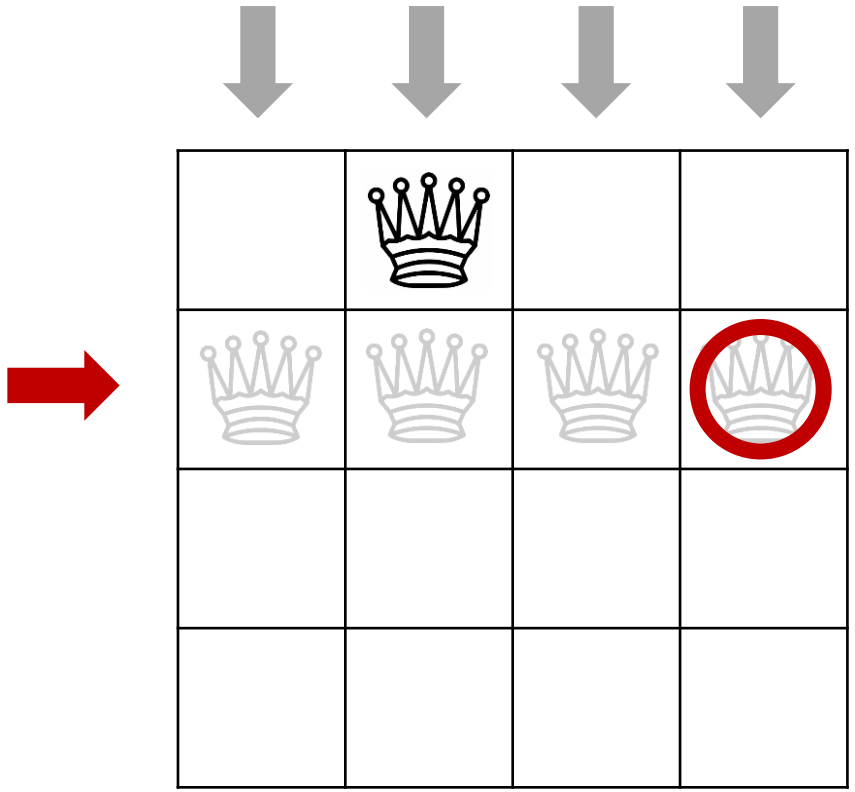
- 각 행을 차례대로 확인하면서, 각 열에 퀸(Queen)을 놓는 경우를 고려합니다.



N-Queen: https://www.acmicpc.net/problem/9663

문제 풀이 핵심 아이디어

- 각 행을 차례대로 확인하면서, 각 열에 퀸(Queen)을 놓는 경우를 고려합니다.
  - 이 때 위쪽 행을 모두 확인하며, 현재 위치에 놓을 수 있는지 확인합니다.



N-Queen: https://www.acmicpc.net/problem/9663

소스코드

```python
# x번째 행에 놓은 Queen에 대해서 검증
def check(x):
    # 이전 행에서 놓았던 모든 Queen들을 확인
    for i in range(x):
        # 위쪽 혹은 대각선을 확인
        if row[x] == row[i]:
            return False
        if abs(row[x] - row[i]) == x - i:
            return False
    return True
```

```python
# x번째 행에 대하여 처리
def dfs(x):
    global result
    if x == n:
        result += 1
    else:
        # x번째 행의 각 열에 Queen을 둔다고 가정
        for i in range(n):
            row[x] = i
            # 해당 위치에 Queen을 두어도 괜찮은 경우
            if check(x):
                # 다음 행으로 넘어가기
                dfs(x + 1)

n = int(input())
row = [0] * n
result = 0
dfs(0)
print(result)
```

I 혼자 힘으로 풀어 보기

알파벳: https://www.acmicpc.net/problem/1987

문제 제목: 알파벳

문제 난이도: 중(Medium)

문제 유형: 백트래킹

추천 풀이 시간: 40분

알파벳: https://www.acmicpc.net/problem/1987

I 문제 풀이 핵심 아이디어

- 말은 상, 하, 좌, 우 네 가지 방향으로 이동할 수 있습니다.
- 지금까지 지나온 모든 칸에 적혀 있는 알파벳과 다른 알파벳이 적힌 칸으로 이동해야 합니다.
- 행의 길이(R)와 열의 길이(C)가 20 이하이므로, 백트래킹을 이용하여 모든 경우의 수를 고려합니다.
- R = 2, C = 4일 때의 **정답** 예시는 다음과 같습니다.

C	A	A	B
A	D	C	B

알파벳 수열:

알파벳: https://www.acmicpc.net/problem/1987

I 문제 풀이 핵심 아이디어

- 말은 상, 하, 좌, 우 네 가지 방향으로 이동할 수 있습니다.
- 지금까지 지나온 모든 칸에 적혀 있는 알파벳과 다른 알파벳이 적힌 칸으로 이동해야 합니다.
- 행의 길이(R)와 열의 길이(C)가 20 이하이므로, 백트래킹을 이용하여 모든 경우의 수를 고려합니다.
- R = 2, C = 4일 때의 **정답** 예시는 다음과 같습니다.

C	A	A	B
A	D	C	B

알파벳 수열: C

알파벳: https://www.acmicpc.net/problem/1987

I 문제 풀이 핵심 아이디어

- 말은 상, 하, 좌, 우 네 가지 방향으로 이동할 수 있습니다.
- 지금까지 지나온 모든 칸에 적혀 있는 알파벳과 다른 알파벳이 적힌 칸으로 이동해야 합니다.
- 행의 길이(R)와 열의 길이(C)가 20 이하이므로, 백트래킹을 이용하여 모든 경우의 수를 고려합니다.
- R = 2, C = 4일 때의 **정답** 예시는 다음과 같습니다.

C	A	A	B
A	D	C	B

알파벳 수열: CA

알파벳: https://www.acmicpc.net/problem/1987

I 문제 풀이 핵심 아이디어

- 말은 상, 하, 좌, 우 네 가지 방향으로 이동할 수 있습니다.
- 지금까지 지나온 모든 칸에 적혀 있는 알파벳과 다른 알파벳이 적힌 칸으로 이동해야 합니다.
- 행의 길이(R)와 열의 길이(C)가 20 이하이므로, 백트래킹을 이용하여 모든 경우의 수를 고려합니다.
- R = 2, C = 4일 때의 **정답** 예시는 다음과 같습니다.

C	A	A	B
A	D	C	B

알파벳 수열: CAD

➡ 정답: 3

알파벳: https://www.acmicpc.net/problem/1987

소스코드

```
# 이동 좌표 (상, 하, 좌, 우)
dx = [-1, 1, 0, 0]
dy = [0, 0, -1, 1]

def bfs(x, y):
    global result
    # 동일한 경우는 한 번만 계산하기 위하여 집합(Set) 자료형 사용
    q = set()
    q.add((x, y, array[x][y]))

    while q:
        x, y, step = q.pop()
        # 가장 긴 이동 거리를 저장
        result = max(result, len(step))

        # 네 방향 (상, 하, 좌, 우)으로 이동하는 경우를 각각 확인
        for i in range(4):
            nx = x + dx[i]
            ny = y + dy[i]

            # 이동할 수 있는 위치이면서, 새로운 알파벳인 경우
            if (0 <= nx and nx < r and 0 <= ny and ny < c and
                array[nx][ny] not in step):
                q.add((nx, ny, step + array[nx][ny]))
```

```
# 전체 보드 데이터를 입력 받습니다.
r, c = map(int, input().split())
array = []
for _ in range(r):
    array.append(input())

# 백트래킹 수행 결과를 출력합니다.
result = 0
bfs(0, 0)
print(result)
```

혼자 힘으로 풀어 보기

암호 만들기: https://www.acmicpc.net/problem/1759

문제 제목: 암호 만들기

문제 난이도: 중(Medium)

문제 유형: 백트래킹

추천 풀이 시간: 30분

암호 만들기: https://www.acmicpc.net/problem/1759

I 문제 풀이 핵심 아이디어

- C개의 문자들이 주어졌을 때, 가능한 L 길이의 암호를 모두 찾아야 합니다.
- 따라서 C개의 문자들 중에서 L 개를 선택하는 모든 조합을 고려해야 합니다.
  - Python의 조합(combinations) 라이브러리를 사용하면 간단히 해결할 수 있습니다.
  - 혹은 DFS를 이용하여 조합 함수를 구현할 수 있습니다.

암호 만들기: https://www.acmicpc.net/problem/1759

I 문제 풀이 핵심 아이디어

- C개의 문자들이 주어졌을 때, 가능한 L 길이의 암호를 모두 찾아야 합니다.
- 따라서 C개의 문자들 중에서 L 개를 선택하는 모든 조합을 고려해야 합니다.
- C = 4, L = 3인 예시는 다음과 같습니다.

주어진 문자들: k n a b

[가능한 모든 조합]

a b k
a b n
a k n
b k n

정답: **3**

소스코드 ①

암호 만들기: https://www.acmicpc.net/problem/1759

```python
from itertools import combinations

vowels = ('a', 'e', 'i', 'o', 'u')
l, c = map(int, input().split(' '))

# 가능한 암호를 사전식으로 출력해야 하므로 정렬 수행
array = input().split(' ')
array.sort()

# 길이가 l인 모든 암호 조합을 확인
for password in combinations(array, l):
    # 모음의 개수를 세기
    count = 0
    for i in password:
        if i in vowels:
            count += 1

    # 최소 한 개의 모음과 최소 두 개의 자음이 있는 경우 출력
    if count >= 1 and count <= l - 2:
        print(''.join(password))
```

소스코드 ②

암호 만들기: https://www.acmicpc.net/problem/1759

```python
import copy

result = []
string = []
visited = []

# 조합(Combination) 함수 구현
def combination(array, length, index):
    # 길이가 length인 모든 조합 찾기
    if len(string) == length:
        result.append(copy.deepcopy(string))
        return
    # 각 원소를 한 번씩만 뽑도록 구성
    for i in range(index, len(array)):
        if i in visited:
            continue
        string.append(array[i])
        visited.append(i)
        combination(array, length, i + 1)
        string.pop()
        visited.pop()
```

```python
vowels = ('a', 'e', 'i', 'o', 'u')
l, c = map(int, input().split(' '))

# 가능한 암호를 사전식으로 출력해야 하므로 정렬 수행
array = input().split(' ')
array.sort()

combination(array, l, 0)

# 길이가 l인 모든 암호 조합을 확인
for password in result:
    # 모음의 개수를 세기
    count = 0
    for i in password:
        if i in vowels:
            count += 1

    # 최소 한 개의 모음과 최소 두 개의 자음이 있는 경우 출력
    if count >= 1 and count <= l - 2:
        print(''.join(password))
```