SSAFY 12/12

최백준 choi@startlink.io


BAE/KJOON
ONLINE JUDGE


STARTLINK

코딩 테스트

코딩 테스트

Coding Test

- 다양한 기업에서 코딩 테스트를 보고 있다

코딩 테스트

Coding Test

- 주로 파싱, 시뮬레이션, 브루트 포스, BFS, 다이나믹 프로그래밍이 문제에 나온다.

코딩 테스트

Coding Test

- 약 2-3시간 동안 1-2문제를 시험 본다

코딩 테스트

Coding Test

기업	문제 수	합격 컷	시간	언어
삼성	2	1-2	3	C/C++/Java/Python
하이닉스 1차	3	1	2	C/C++
하이닉스 2차	3	1-2	2	C/C++
한화	10	4-5	2	C/C++/Java
카카오 1차	4	3.5	5	C/C++/Java/Python
카카오 2차	1	1	5	모든 것이 자유 (인터넷 가능)
라인 1차	4	2	2	C/C++/Java/Python
라인 2차	4	?	2	C/C++/Java/Python
NHN	4	1	2	C/C++/Java
11번가	4	3	2	C/C++/Java/그 외… (SQL 문제 있음)
우아한형제들 1차	5	4	4	C/C++
우아한형제들 2차	3	2	4	C/C++/Java/Python
넥슨	5	4	2	다양함
KT	2	?	0.5	손
마이다스 아이티	6	3	3	C/C++/Java/Python/Scala

코딩 테스트

Coding Test

- 코딩 테스트를 보는 이유는 알고리즘 문제를 잘 푸는 사람을 뽑는 것이 목적이 아님

알고리즘

Algorithm

- 알고리즘(라틴어, 독일어: Algorithmus, 영어: algorithm 알고리듬, IPA: [ælgərìðm])은 수학과 컴퓨터 과학, 언어학 또는 관련 분야에서 **어떠한 문제를 해결하기 위해 정해진 일련의 절차나 방법을 공식화한 형태로 표현한 것**을 말한다.
- 출처: https://ko.wikipedia.org/wiki/알고리즘

알고리즘

Algorithm

- 많은 개발은 어떠한 문제를 해결해야 하는 것이 목적인 경우가 많다.

알고리즘

Algorithm

- 알고리즘 문제는 상대적으로 대부분의 개발보다 크기가 작다.
- 문제도 다른 분야의 특정 문제를 알고리즘 문제 스타일로 변형하는 경우가 많다.
- 문제를 모델링하고 해결하는 능력을 알아보기 위해서 알고리즘 문제를 푼다.

알고리즘

Algorithm

- 매우 많은 알고리즘과 자료구조가 있다.

알고리즘

Algorithm

- Dynamic Programming, Dijkstra Algorithm, Floyd-Warshall Algorithm, Greedy Algorithm, Kruskal’s Algorithm, Prim’s Algorithm, Divide & Conquer, Network Flow, Ford-Fulkerson Algorithm, Edmond-Karp Algorithm, BFS, DFS, Heuristic, Tree, Graph, Stack, Queue, Binary Search Tree, Binary Indexed Tree, Binary Search, Segment Tree, GCD, LCM, Quick Sort, Merge Sort, Eratosthenes Sieve, Heap, Adjacency Matrix, Adjacency List, Brute Force, Disjoint Set, Hash, Backtracking, Bitmask, Topological Sort, LCA, Meet in the Middle, Sqrt Decomposition, Min-cost Max-flow, Tarjan’s Algorithm, Kosaju’s Algorithm, Strongly Connected Component, BCC, Suffix Array, KMP, Trie, 2-SAT, CCW, Graham Scan, Sprague-Grundy Function

알고리즘

Algorithm

- 알고리즘의 분류와 문제의 난이도는 크게 상관이 없다.
- 모든 브루트 포스 문제는 모든 다이나믹 문제보다 쉽다 (X)
- 모든 다이나믹 문제는 모든 브루트 포스 문제보다 쉽다 (X)
- 이 브루트 포스 문제는 이 다이나믹 문제보다 쉽다 (O)
- 이 다이나믹 문제는 이 브루트 포스 문제보다 쉽다 (O)

코딩 테스트

Coding Test

- 주로 파싱, 시뮬레이션, 브루트 포스, BFS, 다이나믹 프로그래밍이 문제에 나온다.
- 개발과 크게 상관이 있거나, 난이도가 쉬운 문제들이 많은 알고리즘, 학교에서 주로 배우는 알고리즘이 이유라 생각

코딩 테스트 방식

코딩 테스트 방식

Coding Test

- 오프라인 시험: 지정된 장소에 모여서, 컴퓨터로 문제를 품
- 온라인 시험: 각자 알아서 컴퓨터로 문제를 품
- 손 코딩: 지정된 장소에 모여서, 손으로 코드를 작성함
- 코딩 인터뷰: 면접관과 같이 문제를 의논하면서, 손으로 코드를 작성함

CBT

Coding Test

- 컴퓨터를 이용해 직접 코딩을 하는 방식
- 오프라인과 온라인의 차이는 크지 않음
- 사용하는 플랫폼에 따른 문제의 차이는 없음
- 플랫폼은 그렇게 중요하지 않고, 문제의 유형이 중요함
- 문제는 최종 형태이며, 주어진 문제 본문만 가지고 문제를 해결해야 함

손 코딩

Coding Test

- 필기 시험과 같은 형태로, 답안 코드를 손으로 작성해야 함
- Java의 경우에는 자동 완성이 없어서 매우 귀찮음

코딩 인터뷰

Coding Test

- 주로 불완전한 문제를 제시한다.
- 빠진 조건이 있다면 면접관과 얘기하면서 문제를 완성시켜 나가야 함
- 문제를 얼마나 잘 푸는지 보기보다는 문제를 어떤 과정으로 푸는지가 더 중요함
- 문제를 제시하자마자 프로그래밍 대회처럼 빠르게 해결하는걸 원하지는 않음
- 손으로 코딩해야 하는데, 컴파일 에러는 걱정할 필요가 없고, 의미만 전달되면 됨

코딩 테스트 방식

Coding Test

- 입/출력을 받고 전체 코드를 작성하는 방식인지, 함수를 구현하는 방식인지는 난이도와 상관이 없음
- 테스트 케이스 방식인지 아닌지도 난이도와 상관 없음

알고리즘 문제

Coding Test

- 알고리즘 문제는 주로 제목, 문제 본문, 입력 형식 설명, 출력 형식 설명, 예제 입/출력, 힌트, 시간 제한, 메모리 제한으로 이루어져 있음
- 가장 먼저 읽으면 좋은 것은 N제한 (입력의 크기)

알고리즘 문제

Coding Test

- 알고리즘 문제는 주로 제목, 문제 본문, 입력 형식 설명, 출력 형식 설명, 예제 입/출력, 힌트로 이루어져 있음
- 가끔 힌트로 어떻게 예제의 정답이 나오는지 설명하는 경우가 있는데, 이 때 설명하기 위한 방법은 문제의 풀이 방법이 높은 확률로 아님

알고리즘 문제

Coding Test

- 시간 제한과 메모리 제한
- 무엇이 더 중요할까?

알고리즘 문제

Coding Test

- 시간 제한이 훨씬 더 중요하다.

알고리즘 문제

Coding Test

- 메모리 초과가 나는 코드는 시간 초과도 날 확률이 매우 높다.

코딩 테스트 준비

코딩 테스트 준비

Coding Test

- 게임을 잘하고 싶다. 어떻게 해야 할까?

코딩 테스트 준비

Coding Test

- 게임을 잘하고 싶다. 어떻게 해야 할까?
- 게임을 많이 해야 한다.

코딩 테스트 준비

Coding Test

- 게임을 잘하고 싶다. 어떻게 해야 할까?
- 게임을 많이 해야 한다.
- 물론, 게임을 무작정 많이 하면 되는 것은 아니다.

코딩 테스트 준비

Coding Test

- 코딩을 잘하고 싶다. 어떻게 해야 할까?

코딩 테스트 준비

Coding Test

- 코딩을 잘하고 싶다. 어떻게 해야 할까?
- 코딩을 많이 해야 한다.

코딩 테스트 준비

Coding Test

- 코딩을 잘하고 싶다. 어떻게 해야 할까?
- 코딩을 많이 해야 한다.
- 물론, 코딩을 무작정 많이 하면 되는 것은 아니다.

알고리즘 공부

Coding Test

- 어떤 알고리즘의 원리와 증명을 이해하는 것이 중요하다.

알고리즘 공부

Coding Test

- 어떤 알고리즘의 원리와 증명을 이해하는 것이 중요하지는 않다.
- 원리와 증명은 매우 중요하기는 하나, 코딩 테스트에서는 응용하는 것을 더 중요하게 생각한다.

알고리즘 공부

Coding Test

- 어떤 문제를 보고 도움 없이 스스로 방법을 떠올려야 한다.

알고리즘 공부

Coding Test

- 어떤 문제를 보고 도움 없이 스스로 방법을 떠올려야 할 필요는 없다.
- 예를 들어, 수학 공부를 할 때 피타고라스의 정리, 미분, 적분을 스스로 떠올리는 것을 요구하지 않는다.

알고리즘 공부

Coding Test

- 이 문제를 어떻게 푸는지 모르겠으니 어떻게든 스스로 풀어내고 말겠다.

알고리즘 공부

Coding Test

- 이 문제를 어떻게 푸는지 모르겠으니 어떻게든 스스로 풀어내는 것이 나쁜 생각은 아니다.

알고리즘 공부

Coding Test

- 이 문제를 어떻게 푸는지 모르겠으니 어떻게든 스스로 풀어내는 것이 나쁜 생각은 아니다.
- 하지만, 인생은 짧고, 시간은 없고, 할 일은 많다.

알고리즘 공부

Coding Test

- 이 문제를 어떻게 푸는지 모르겠으니 어떻게든 스스로 풀어내는 것이 나쁜 생각은 아니다.
- 하지만, 인생은 짧고, 시간은 없고, 할 일은 많다.
- 개발은 모든 것을 스스로 해결하는 것을 요구하지 않는다.
- 도움을 받는 것과 이해하는 것, 검색을 하는 것도 매우 중요한 능력이다.

알고리즘 공부

Coding Test

- 이 문제를 어떻게 푸는지 모르겠으니 어떻게든 스스로 풀어내는 것이 나쁜 생각은 아니다.
- 하지만, 인생은 짧고, 시간은 없고, 할 일은 많다.
- 개발은 모든 것을 스스로 해결하는 것을 요구하지 않는다.
- 도움을 받는 것과 이해하는 것, 검색을 하는 것도 매우 중요한 능력이다.
- 조금 고민해보고 모르겠으면, 답을 보고 이해하자.

알고리즘 공부

Coding Test

- 나는 이해는 못하겠으니, 문제의 답을 외워야겠다.

알고리즘 공부

Coding Test

- 나는 이해는 못하겠으니, 문제의 답을 외워야겠다.
- 코딩 테스트엔 내가 안 외운 문제가 나온다.

알고리즘 공부

Coding Test

- 기출 문제를 열심히 풀어보고, 기출 문제와 비슷한 내용이 나오는 문제를 풀어봐야겠다.

알고리즘 공부

Coding Test

- 기출 문제를 열심히 풀어보고, 기출 문제와 비슷한 내용이 나오는 문제를 풀어봐야겠다.
- 기출 문제와 비슷한 내용이 나오는 문제는 별로 안 중요하고, 비슷한 알고리즘을 사용하는 문제를 풀어보는 것이 좋다.
- 참고로, 기출 문제를 풀고, 답을 외우고 하는 것 보다는, 어떻게 문제를 풀어내는지 과정을 알고있는 것이 더 중요하다.

알고리즘 공부

Coding Test

- 나는 BFS도 알고, 브루트 포스도 알고, 다이나믹 프로그래밍도 안다. 그런데 문제는 못 풀겠다.
- 그러니까 이 문제를 푸는 알고리즘이 무엇인지는 어떻게 알아내는 것인가?

알고리즘 공부

Coding Test

- 나는 BFS도 알고, 브루트 포스도 알고, 다이나믹 프로그래밍도 안다. 그런데 문제는 못 풀겠다.
- 그러니까 **이 문제를 푸는 알고리즘이 무엇인지**는 어떻게 알아내는 것인가?
- 이것이 요구하는 능력이다.
- 따로 법칙은 없기 때문에, 각각의 알고리즘의 특징을 알고, 왜 그 알고리즘으로 다른 문제를 풀 수 있었는지를 위주로 기억해서 문제에 적용해봐야 한다.

알고리즘 공부

Coding Test

- 먼저, 각각의 알고리즘 문제를 풀어보면서, 알고리즘을 익히는 것이 좋다.

알고리즘 공부 준비

Coding Test

SW Expert Academy

알고리즘 공부 준비

Coding Test

BAE/KJOON

ONLINE JUDGE

알고리즘 공부 준비

Coding Test

코딩 테스트 준비

Coding Test

- SW 역량 테스트 문제 분석: https://startl.ink/30g80nL

브루트 포스

브루트 포스

Brute Force

- 브루트 포스는 모든 경우의 수를 다 해보는 것이다.

브루트 포스

Brute Force

- 예를 들어, 비밀번호가 4자리이고, 숫자로만 이루어져 있다고 한다면
- 0000부터 9999까지 다 입력해보면 된다.
- 경우의 수가 10,000가지 이다.

브루트 포스

Brute Force

- 예를 들어, 비밀번호가 4자리이고, 숫자로만 이루어져 있다고 한다면
- 0000부터 9999까지 다 입력해보면 된다.
- 경우의 수가 10,000가지 이다.
- 사람이 직접 비밀번호를 입력하는데 1초가 걸린다면 10,000초 = 2.7시간 정도 걸린다.

브루트 포스

Brute Force

- 예를 들어, 비밀번호가 12자리이고, 숫자로만 이루어져 있다고 한다면
- 000000000000부터 999999999999까지 다 입력해보면 된다.
- 경우의 수가 1,000,000,000,000가지 이다.

브루트 포스

Brute Force

- 예를 들어, 비밀번호가 12자리이고, 숫자로만 이루어져 있다고 한다면
- 000000000000부터 999999999999까지 다 입력해보면 된다.
- 경우의 수가 1,000,000,000,000가지 이다.
- 사람이 직접 비밀번호를 입력하는데 1초가 걸린다면 1,000,000,000,000초 = 약 31688년이 걸린다.

브루트 포스

Brute Force

- 브루트 포스는 모든 경우의 수를 다 해보는 것이다.
- 이 때, 경우의 수를 다 해보는데 걸리는 시간이 문제의 시간 제한을 넘지 않아야 한다.

브루트 포스

Brute Force

- 브루트 포스로 문제를 풀기 위해서는 다음과 같은 3가지 단계를 생각해볼 수 있다.

1. 문제의 가능한 경우의 수를 계산해본다.
2. 가능한 모든 방법을 다 만들어본다.
3. 각각의 방법을 이용해 답을 구해본다.

브루트 포스

Brute Force

- 브루트 포스로 문제를 풀기 위해서는 다음과 같은 3가지 단계를 생각해볼 수 있다.

1. 문제의 가능한 경우의 수를 계산해본다.
    - 직접 계산을 통해서 구한다. 대부분 손으로 계산해볼 수 있다.
2. 가능한 모든 방법을 다 만들어본다.
3. 각각의 방법을 이용해 답을 구해본다.

브루트 포스

Brute Force

- 브루트 포스로 문제를 풀기 위해서는 다음과 같은 3가지 단계를 생각해볼 수 있다.

1. 문제의 가능한 경우의 수를 계산해본다.
    - 직접 계산을 통해서 구한다. 대부분 손으로 계산해볼 수 있다.
2. 가능한 모든 방법을 다 만들어본다.
    - 하나도 빠짐 없이 만들어야 한다.
    - 대표적으로 그냥 다 해보는 방법, for문 사용, 순열 사용, 재귀 호출 사용, 비트마스크 사용이 있다.
3. 각각의 방법을 이용해 답을 구해본다.

브루트 포스

Brute Force

- 브루트 포스로 문제를 풀기 위해서는 다음과 같은 3가지 단계를 생각해볼 수 있다.

1. 문제의 가능한 경우의 수를 계산해본다.
    - 직접 계산을 통해서 구한다. 대부분 손으로 계산해볼 수 있다.
2. 가능한 모든 방법을 다 만들어본다.
    - 하나도 빠짐 없이 만들어야 한다.
    - 대표적으로 그냥 다 해보는 방법, for문 사용, 순열 사용, 재귀 호출 사용, 비트마스크 사용이 있다.
3. 각각의 방법을 이용해 답을 구해본다.
    - 이 단계는 보통은 어렵지 않다. 문제에 나와있는 대로 답을 계산해본다.

브루트 포스

Brute Force

- 브루트 포스로 문제를 풀기 위해서는 다음과 같은 3가지 단계를 생각해볼 수 있다.

1. 문제의 가능한 경우의 수를 계산해본다.
    - 직접 계산을 통해서 구한다. 대부분 손으로 계산해볼 수 있다.
2. 가능한 모든 방법을 다 만들어본다.
    - 하나도 빠짐 없이 만들어야 한다.
    - 대표적으로 그냥 다 해보는 방법, for문 사용, 순열 사용, 재귀 호출 사용, 비트마스크 사용이 있다.
3. 각각의 방법을 이용해 답을 구해본다.
    - 이 단계는 보통은 어렵지 않다. 문제에 나와있는 대로 답을 계산해본다.

- 브루트 포스 문제의 시간 복잡도는 대부분 O(경우의 수 * 방법 1개를 시도해보는데 걸리는 시간 복잡도)가 걸린다.

경우의 수

경우의 수

Brute Force

- 문제의 가능한 경우의 수를 계산하기 위해 몇 가지 경우의 수를 계산하는 방법을 연습해보자

경우의 수

Brute Force

- N명의 사람이 한 줄로 서는 경우의 수 →
- N명의 사람 중에서 대표 두 명을 뽑는 경우의 수 →
- N명의 사람 중에서 대표 세 명을 뽑는 경우의 수 →
- N명의 사람 중에서 반장 1명과 부반장 1명을 뽑는 경우의 수 →
- N명의 사람이 있을 때, 각 사람이 영화를 볼지, 보지 않을지 결정한다. 가능한 조합의 수 →

경우의 수

Brute Force

- N명의 사람이 한 줄로 서는 경우의 수 → $N \times (N-1) \times \cdots \times 1 = N!$
- N명의 사람 중에서 대표 두 명을 뽑는 경우의 수 → $N \times (N-1) / 2$
- N명의 사람 중에서 대표 세 명을 뽑는 경우의 수 → $N \times (N-1) \times (N-2) / 3!$
- N명의 사람 중에서 반장 1명과 부반장 1명을 뽑는 경우의 수 → $N \times (N-1)$
- N명의 사람이 있을 때, 각 사람이 영화를 볼지, 보지 않을지 결정한다. 가능한 조합의 수 → 2^N

N과 M

N과 M (1)

https://www.acmicpc.net/problem/15649

- 1부터 N까지 자연수 중에서 중복 없이 M개를 고른 수열을 모두 구하는 문제
- $1 \leq M \leq N \leq 8$

N과 M (1)

https://www.acmicpc.net/problem/15649

```
bool c[10]; int a[10];
void go(int index, int n, int m) {
    if (index == m) {
        // 수열을 출력
        return;
    }
    for (int i=1; i<=n; i++) {
        if (c[i]) continue;
        c[i] = true; a[index] = i;
        go(index+1, n, m);
        c[i] = false;
    }
}
// go(0, n, m);
```

N과 M (1)

https://www.acmicpc.net/problem/15649

- 소스: http://codeplus.codes/ba90562cd98b4b4e9162e59a487c2d68

N과 M (2)

https://www.acmicpc.net/problem/15650

- 1부터 N까지 자연수 중에서 중복 없이 M개를 고른 수열을 모두 구하는 문제 (오름차순)
- $1 \leq M \leq N \leq 8$

N과 M (2)

https://www.acmicpc.net/problem/15650

```
bool c[10]; int a[10];
void go(int index, int start, int n, int m) {
    if (index == m) {
        // 수열을 출력
        return;
    }
    for (int i=start; i<=n; i++) {
        if (c[i]) continue;
        c[i] = true; a[index] = i;
        go(index+1, i+1, n, m);
        c[i] = false;
    }
}
// go(0, 1, n, m);
```

N과 M (2)

https://www.acmicpc.net/problem/15650

- 소스: http://codeplus.codes/49792b1bb002438786b68ea546ecb10c

N과 M (2)

https://www.acmicpc.net/problem/15650

- 1부터 N까지 자연수 중에서 중복 없이 M개를 고른 수열을 모두 구하는 문제 (오름차순)
- $1 \leq M \leq N \leq 8$
- 오름차순만 고르는 것이기 때문에, 다른 방식도 가능하다.
- 각각의 자연수를 선택하는 경우와 선택하지 않는 경우가 있다.

N과 M (2)

https://www.acmicpc.net/problem/15650

```
int a[10];
void go(int index, int selected, int n, int m) {
    if (selected == m) {
        // 수열 출력
        return;
    }
    if (index > n) return;
    a[selected] = index;
    go(index+1, selected+1, n, m);
    a[selected] = 0;
    go(index+1, selected, n, m);
}
// go(1, 0, n, m);
```

N과 M (2)

https://www.acmicpc.net/problem/15650

- 소스: http://codeplus.codes/c0631db97cac4ebe8413d2e55b95e09b

N과 M (3)

https://www.acmicpc.net/problem/15651

- 1부터 N까지 자연수 중에서 M개를 고른 수열을 모두 구하는 문제 (중복 선택 가능)
- $1 \leq M \leq N \leq 7$

N과 M (3)

https://www.acmicpc.net/problem/15651

```
bool c[10]; int a[10];
void go(int index, int n, int m) {
    if (index == m) {
        // 수열을 출력
        return;
    }
    for (int i=1; i<=n; i++) {
        //if (c[i]) continue;
        c[i] = true; a[index] = i;
        go(index+1, n, m);
        c[i] = false;
    }
}
// go(0, n, m);
```

N과 M (3)

https://www.acmicpc.net/problem/15651

- 소스: http://codeplus.codes/4c8fd01bd9764810bc53695358f71f56

N과 M (4)

https://www.acmicpc.net/problem/15652

- 1부터 N까지 자연수 중에서 M개를 고른 수열을 모두 구하는 문제 (중복 선택 가능, 비내림차순)
- $1 \leq M \leq N \leq 8$

N과 M (4)

https://www.acmicpc.net/problem/15652

```
bool c[10]; int a[10];
void go(int index, int start, int n, int m) {
    if (index == m) {
        // 수열을 출력
        return;
    }
    for (int i=start; i<=n; i++) {
        //if (c[i]) continue;
        c[i] = true; a[index] = i;
        go(index+1, i, n, m);
        c[i] = false;
    }
}
// go(0, 1, n, m);
```

N과 M (4)

https://www.acmicpc.net/problem/15652

- 소스: http://codeplus.codes/9c31d2a89eca43d28a46089e62ac56fb

N과 M (4)

https://www.acmicpc.net/problem/15652

- 1부터 N까지 자연수 중에서 M개를 고른 수열을 모두 구하는 문제 (중복 선택 가능, 비내림차순)
- $1 \leq M \leq N \leq 8$
- 비내림차순만 고르는 것이기 때문에, 다른 방식도 가능하다.
- 각각의 자연수를 선택하는 경우와 선택하지 않는 경우가 있다.
- 하지만, 중복 선택이 가능하기 때문에, 선택하는 경우를 i개 선택하는 경우로 세분화해야 한다.

N과 M (4)

https://www.acmicpc.net/problem/15652

```
int cnt[10];
void go(int index, int selected, int n, int m) {
    if (selected == m) {
        // 수열 출력
        return;
    }
    if (index > n) return;
    for (int i=m-selected; i>=1; i--) {
        cnt[index] = i;
        go(index+1, selected+i, n, m);
    }
    cnt[index] = 0;
    go(index+1, selected, n, m);
}
```

N과 M (4)

https://www.acmicpc.net/problem/15652

```
// 수열 출력
for (int i=1; i<=n; i++) {
    for (int j=1; j<=cnt[i]; j++) {
        cout << i << ' ';
    }
}
cout << '\n';
```

N과 M (4)

https://www.acmicpc.net/problem/15652

- 소스: http://codeplus.codes/e95eb2f28fbc4a8bb9c0885ac896ad3c

BFS

BFS

BFS

- BFS의 목적은 임의의 정점에서 시작해서, 모든 정점을 한 번씩 방문하는 것이다.

BFS

BFS

- BFS는 최단 거리를 구하는 알고리즘이다.

BFS

BFS

- BFS는 모든 가중치가 1일 때, 최단 거리를 구하는 알고리즘이다.

BFS

BFS

- BFS를 이용해 해결할 수 있는 문제는 아래와 같은 조건을 만족해야 한다

1. 최소 비용 문제이어야 한다
2. 간선의 가중치가 1이어야 한다
3. 정점과 간선의 개수가 적어야 한다. (적다는 것은 문제의 조건에 맞춰서 해결할 수 있다는 것을 의미한다)

BFS

BFS

- BFS를 이용해 해결할 수 있는 문제는 아래와 같은 조건을 만족해야 한다

1. **최소 비용** 문제이어야 한다
2. **간선의 가중치**가 1이어야 한다
3. 정점과 간선의 개수가 적어야 한다. (적다는 것은 문제의 조건에 맞춰서 해결할 수 있다는 것을 의미한다)

- 간선의 가중치가 문제에서 구하라고 하는 최소 비용과 의미가 일치해야 한다
- 즉, 거리의 최소값을 구하는 문제라면 가중치는 거리를 의미해야 하고, 시간의 최소값을 구하는 문제라면 가중치는 시간을 의미해야 한다

BFS

숨바꼭질

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

- 수빈이의 위치: N
- 동생의 위치: K
- 동생을 찾는 가장 빠른 시간을 구하는 문제

- 수빈이가 할 수 있는 행동 (위치: X)

1. 걷기: X+1 또는 X-1로 이동 (1초)
2. 순간이동: 2*X로 이동 (1초)

숨바꼭질

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

- 수빈이의 위치: N
- 동생의 위치: K
- 동생을 찾는 **가장 빠른 시간**을 구하는 문제

- 수빈이가 할 수 있는 행동 (위치: X)

1. 걷기: X+1 또는 X-1로 이동 (**1초**)
2. 순간이동: 2*X로 이동 (**1초**)

숨바꼭질

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

- 수빈이의 위치: 5
- 동생의 위치: 17

- 5-10-9-18-17 로 4초만에 동생을 찾을 수 있다.

숨바꼭질

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

- 큐에 수빈이의 위치를 넣어가면서 이동시킨다
- 한 번 방문한 곳은 다시 방문하지 않는 것이 좋기 때문에, 따로 배열에 체크하면서 방문

숨바꼭질

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

- check[i] = i를 방문했는지
- dist[i] = i를 몇 번만에 방문했는지

숨바꼭질

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

- now -> next를 갔다고 한다면

```
if (check[next] == false) {
    q.push(next);
    check[next] = true;
    dist[next] = dist[now] + 1;
}
```

숨바꼭질

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

```
check[n] = true;
dist[n] = 0;
queue<int> q;
q.push(n);
while (!q.empty()) {
    int now = q.front();
    q.pop();
    if (now-1 >= 0) {
        if (check[now-1] == false) {
            q.push(now-1);
            check[now-1] = true;
            dist[now-1] = dist[now] + 1;
        }
    }
    if (now+1 < MAX) {
        if (check[now+1] == false) {
            q.push(now+1);
            check[now+1] = true;
            dist[now+1] = dist[now] + 1;
        }
    }
    if (now*2 < MAX) {
        if (check[now*2] == false) {
            q.push(now*2);
            check[now*2] = true;
            dist[now*2] = dist[now] + 1;
        }
    }
}
```

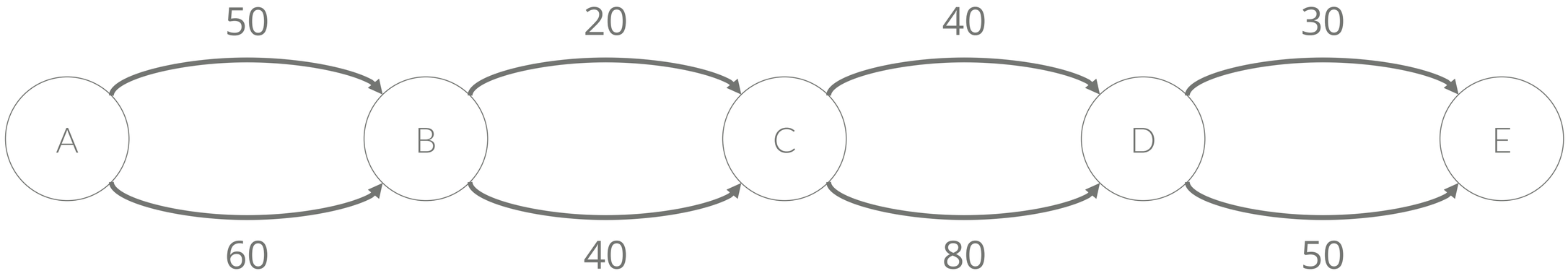
숨바꼭질

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

- 소스: http://codeplus.codes/fa34029958854eaa970dcb65bb8db39b

BFS

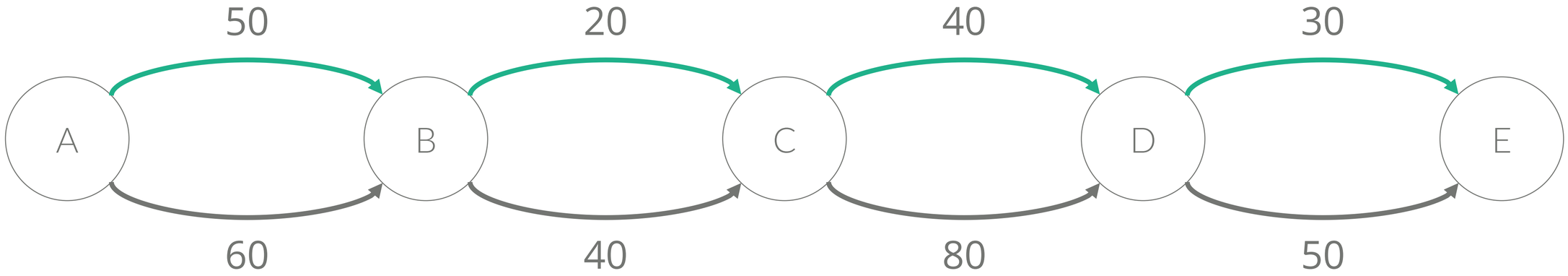
BFS



- A에서 E로 가는 가장 빠른 길은 무엇일까?

BFS

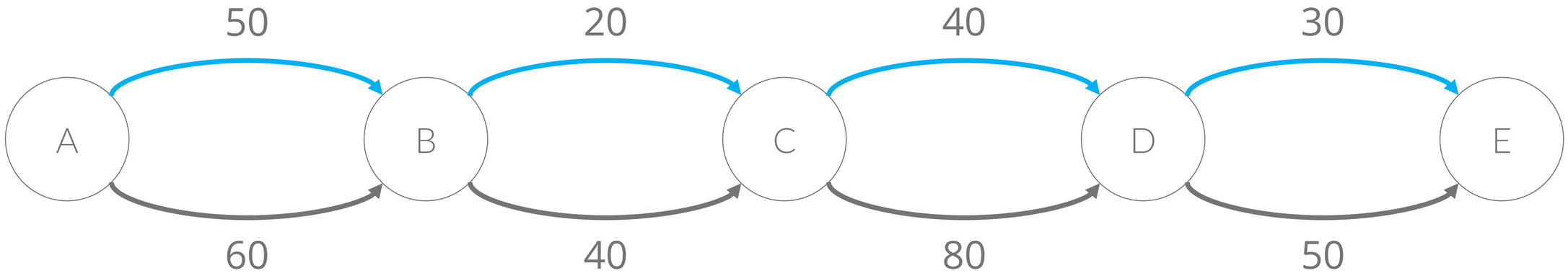
BFS



- A에서 E로 가는 가장 빠른 길은 무엇일까?
- A에서 B로 가는 가장 빠른 길 + B에서 E로 가는 가장 빠른 길

BFS

BFS

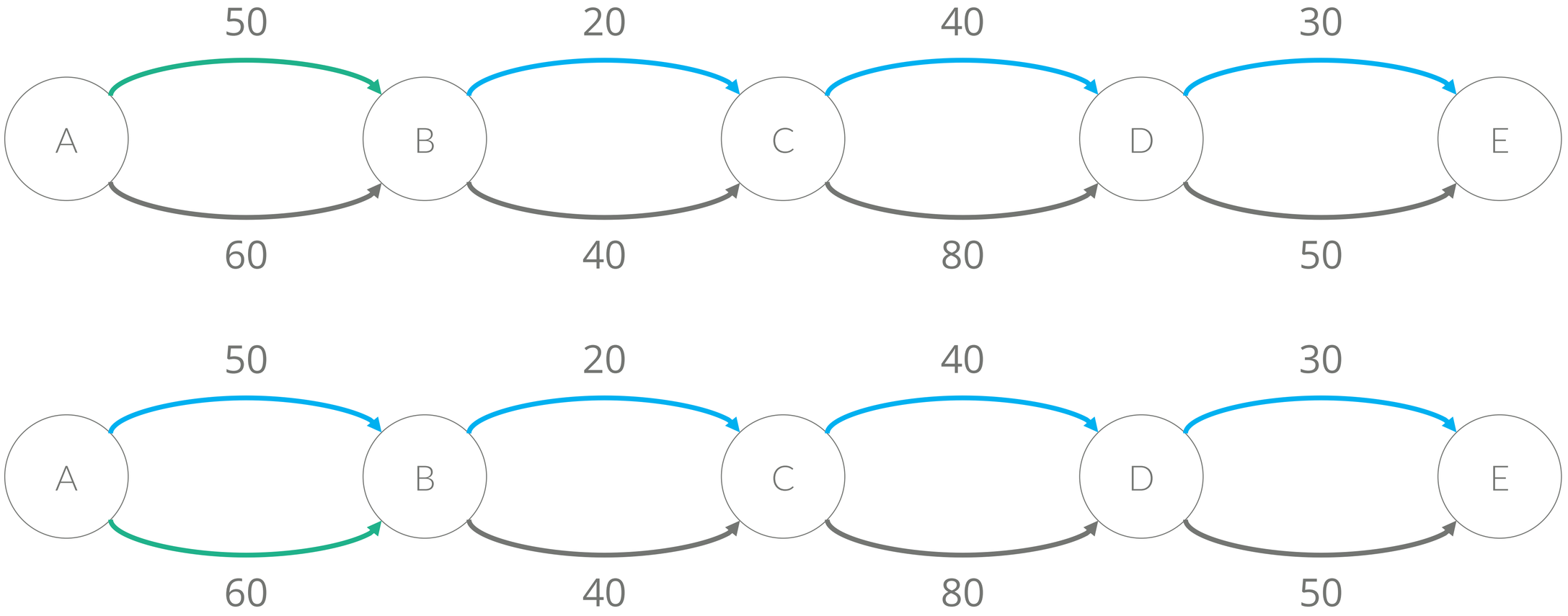


- A에서 E로 가는 가장 빠른 길은 무엇일까?
- 단, 파란 간선은 한 번만 사용할 수 있다.

BFS

BFS

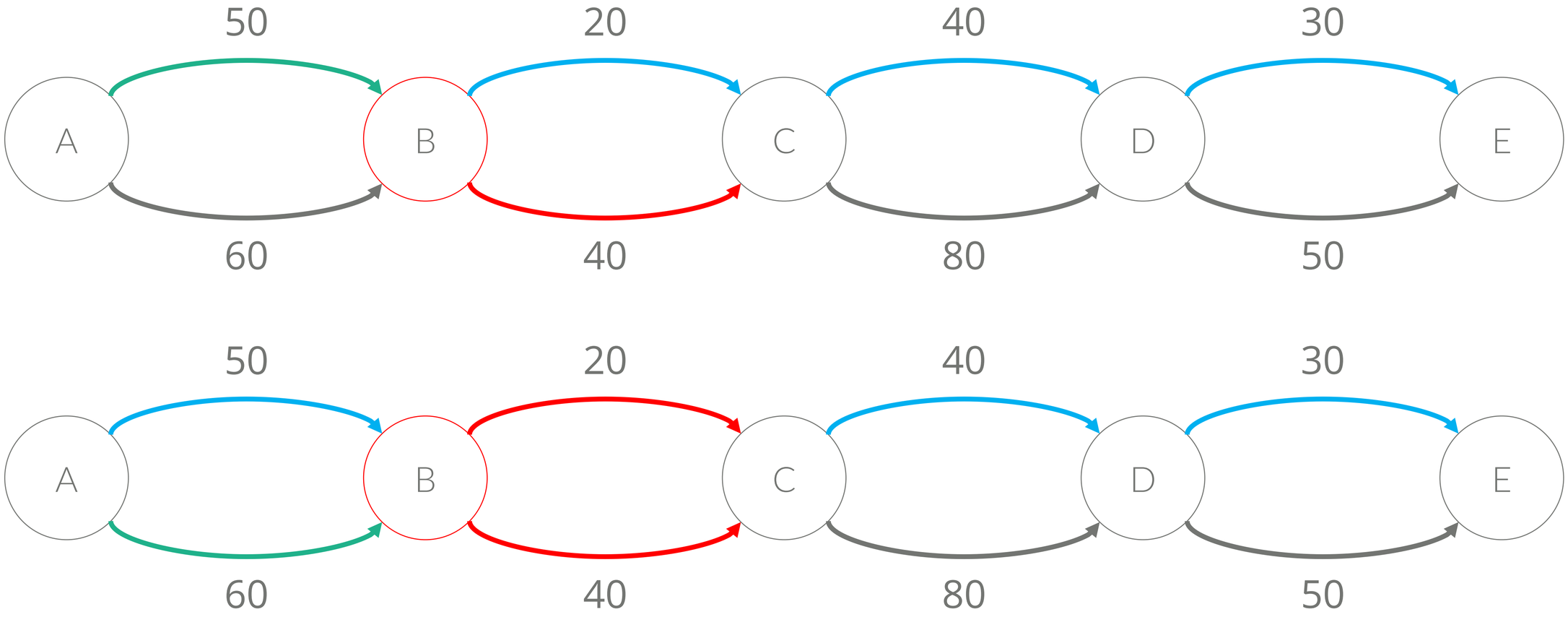
- 파란 간선을 한 번만 사용할 수 있다면, 위 B와 아래 B는 같은 정점이라고 할 수 없다



BFS

BFS

- 위 B와 아래 B에서 이동할 수 있는 방법이 다르기 때문에 같은 정점이 아니다



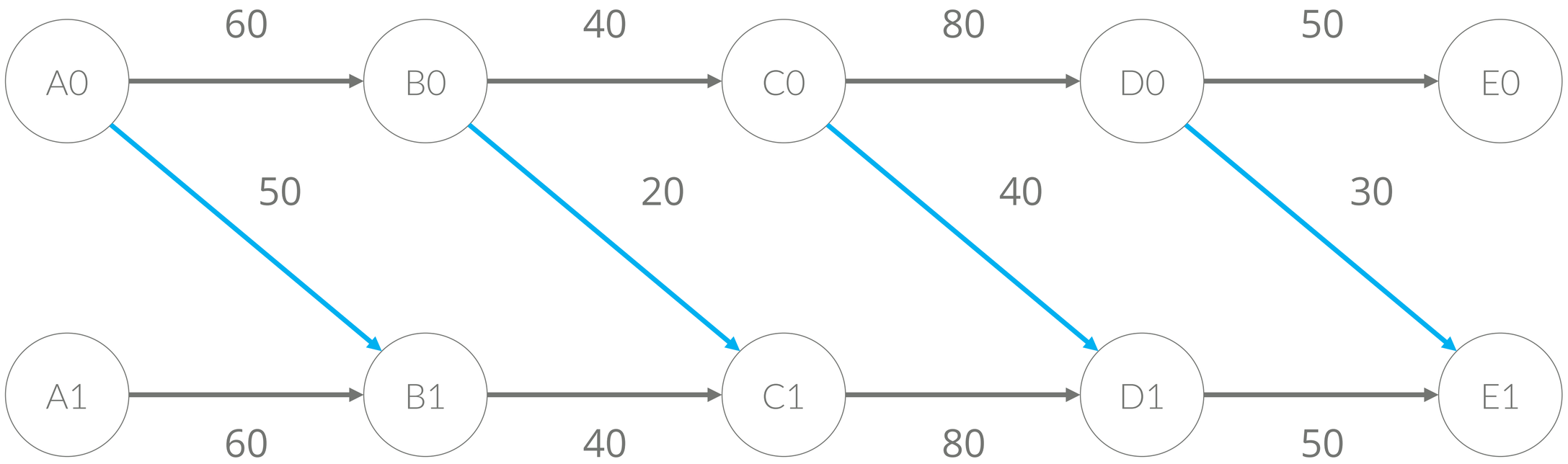
BFS

BFS

- 위 B와 아래 B를 다르다고 하는 기준은 파란 간선을 사용한 횟수이다
- 따라서, 정점을 파란 간선을 사용한 횟수를 기준으로 나눌 수 있다.

BFS

BFS



이모티콘

https://www.acmicpc.net/problem/14226

- 화면에 이모티콘은 1개다
- 할 수 있는 연산
  - 화면에 있는 이모티콘을 모두 복사해서 클립보드에 저장
  - 클립보드에 있는 모든 이모티콘을 화면에 붙여넣기
  - 화면에 있는 이모티콘 중 하나를 삭제
- S개의 이모티콘을 만드는데 걸리는 시간의 최소값을 구하는 문제

이모티콘

https://www.acmicpc.net/problem/14226

- BFS에서 하나의 정점이 서로 다른 두 개의 정보를 저장하고 있으면 안된다
- 화면에 있는 이모티콘의 개수가 5개인 경우
- 클립보드에 있는 이모티콘의 개수에 따라서, 클립보드에서 복사하기 연산의 결과가 다르다
- 즉, 화면에 이모티콘의 개수 s와 클립보드에 있는 이모티콘의 개수 c가 중요하다

이모티콘

https://www.acmicpc.net/problem/14226

- 복사: (s, c) -> (s, s)
- 붙여넣기: (s, c) -> (s+c, c)
- 삭제: (s, c) -> (s-1, c)
- $2 \leq S \leq 1{,}000$ 이기 때문에 BFS 탐색으로 가능하다.

이모티콘

https://www.acmicpc.net/problem/14226

- 소스: http://codeplus.codes/26314b769c684d42b406505723fa8846

게리맨더링 2

https://www.acmicpc.net/problem/17779

- 크기가 N×N인 격자 모양의 도시를 5개의 선거구로 나누는 문제
- 한 선거구에 포함되어 있는 구역은 모두 연결되어 있어야 한다.
- 선거구를 나누는 방법 중에서, 인구가 가장 많은 선거구와 가장 적은 선거구의 인구 차이의 최솟값을 구하는 문제
- 선거구를 나누는 방법은 문제 참고

1	1	1	1	2	2	2
1	1	1	5	2	2	2
1	1	5	5	5	2	2
3	5	5	5	5	5	2
3	3	5	5	5	4	4
3	3	3	5	4	4	4
3	3	3	4	4	4	4

1	1	1	1	1	2	2
1	1	1	1	5	2	2
1	1	1	5	5	5	2
1	1	5	5	5	5	5
3	5	5	5	5	5	4
3	3	5	5	5	4	4
3	3	3	5	4	4	4

게리맨더링 2

https://www.acmicpc.net/problem/17779

- 선거구를 나누는 방법의 수를 구해보자
- 기준점 (x, y)와 경계의 길이 d_1, d_2를 정해야 한다.
- 기준점은 N^2개를 넘지 않는다.
- 경계는 대각선 모양이기 때문에, 길이 d_1, d_2는 2N을 넘지 않는다.
- 따라서, x, y, d_1, d_2의 조합은 $N \times N \times 2N \times 2N$ 개를 넘지 않는다.
- 선거구를 나눈 후에 인구의 합을 구하려면 전체 격자를 순회해야 한다. $\rightarrow N^2$
- $O(N^6) \leq$ 1억

게리맨더링 2

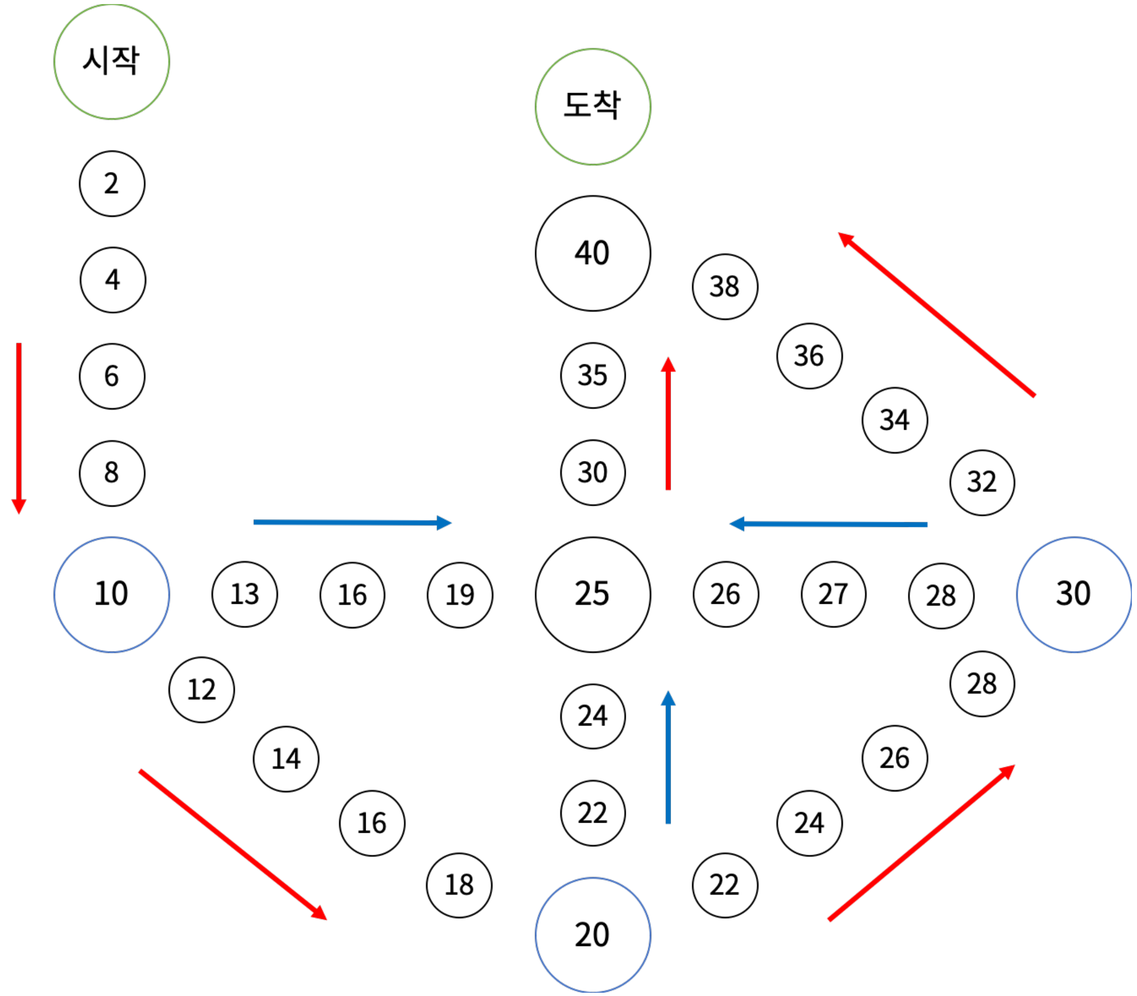
https://www.acmicpc.net/problem/17779

- 소스: http://codeplus.codes/723b5abd4914417190a1c857c0535643

주사위 윷놀이

https://www.acmicpc.net/problem/17825

- 주사위에서 나올 수 10개가 순서대로 주어졌을 때, 얻을 수 있는 점수의 최댓값을 구하는 문제



주사위 윷놀이

https://www.acmicpc.net/problem/17825

- 주사위는 1부터 5까지의 수가 써있다.
- 말은 총 4개이며, 모두 시작에 있다.
- 파란색 칸에서 말이 이동을 시작하는 경우 파란색 화살표의 방향으로 이동해야 한다.
- 파란색 칸을 지나는 경우는 빨간 화살표의 방향대로 이동해야 한다.
- 이동하려고 하는 칸에 말이 이미 있으면, 이동할 수 없다. (시작/도착 제외)
- 말이 이동을 마칠때마다 칸에 적혀있는 수가 점수에 추가된다.

주사위 윷놀이

https://www.acmicpc.net/problem/17825

- 주사위를 한 번 던질때마다 말 4개중 하나를 옮겨야 한다.
- 주사위를 10번 던져야하기 때문에, 총 방법의 수는 4^{10}이다.

주사위 윷놀이

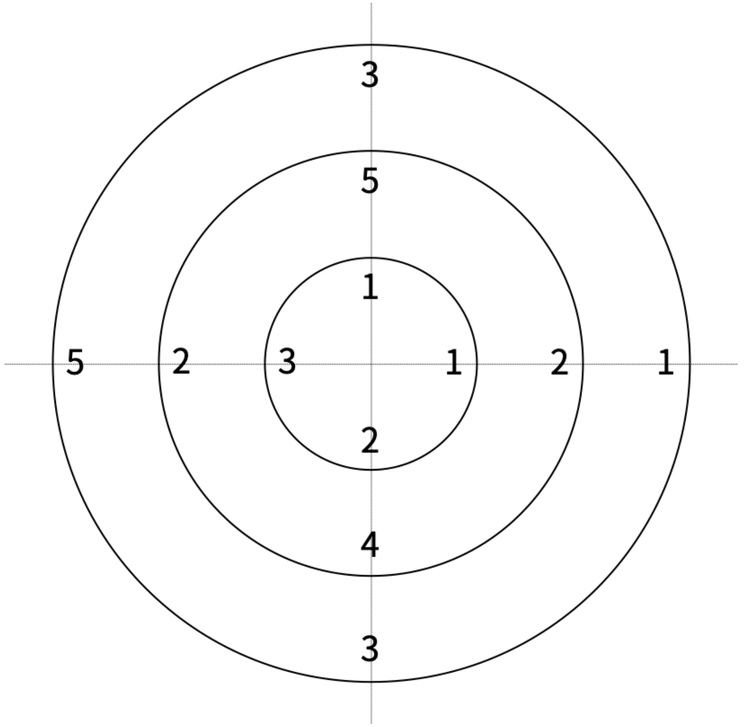
https://www.acmicpc.net/problem/17825

- 소스: http://codeplus.codes/8ec70197514843b396e7cbfbc53f121a

원판 돌리기

https://www.acmicpc.net/problem/17822

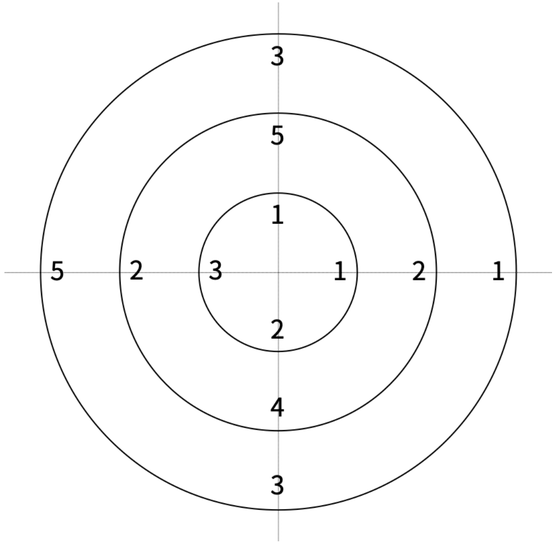
- 반지름이 1부터 N까지인 원판이 있다.
- i번째 원판의 j번째 수의 위치는 (i, j)이다.
- 원판을 회전시켜야 한다.

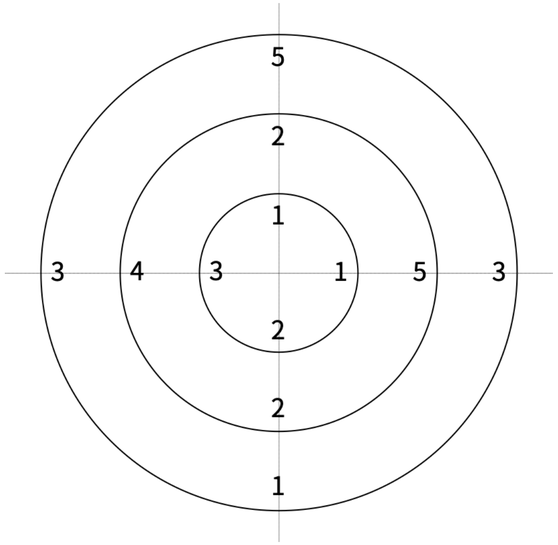


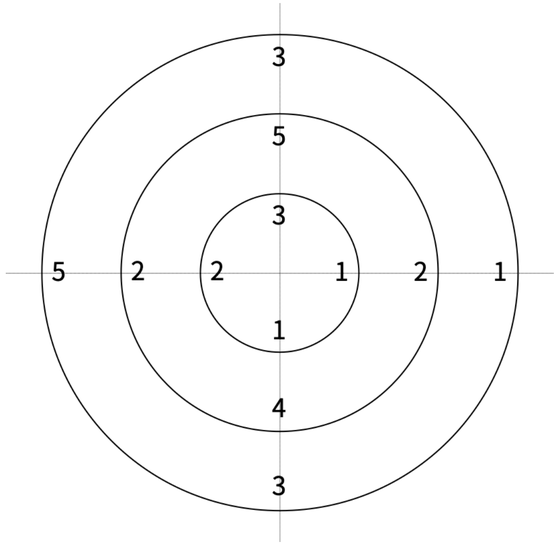
원판 돌리기

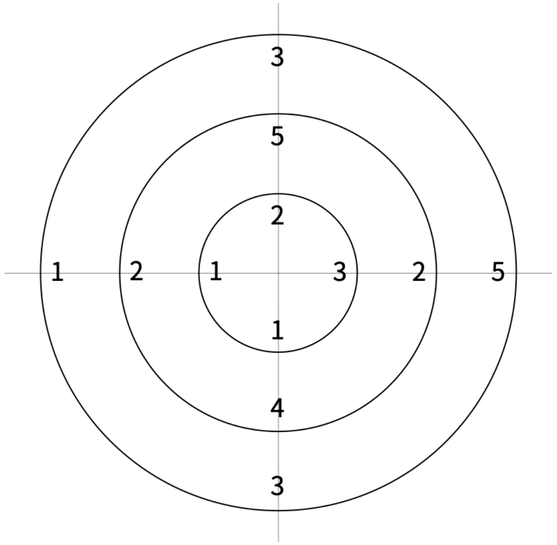
https://www.acmicpc.net/problem/17822

1. 1번 원판을 시계 방향으로 1칸 회전
2. 2, 3번 원판을 반시계 방향으로 3칸 회전
3. 1, 3번 원판을 시계 방향으로 2칸 회전









원판 돌리기

https://www.acmicpc.net/problem/17822

- 원판을 T번 회전시키려고 한다.

1. 번호가 x_i의 배수인 원판을 d_i방향으로 k_i칸 회전시킨다. d_i가 0인 경우는 시계 방향, 1인 경우는 반시계 방향이다.
2. 인접하면서 수가 같은 것을 모두 찾는다.
    1. 그러한 수가 있는 경우에는 원판에서 인접하면서 같은 수를 모두 지운다.
    2. 없는 경우에는 원판에 적힌 수의 평균을 구하고, 평균보다 큰 수에서 1을 빼고, 작은 수에는 1을 더한다.

원판 돌리기

https://www.acmicpc.net/problem/17822

- 소스: http://codeplus.codes/31f80373a70f4986a98978901c0b63c3

새로운 게임

https://www.acmicpc.net/problem/17780

- N×N 크기의 체스판에서 말 K개로 진행하는 게임
- 말은 원판 모양, 말 위에 말을 올릴 수 있고, 1부터 K까지 번호가 매겨져 있음
- 체스판의 색상은 흰색, 빨간색, 파란색 중 하나
- 말의 이동 방향(위, 아래, 왼쪽, 오른쪽)은 미리 정해져 있음
- 턴:1번부터 K번까지 말을 순서대로 이동시키는 것
- 한 말이 이동할 때, 위에 올려져 있는 말도 함께 이동하고, 가장 아래에 있는 말만 이동 가능
- 말이 4개 쌓이면 게임이 종료됨

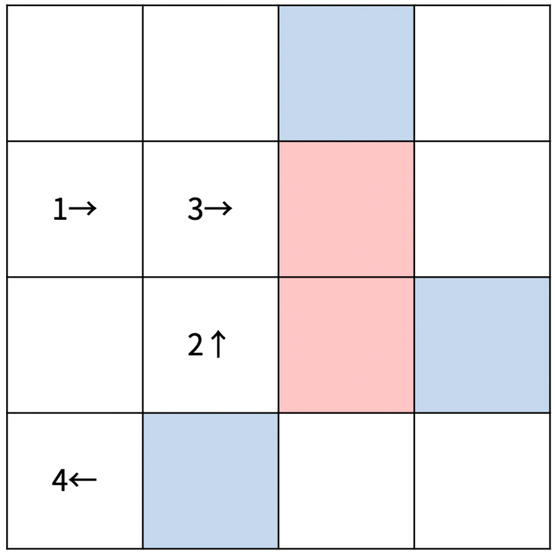
새로운 게임

https://www.acmicpc.net/problem/17780

- A번 말이 이동하려는 칸이
  - 흰색이면 그 칸으로 이동, 이동하려는 칸에 말이 이미 있으면 가장 위에 A번 말을 올려놓음
    - A, B, C로 쌓여있고, 이동하려는 칸에 D, E가 있으면, D, E, A, B, C
  - 빨간색이면 이동하고 A번 말과 그 위에 있는 말의 순서를 반대로 바꿈
    - A, B, C가 이동하고, 이동하려는 칸에 말이 없으면 C, B, A
    - A, D, F, G가 이동할고, 이동하려는 칸에 E, C, B가 있으면 E, C, B, G, F, D, A
  - 파란색인 경우에는 A번 말의 이동 방향을 반대로 하고 한 칸을 이동, 이동하려는 칸도 파란색이면 방향만 반대로 바꿈
  - 체스판을 벗어나는 경우에는 파란색과 같은 경우

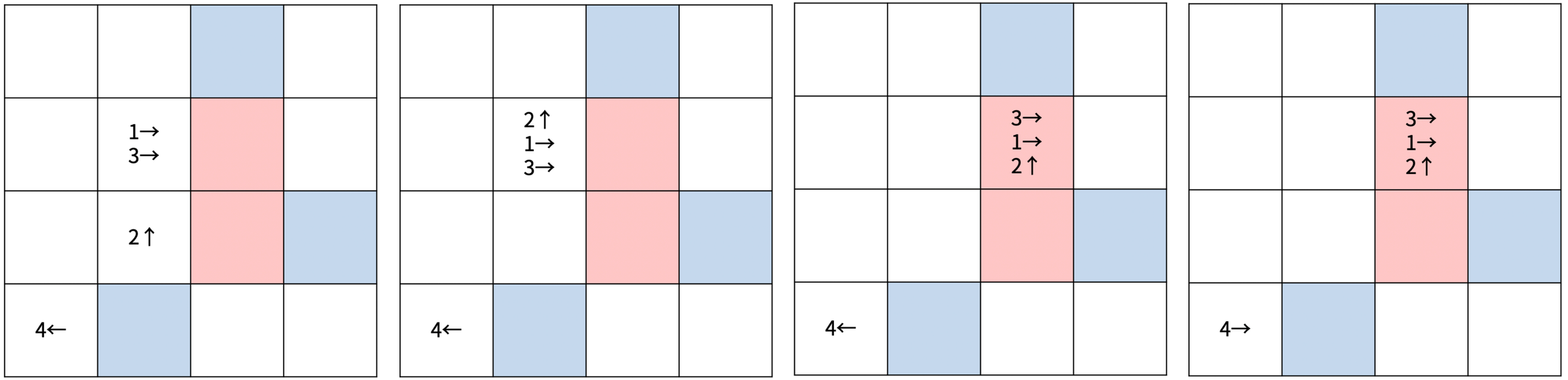
새로운 게임

https://www.acmicpc.net/problem/17780



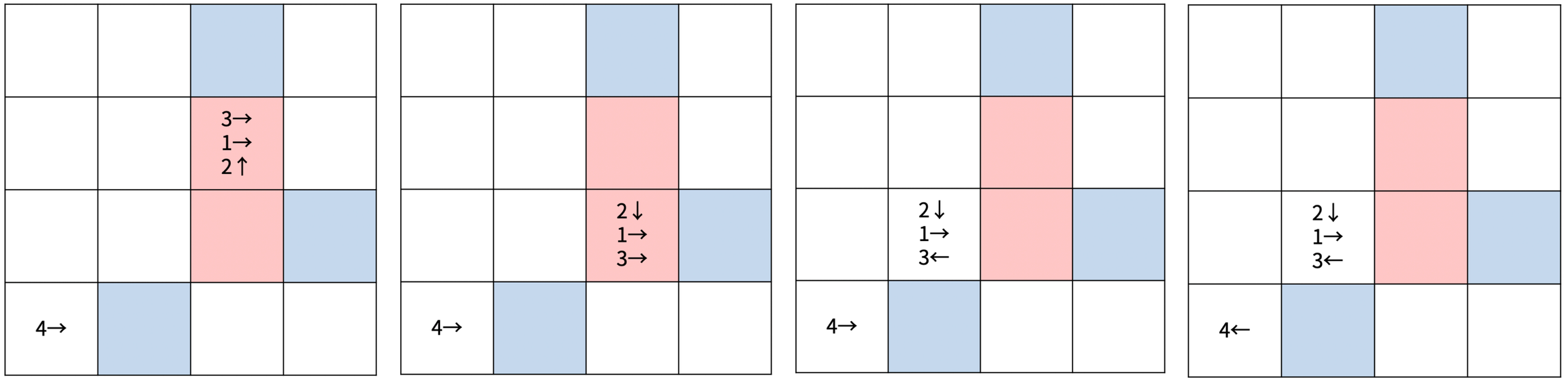
새로운 게임

https://www.acmicpc.net/problem/17780



새로운 게임

https://www.acmicpc.net/problem/17780



새로운 게임

https://www.acmicpc.net/problem/17780

- 게임이 언제 종료되는지 구하는 문제

새로운 게임

https://www.acmicpc.net/problem/17780

- 소스: http://codeplus.codes/f6ac04f74aa944518f0abe5e5d3ad39e

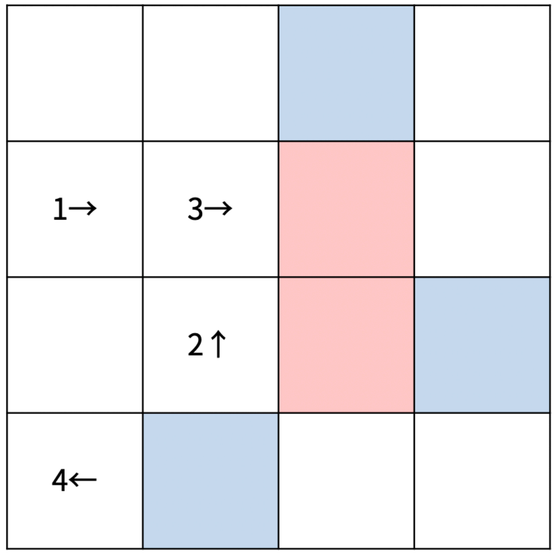
새로운 게임 2

https://www.acmicpc.net/problem/17837

- N×N 크기의 체스판에서 말 K개로 진행하는 게임
- 말은 원판 모양, 말 위에 말을 올릴 수 있고, 1부터 K까지 번호가 매겨져 있음
- 체스판의 색상은 흰색, 빨간색, 파란색 중 하나
- 말의 이동 방향(위, 아래, 왼쪽, 오른쪽)은 미리 정해져 있음
- 턴:1번부터 K번까지 말을 순서대로 이동시키는 것
- 한 말이 이동할 때, 위에 올려져 있는 말도 함께 이동하고, ~~가장 아래에 있는 말만 이동 가능~~
- 말이 4개 쌓이면 게임이 종료됨

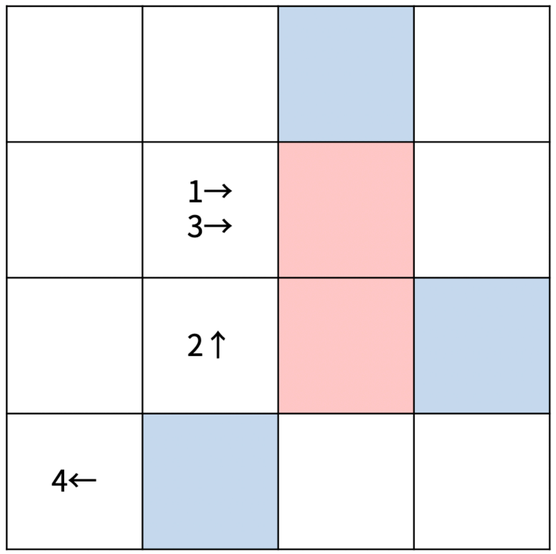
새로운 게임 2

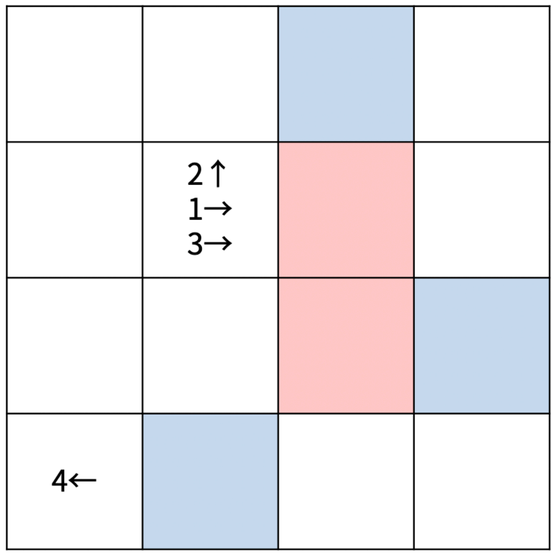
https://www.acmicpc.net/problem/17837

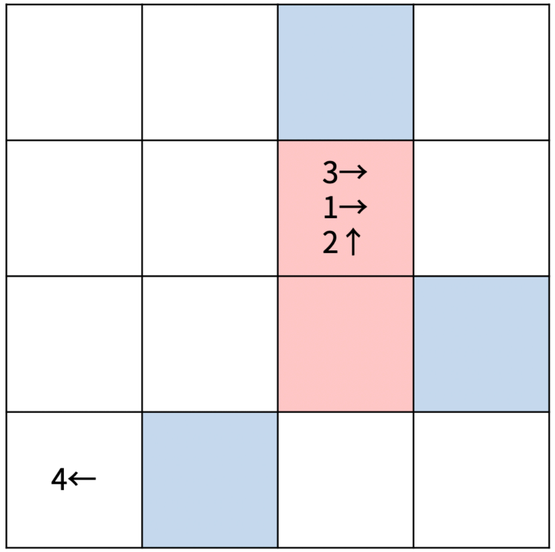


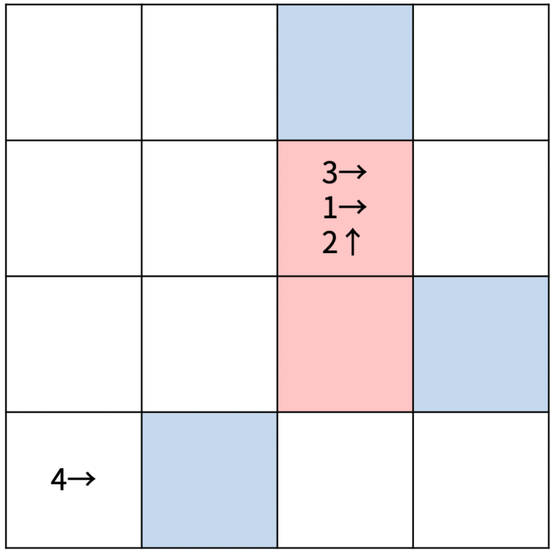
새로운 게임 2

https://www.acmicpc.net/problem/17837



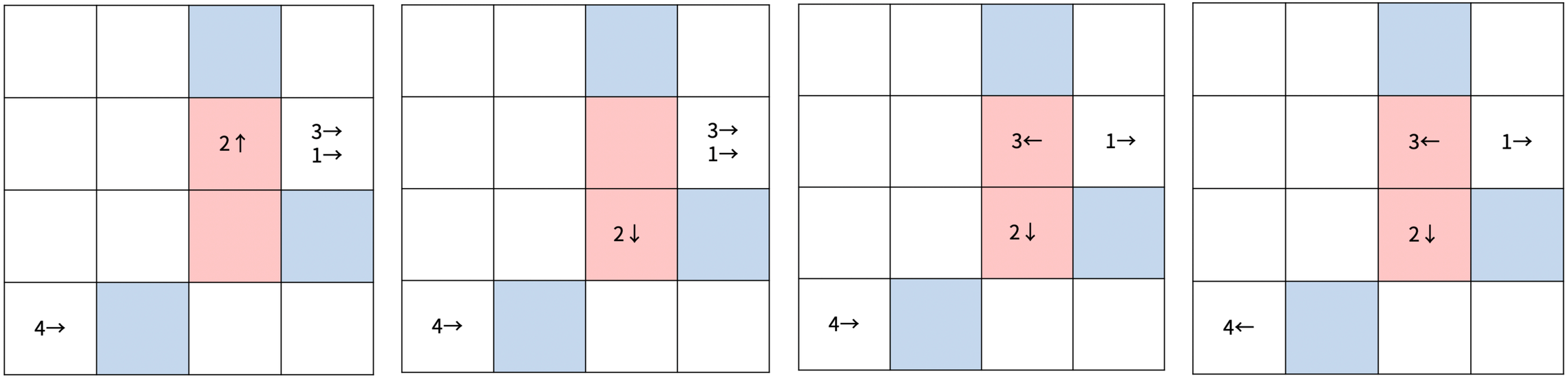






새로운 게임 2

https://www.acmicpc.net/problem/17837



새로운 게임 2

https://www.acmicpc.net/problem/17837

- 소스: http://codeplus.codes/b99161cb6c9d4d208452c1140e8afb48