# 다이나믹 프로그래밍 5

최백준 choi@startlink.io

# 기대값 DP

# 토끼의 이동

https://www.acmicpc.net/problem/13247

- N개의 칸이 (0~N-1)이 가로로 놓여져 있다
- 각 칸은 흰색, 검정색, 빨간색 중 하나로 색칠되어 있다
- r마리의 토끼가 모여서 게임을 하려고 한다
- 각 토끼는 게임을 시작할 칸을 무작위로 고르며, 두 토끼가 같은 칸에 있는 경우는 없다
- r개의 칸이 골라질 확률은 모두 같다
- 게임판의 크기가 2보다 큰 동안 다음 페이지의 내용을 반복한다

# 토끼의 이동

https://www.acmicpc.net/problem/13247

- 0번 칸에 있는 토끼는 1번 칸으로 이동한다
- N-1, N-2에 있는 토끼는 왼쪽 칸으로 이동한다
- 나머지 토끼는 색상에 따라 다르다
    - 흰색: 왼쪽
    - 검정색: 오른쪽
    - 빨간색: 처음이면 왼쪽, 이동한 적이 있으면 이전 칸
- 이동이 모두 끝난 후에 두 마리 이상 토끼가 있는 칸에 있는 토끼를 제외한다
- 가장 오른쪽 칸을 제거한다
- 게임판에 남아있을 수 있는 토끼의 기댓값을 구하는 문제

# 토끼의 이동



https://www.acmicpc.net/problem/13247

- 실제로 토끼의 이동을 시뮬레이션 해본다

# 토끼의 이동

https://www.acmicpc.net/problem/13247

- 정답 = 모든 경우에서 남은 토끼의 합 / 가능한 상태의 개수

# 토끼의 이동

https://www.acmicpc.net/problem/13247

- https://gist.github.com/Baekjoon/549080241846a40bc7d2cc070bf35817

# 공의 충돌

https://www.acmicpc.net/problem/13249

- 무게가 모두 같고, 크기가 0인 공 N개가 일직선 위에 놓여져 있다. ($1 \leq N \leq 12$)
- 오른쪽으로 굴러가는 공과 왼쪽으로 굴러가는 공이 같은 속도로 충돌하면, 속도는 변하지 않고 공의 진행 방향만 바뀌게 된다.
- 공 N개의 위치가 주어진다. 효빈이는 공 N개의 진행 방향(오른쪽, 왼쪽)을 같은 확률로 결정한다. 시간 0일 때, 효빈이는 공을 결정한 방향으로 동시에 1초에 1만큼 이동하는 속도로 굴린다.
- T초 후에 공이 충돌한 횟수의 기댓값을 구하는 문제 (T초에 충돌한 것도 포함해야 한다)

# 공의 충돌



https://www.acmicpc.net/problem/13249

- 공은 모두 같은 속도로 움직인다
- 공의 크기도 0이기 때문에, 공은 구분할 수 없다
- 이 문제는 공이 서로 충돌하지 않고 통과한다고 가정하고 문제를 풀어도 된다
- 공의 개수가 작기 때문에, 가능한 모든 방향에 대해서 문제를 푼다

# 공의 충돌

https://www.acmicpc.net/problem/13249

- 가능한 모든 방향에 대해서, 공이 서로 통과할 수 있다면, 충돌이 1번 일어나는 것이다

# 공의 충돌



https://www.acmicpc.net/problem/13249

```
sort(a.begin(),a.end());
for (int s=0; s<(1<<n); s++) {
    for (int i=0; i<n; i++) {
        if (s&(1<<i)) {
            for (int j=i+1; j<n; j++) {
                if (!(s&(1<<j))) {
                    if (a[i]+t >= a[j]-t) {
                        ans += 1;
                    }
                }
            }
        }
    }
}
```

# 공의 충돌

https://www.acmicpc.net/problem/13249

- https://gist.github.com/Baekjoon/d1d20503cd21ed543aa393f8b83cfc36

# 축구

https://www.acmicpc.net/problem/1344

- 축구는 90분동안 이루어지고, 5분 간격으로 나눴다
- 경기가 진행되는 동안 각 간격에서 A팀이 득점할 확률과 B팀이 득점할 확률이 주어진다
- 각 간격에서 두 팀은 각각 많아야 한 골을 득점할 수 있다
- 경기가 끝난 후 적어도 한 팀이 골을 소수로 득점할 확률을 구하는 문제

# 축구

https://www.acmicpc.net/problem/1344

- D[N][A][B] = N번째 간격에서 스코어가 A:B일 확률
- D[0][0][0] = 1.0
- D[N][A][B]는 총 4가지 경우가 가능
    - A가 득점
    - B가 득점
    - A와 B가 득점
    - 두 팀 모두 득점하지 못함

# 축구

https://www.acmicpc.net/problem/1344

- D[N][A][B] = N번째 간격에서 스코어가 A:B일 확률
- D[0][0][0] = 1.0
- D[N][A][B]는 총 4가지 경우가 가능
  - A가 득점 = D[N-1][A-1][B]
  - B가 득점 = D[N-1][A][B-1]
  - A와 B가 득점 = D[N-1][A-1][B-1]
  - 두 팀 모두 득점하지 못함 = D[N-1][A][B]

# 축구

https://www.acmicpc.net/problem/1344

- D[N][A][B] = N번째 간격에서 스코어가 A:B일 확률
- D[0][0][0] = 1.0
- D[N][A][B]는 총 4가지 경우가 가능
    - A가 득점 = D[N-1][A-1][B] * PA * (1-PB)
    - B가 득점 = D[N-1][A][B-1] * (1-PA) * PB
    - A와 B가 득점 = D[N-1][A-1][B-1] * PA * PB
    - 두 팀 모두 득점하지 못함 = D[N-1][A][B] * (1-PA) * (1-PB)

# 축구



https://www.acmicpc.net/problem/1344

```
d[0][0][0] = 1.0;
for (int i=1; i<=90/5; i++) {
    for (int j=0; j<=i; j++) {
        for (int k=0; k<=i; k++) {
            if (j >= 1 && k >= 1)
                d[i][j][k] += d[i-1][j-1][k-1]*a*b;
            if (j >= 1)
                d[i][j][k] += d[i-1][j-1][k]*a*(1.0-b);
            if (k >= 1)
                d[i][j][k] += d[i-1][j][k-1]*(1.0-a)*b;
            d[i][j][k] += d[i-1][j][k]*(1.0-a)*(1.0-b);
        }
    }
}
```

# 축구

https://www.acmicpc.net/problem/1344

- C/C++: https://gist.github.com/Baekjoon/0932e9a9e9f046216e6b

# 복권

https://www.acmicpc.net/problem/1359

- N개의 수 중에서 M개의 수를 고르는 복권이 있다
- 당첨 번호와 적어도 K개가 같으면 당첨이다
- 복권에 당첨될 확률을 구하는 문제

# 복권

https://www.acmicpc.net/problem/1359

- N개의 수 중에서 M개의 수를 고르는 복권이 있다
- 당첨 번호와 K개가 같을 확률
- 당천 번호 M개 중에 K개 같고, 나머지 N-M개 중에 M-K개 같으면 된다
- C[M][K] * C[N-M][M-K]

# 복권

https://www.acmicpc.net/problem/1359

- https://gist.github.com/Baekjoon/e20b2f198768664a8a9c0bf60b6db2a4

# 주사위 게임

https://www.acmicpc.net/problem/13250

- 주사위를 던질 때 마다 윗 면에 적힌 수 만큼 사탕을 받는다
- 사탕을 적어도 N개 받기 위해 던져야 하는 횟수의 기댓값을 구하는 문제

# 주사위 게임

https://www.acmicpc.net/problem/13250

- D[N] = 사탕을 적어도 N개 받기 위해 던져야 하는 횟수의 기댓값

# 주사위 게임



https://www.acmicpc.net/problem/13250

- D[N] = 사탕을 적어도 N개 받기 위해 던져야 하는 횟수의 기댓값
- D[N] = 1/6 * (D[N-1] + 1) + 1/6 * (D[N-2] + 1) + 1/6 * (D[N-3] + 1) + 1/6 * (D[N-4] + 1) + 1/6 * (D[N-5] + 1) + 1/6 * (D[N-6] + 1)
- D[N] = 1 + D[N-1]/6 + D[N-2]/6 + D[N-3]/6 + D[N-4]/6 + D[N-5]/6 + D[N-6]/6

# 주사위 게임



https://www.acmicpc.net/problem/13250

- https://gist.github.com/Baekjoon/f95651d705af1dffb7a9572633b65542

# 조약돌 꺼내기



https://www.acmicpc.net/problem/13251

- 효빈이의 비밀 박스에는 조약돌을 N개 들어있다. 조약돌의 색상은 1부터 M까지 중의 하나이다.
- 비밀 박스에서 조약돌을 랜덤하게 K개 뽑았을 때, 뽑은 조약돌이 모두 같은 색일 확률을 구하는 문제

# 조약돌 꺼내기

https://www.acmicpc.net/problem/13251

- 첫 번째로 꺼낸 조약돌이 1번 색일 확률: A[i] / total

# 조약돌 꺼내기

https://www.acmicpc.net/problem/13251

- 첫 번째로 꺼낸 조약돌이 1번 색일 확률: A[i] / total
- 두 번째로 꺼낸 조약돌이 1번 색일 확률: (A[i] – 1) / (total – 1)
- …
- K개의 조약돌이 모두 1번색 일 확률
- Σ (A[i]-j) / (total – j) ($0 \leq j < K$)

# 조약돌 꺼내기



https://www.acmicpc.net/problem/13251

- 각각의 색에 대해서 모두 구해보면 된다

# 조약돌 꺼내기



https://www.acmicpc.net/problem/13251

- https://gist.github.com/Baekjoon/5f96d12c64e2163c5641e744dc800b7c

# 카지노



https://www.acmicpc.net/problem/13252

- N명이 카지노를 방문했다.
- 게임은 총 K개의 라운드로 이루어져 있다
- 각 라운드는 다음과 같다

1. 각 플레이어는 M개의 영역 중 하나에 칩을 놓는다.
2. 딜러가 M개의 영역 중에 하나를 랜덤으로 고른다.
3. 딜러가 고른 영역에 칩을 놓은 사람은 게임에서 제외된다.

- 효빈이와 친구들이 게임을 최적의 방법으로 진행했을 때, 적어도 한 사람이 게임을 승리할 확률을 구하는 문제

# 카지노

https://www.acmicpc.net/problem/13252

- 문제는 두 가지 부분으로 나누어져 있다
- 첫 번째는 각 라운드에서 최적의 전략을 찾는 것이다
- 적어도 한 사람은 게임을 승리하는 최적의 전략을 찾아야 한다

# 카지노



https://www.acmicpc.net/problem/13252

- N명의 사람이 있고, M개의 영역이 있다
- 각 영역은 적어도 ⌊N/M⌋ 개의 동전이 있고
- N%M개의 영역에는 ⌈N/M⌉ 개의 동전이 있게 된다
- 이렇게 되면 최악의 경우에 각 라운드에 살아남는 사람의 수는 N - ⌈N/M⌉ 이 된다

# 카지노

https://www.acmicpc.net/problem/13252

- N명의 사람이 있고, M개의 영역이 있다
- 각 영역은 적어도 ⌊N/M⌋ 개의 동전이 있고
- N%M개의 영역에는 ⌈N/M⌉ 개의 동전이 있게 된다
- 이렇게 되면 최악의 경우에 각 라운드에 살아남는 사람의 수는 N - ⌈N/M⌉ 이 된다
- D[N][K] = 처음 N명이 있고, 게임이 총 K개 라운드로 이루어져 있을 때, 적어도 한 명이 게임을 승리할 확률 이라고 하면
- D[N][K] ≥ D[N - ⌈N/M⌉][K-1] 이다

# 카지노

https://www.acmicpc.net/problem/13252

- 각 영역이 선택될 확률은 모두 1/M 이다
- $\lceil N/M \rceil$개의 동전이 있는 영역이 선택될 확률은 P = (N%M)/M 이고
- $\lfloor N/M \rfloor$개의 동전이 있는 영역이 선택될 확률은 1-P 이다

# 카지노

https://www.acmicpc.net/problem/13252

- 각 영역이 선택될 확률은 모두 1/M 이다
- $\lceil N/M \rceil$개의 동전이 있는 영역이 선택될 확률은 P = (N%M)/M 이고
- $\lfloor N/M \rfloor$개의 동전이 있는 영역이 선택될 확률은 1-P 이다
- $D[N][K] = p * D[N - \lceil N/M \rceil][K] + (1-p) * D[N - \lfloor N/M \rfloor][K]$

# 카지노



https://www.acmicpc.net/problem/13252

- N이 너무 크기 때문에, 배열을 잡을 수 없다

# 카지노

https://www.acmicpc.net/problem/13252

- $N = 52, M = 7$인 경우에
- $N - \lceil N/M \rceil = 44$
- $N - \lfloor N/M \rfloor = 45$

# 카지노



https://www.acmicpc.net/problem/13252

- N = 44, 45,  M = 7인 경우에
- N = 37, 38, 39가 된다

# 카지노

https://www.acmicpc.net/problem/13252

- N = 37, 38, 39,  M = 7인 경우에
- N = 31, 32, 33, 34 이다

# 카지노

https://www.acmicpc.net/problem/13252

- 즉, N은 항상 구간을 나타내고, 구간의 크기는 최대 2씩 늘어나게 된다
- map 같은 것을 사용해서 DP를 하면 된다

# 카지노

https://www.acmicpc.net/problem/13252

- https://gist.github.com/Baekjoon/71540cd1c5c3b4fe0516f77f4898a69f

# 토러스

https://www.acmicpc.net/problem/13253

- 도넛해의 모든 칸은 좌표 (n, m)으로 나타낼 수 있다 ($0 \le n < N$, $0 \le m < M$)
- 상근이가 이동할 수 있는 방법은 두 좌표를 1씩 증가시키거나, 1씩 감소시키는 것이다
- 즉, 상근이가 (n, m)에 있다면, 이동할 수 있는 칸은 ((n+1)%N, (m+1)%M) 또는 ((n-1)%N, (m-1)%M) 이다
- 두 칸으로 이동할 확률은 같으며, 이동에는 하루가 걸린다
- 상근이가 (x, y)에 도착하는데 필요한 일의 기댓값을 구하는 문제

# 토러스

https://www.acmicpc.net/problem/13253

- N = 3, M = 3 이고
- (1, 1)에 가야 하는 경우
- 1/2의 확률로 1일만에 도착
- 1/4의 확률로 2일만에 도착
- 1/8의 확률로 3일만에 도착
- 1/16의 확률로 4일만에 도착
- …
- 정답: (1 * 1/2 ) + (2 * 1/4) + (3 * 1/8) + (4 * 1/16) + … = 2.0

# 토러스



https://www.acmicpc.net/problem/13253

- 상근이가 갈 수 있는 칸을 모두 구해보자
- (0, 0)에서 (1, 1)이나 (n-1, m-1)으로 이동하 수 있다.
- (1, 1)에서는 (0, 0)이나 (2 % n, 2 % m)으로 이동할 수 있다.
- 그런데, (0, 0)은 이미 계산을 했기 때문에, 다시 계산할 필요가 없다
- 즉, (x, y)에서 (0, 0)으로 돌아올 때 까지 ((x+1) % n, (y+1) % m)으로 이동하면 된다

# 토러스



https://www.acmicpc.net/problem/13253

- (0, 0)부터 시작해서, 방문할 수 있는 칸은 (i % n, i % m) 으로 나타낼 수 있고
- 개수는 cnt 라고 했을 때
- cnt + 1 = 0 (mod n), cnt + 1 = 0 (mod m) 을 만족하는 가장 작은 cnt 이다
- 즉, 0 ≤ i ≤ cnt 를 만족하는 (i % n, i % m)은 모두 다른 칸이다

# 토러스

https://www.acmicpc.net/problem/13253

- 토러스를 총 cnt개의 정점(0, 1, ⋯, cnt-1)으로 이루어진 그래프라고 했을 때
- x와 (x+1)%cnt 사이에 간선이 있는 것이다
- 목표로 하는 칸 (x, y) = (goal % n, goal % m)에서 정점 goal 이다

# 토러스



https://www.acmicpc.net/problem/13253

- 정점 0에서 이동을 시작한다.
- 인접한 정점으로 같은 확률로 이동했을 때 (Random walk)
- 정점 goal에 처음으로 도착하는 기댓값은? (Hitting time)
- Hitting time = (cnt – goal) * goal

# 토러스

https://www.acmicpc.net/problem/13253

- https://gist.github.com/Baekjoon/6a0495277fb7d95a99f6ca4234124fa8

# 토러스

https://www.acmicpc.net/problem/13253



- E[node] = 정점 node에 처음으로 도착하는 기댓값
- 기댓값 = 확률의 합 * 값
- 정점 x에서 갈 수 있는 정점은 x+1과 x-1 (모두 같은 확률: 1/2)

# 토러스



https://www.acmicpc.net/problem/13253

- E[node] = 정점 node에 처음으로 도착하는 기댓값
- 기댓값 = 확률의 합 * 값
- 정점 x에서 갈 수 있는 정점은 x+1과 x-1 (모두 같은 확률: 1/2)
- x에서 x-1로 갈 때: E[x-1] + 1
- x에서 x+1로 갈 때: E[x+1] + 1

# 토러스

https://www.acmicpc.net/problem/13253

- E[node] = 정점 node에 처음으로 도착하는 기댓값
- 기댓값 = 확률의 합 * 값
- 정점 x에서 갈 수 있는 정점은 x+1과 x-1 (모두 같은 확률: 1/2)
- x에서 x-1로 갈 때: E[x-1] + 1
- x에서 x+1로 갈 때: E[x+1] + 1
- E[x] = (E[x-1] + E[x+1]) / 2 + 1
- E[0] = 0

# 토러스

https://www.acmicpc.net/problem/13253

- $0 < x < cnt$
- $E[x] = (E[x-1] + E[x+1]) / 2 + 1$
- $E[0] = (E[1] + E[cnt]) / 2 + 1 = 0$
- $E[cnt] = (E[cnt-1] + E[0]) / 2 + 1 = E[cnt-1] / 2 + 1$

# 토러스

https://www.acmicpc.net/problem/13253

- E[i] = x*E[1] + y의 꼴로 나타내보자
- 이유는 E[1]은 0이 절대로 되지 않기 때문

# 토러스

https://www.acmicpc.net/problem/13253

- E[x] = (E[x - 1] + E[x + 1]) / 2 + 1
- 2 * E[x] = E[x - 1] + E[x + 1] + 2
- 2 * E[x] - E[x - 1] - E[x + 1] = 2
- 2 * E[x - 1] - E[x - 2] - E[x] = 2
- E[x] = 2 * E[x - 1] - E[x - 2] - 2

# 토러스



https://www.acmicpc.net/problem/13253

- E[x] = 2 * E[x - 1] - E[x - 2] - 2
- E[x] = a[x] * E[1] + b[x]
- a[0] = 0; b[0] = 0
- a[1] = 1; b[1] = 1

# 토러스

https://www.acmicpc.net/problem/13253

- E[x] = a[x] * E[1] + b[x] = 2 * (a[x - 1] * E[1] + b[x - 1]) - (a[x - 2] * E[1] + b[x - 2]) - 2
- a[x] * E[1] + b[x] = E[1] * (2 * a[x - 1] - a[x - 2]) + 2 * b[x - 1] - b[x - 2] - 2

- a[x] = 2 * a[x - 1] - a[x - 2]
- b[x] = 2 * b[x - 1] - b[x - 2] - 2

# 토러스



https://www.acmicpc.net/problem/13253

- E[0] = a[cnt] * E[1] + b[cnt] = 0
- E[1] = -b[cnt] / a[cnt]
- 정답 = a[goal]*E[1] + b[goal]

# 토러스



https://www.acmicpc.net/problem/13253

- https://gist.github.com/Baekjoon/7880c8bffab59565e4a47e8bd7ec4aea

# 토러스



https://www.acmicpc.net/problem/13253

- a[0] = 0; b[0] = 0
- a[1] = 1; b[1] = 1

- a[x] = 2 * a[x - 1] - a[x - 2]
- b[x] = 2 * b[x - 1] - b[x - 2] - 2

- a[x] = x
- b[x] = -i(i-1)

# 토러스

https://www.acmicpc.net/problem/13253

- a[x] = x
- b[x] = -i(i-1)
- E[1] = -b[cnt] / a[cnt]
- E[1] = (cnt-1)
- 정답 = a[goal]*E[1] + b[goal] = goal * (cnt – goal)

# 연휴

https://www.acmicpc.net/problem/13254

- 트리 형태의 도시가 주어진다
- 각각의 가족이 거주하는 도시가 주어진다
- 각 가족은 이동할 도시 하나를 랜덤하게 고른다
- 이 때, 모든 가족이 사용하는 도로 개수의 기댓값을 고르는 문제

# 연휴



https://www.acmicpc.net/problem/13254

- 각각의 도로가 모든 가족이 사용할 확률을 구해보자

# 연휴

https://www.acmicpc.net/problem/13254

- 각각의 도로가 모든 가족이 사용할 확률을 구해보자
- 모든 가족의 선택은 독립적이다
- 따라서, 각각의 도로에 대해서 각 가족이 사용할 확률을 구하고 그 확률을 모두 곱하면 된다

# 연휴

https://www.acmicpc.net/problem/13254

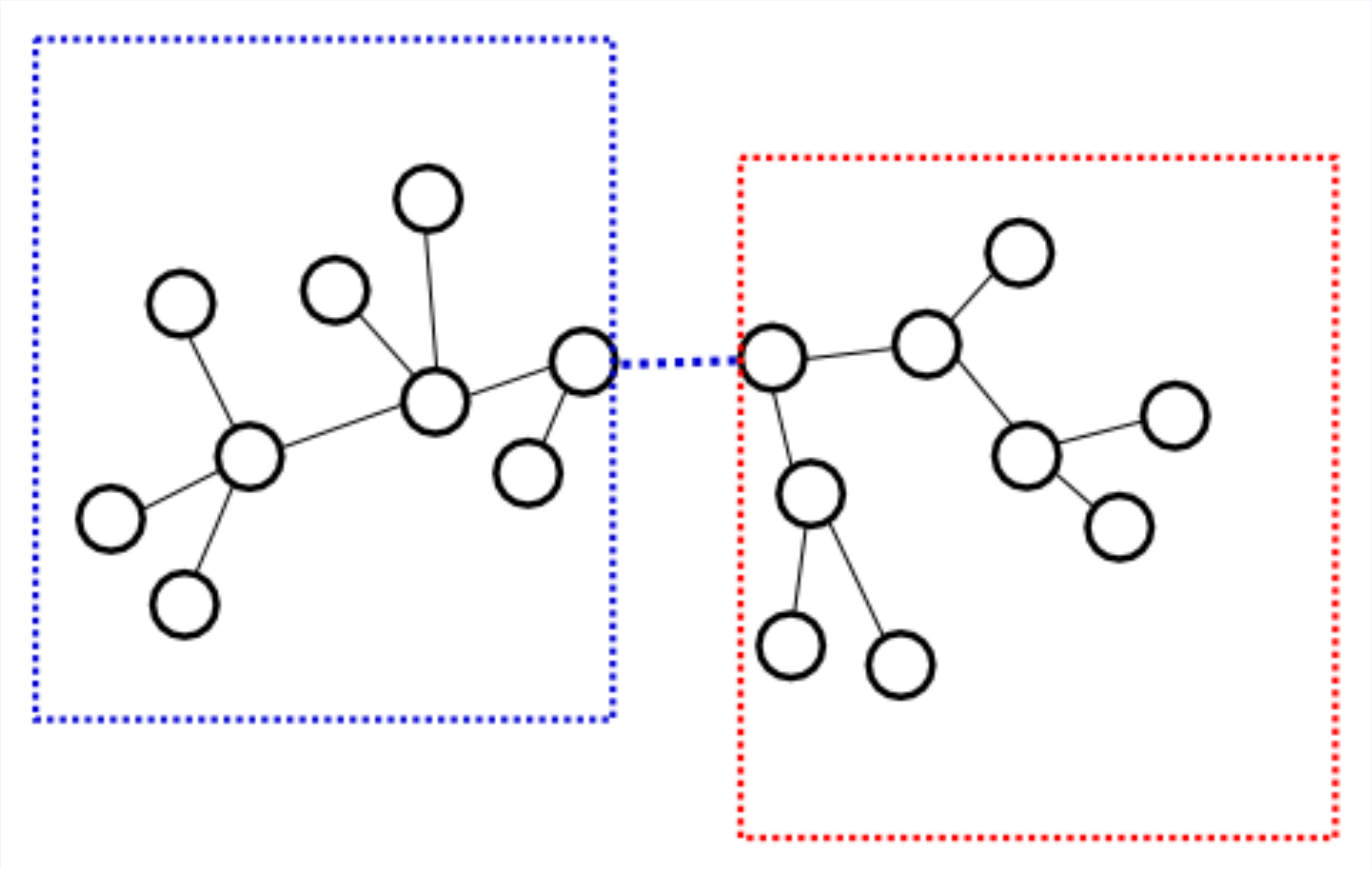
- 어떤 가족이 어떤 도로를 사용할 확률을 구해보자

# 연휴



https://www.acmicpc.net/problem/13254

- 파란 점선으로 나타낸 간선은 트리를 이등분 한다

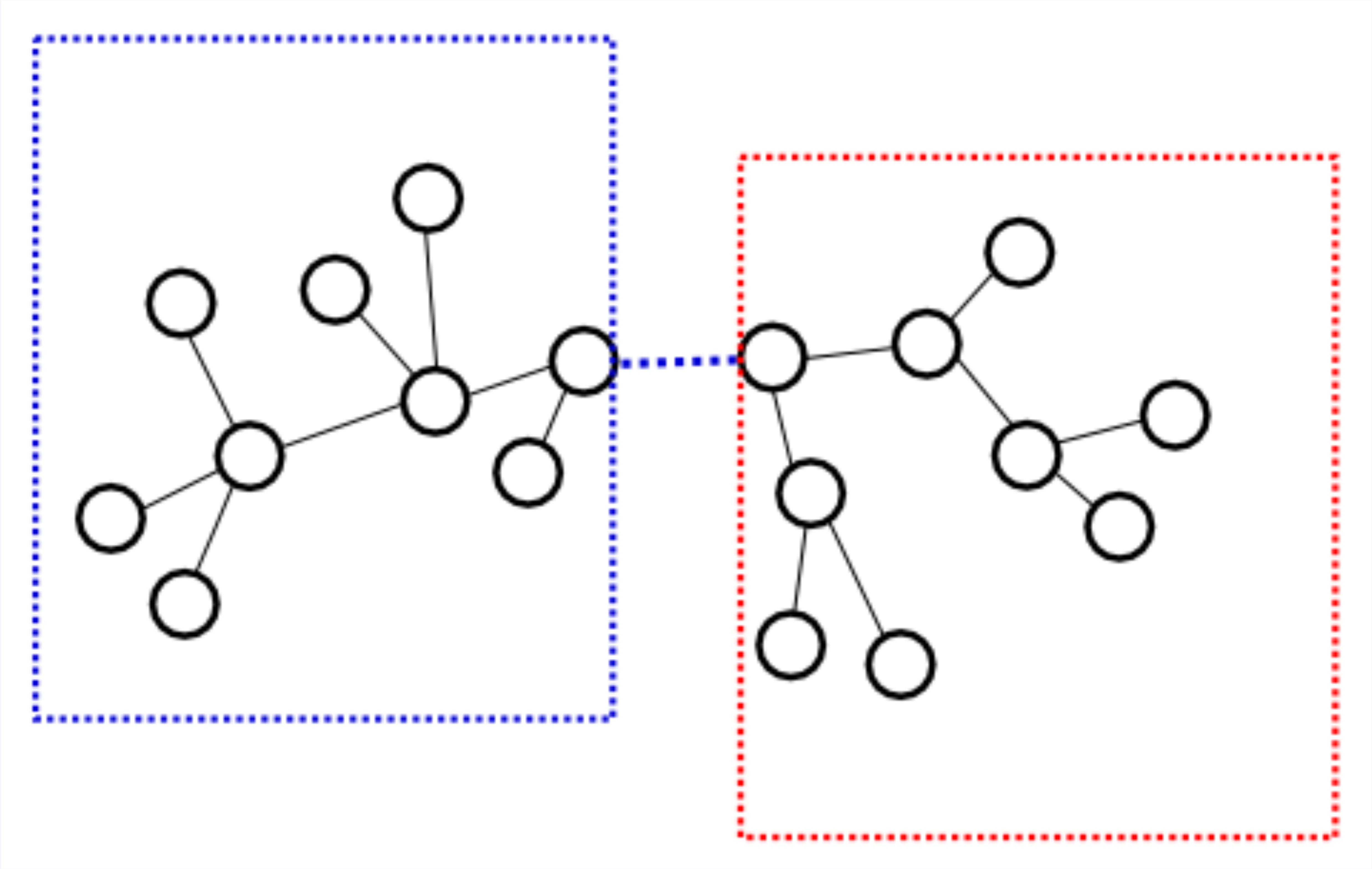


# 연휴



https://www.acmicpc.net/problem/13254

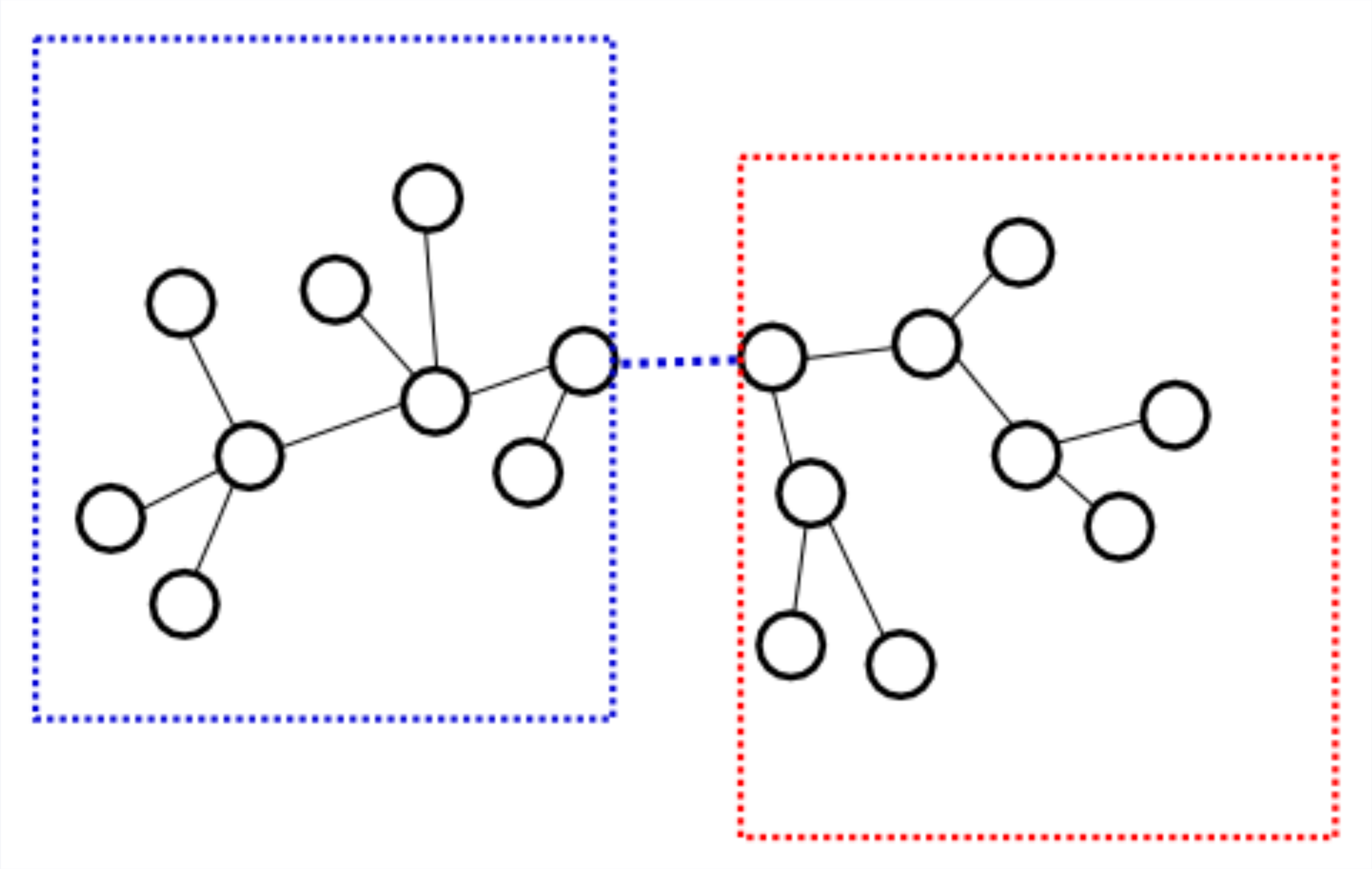
- 같은 쪽에 있는 경로는 파란 점선 간선을 사용하지 않는다
- 집과 선택한 도시가 서로 다른 쪽에 있는 경우만 파란 점선 간선을 사용한다



# 연휴

https://www.acmicpc.net/problem/13254

- 집이 파란색에 있을 때 파란 점선 간선을 사용할 확률
- 빨간색에 있는 정점의 개수 / (N-1)



# 연휴

https://www.acmicpc.net/problem/13254

- 각각의 도로에 대해서 확률을 구한 다음, 모두 더해주면 기댓값이 나온다.

# 연휴



https://www.acmicpc.net/problem/13254

- https://gist.github.com/Baekjoon/f9d380e1411d3e0e6436a7785498d44a

# 동전 뒤집기



https://www.acmicpc.net/problem/13255

- N개의 동전이 탁자 위에 놓여져 있다. 동전은 모두 앞면이 위를 향하고 있다.
- K개의 정수 A[i]가 주어진다. 가장 처음에 A[1]개의 동전을 랜덤하게 골라서 뒤집는다. 그 다음에는 A[2]개의 동전을 랜덤하게 골라서 뒤집는다. 이 과정을 계속해서 반복하고, 마지막에는 A[K]개의 동전을 랜덤하게 골라서 뒤집는다.
- 모든 과정을 완료했을 때, 앞면이 위를 향하는 동전 개수의 기댓값을 구하는 문제

# 동전 뒤집기



https://www.acmicpc.net/problem/13255

- 각각의 동전이 선택될 확률은 모두 같다
- N개의 동전이 있고, 여기서 A[i]개를 골랐을 때, j번째 동전이 선택될 확률은?

# 동전 뒤집기

https://www.acmicpc.net/problem/13255

- 각각의 동전이 선택될 확률은 모두 같다
- N개의 동전이 있고, 여기서 A[i]개를 골랐을 때, j번째 동전이 선택될 확률은?
- C(N-1, A[i]-1) / C(N, A[i]) = A[i] / N
- 즉, 모든 동전이 선택될 확률은 A[i] / N 이다.

# 동전 뒤집기

https://www.acmicpc.net/problem/13255

- 동전이 짝수번 선택되었다면, 앞면을 나타내게 된다
- 즉, 각 동전이 짝수번 선택되는 횟수의 확률을 구해야 한다
- 모든 동전은 구분할 수 없기 때문에
- 각 동전이 짝수번 선택되는 횟수의 확률을 구하고, N을 곱해주면 기댓값이 된다

# 동전 뒤집기

https://www.acmicpc.net/problem/13255

- t단계까지 수행했을 때, 어떤 동전이 짝수번 뒤집혔을 확률을 구해보자
- t = 0인 경우:

# 동전 뒤집기

https://www.acmicpc.net/problem/13255

- t단계까지 수행했을 때, 어떤 동전이 짝수번 뒤집혔을 확률을 구해보자
- t = 0인 경우: 1.0

# 동전 뒤집기

https://www.acmicpc.net/problem/13255

- t단계까지 수행했을 때, 어떤 동전이 짝수번 뒤집혔을 확률을 구해보자
- t = 1인 경우:

# 동전 뒤집기

https://www.acmicpc.net/problem/13255

- t단계까지 수행했을 때, 어떤 동전이 짝수번 뒤집혔을 확률을 구해보자
- t = 1인 경우:
- p = A[1] / N
- 1은 홀수이기 때문에, 선택되지 않아야 짝수번 뒤집힌 것이다. 따라서, (1 – p)가 된다

# 동전 뒤집기

https://www.acmicpc.net/problem/13255

- t단계까지 수행했을 때, 어떤 동전이 짝수번 뒤집혔을 확률을 구해보자
- t-1단계에서 동전이 선택되었을 확률: $p = a[t-1] / N$
- t-1단계에서 동전이 짝수번 뒤집혔을 확률: $q$

# 동전 뒤집기

https://www.acmicpc.net/problem/13255

- t단계까지 수행했을 때, 어떤 동전이 짝수번 뒤집혔을 확률을 구해보자
- t단계에서 동전이 선택되었을 때 (p)
  - t-1단계에서 동전은 홀수번 뒤집혔어야 한다
  - t-1단계에서 동전이 짝수번 뒤집혔을 확률: q
  - t-1단계에서 동전이 홀수번 뒤집혔을 확률: 1 - q
  - 따라서, 짝수번 뒤집혔을 확률: p * (1 – q)

# 동전 뒤집기



https://www.acmicpc.net/problem/13255

- t단계까지 수행했을 때, 어떤 동전이 짝수번 뒤집혔을 확률을 구해보자
- t단계에서 동전이 선택되지 않았을 때 (1 – p)
  - t-1단계에서 동전은 짝수번 뒤집혔어야 한다
  - t-1단계에서 동전이 짝수번 뒤집혔을 확률: q
  - 따라서, 짝수번 뒤집혔을 확률: q * (1 – p)

# 동전 뒤집기

https://www.acmicpc.net/problem/13255

- t단계까지 수행했을 때, 어떤 동전이 짝수번 뒤집혔을 확률을 구해보자
- t단계에서 동전이 선택되었을 확률: $p * (1 - q) + q * (1 - p)$

# 동전 뒤집기



https://www.acmicpc.net/problem/13255

- https://gist.github.com/Baekjoon/7eaa6e3081aad20029a95cfc78e4beb5

# 생태학

https://www.acmicpc.net/problem/13257

- 먼저, 데이터 수집은 D일동안 진행된다
- 초기에 모든 새에 측정기를 부착되어있지 않다
- 데이터 수집이 진행되는 동안 매일 매일 C마리의 새를 잡을 것이다
- 그 다음, C마리의 새 중에 측정기가 부착되어 있지 않은 새에는 측정기를 부착할 것이고, 이미 부착되어 있는 새는 그냥 놔둘 것이다
- 그 다음 하루가 끝날 때, 모든 새를 다시 풀어준다.
- 이터를 수집한 영역의 새의 개체수가 N마리라고 가정했을 때, 데이터 수집 기간이 끝난 후에 측정기가 부착되어 있는 새의 수가 M마리일 확률을 구하는 문제

# 생태학

https://www.acmicpc.net/problem/13257

- D[d][m] = d일이 지난후, m마리의 새에 측정기가 부착되어 있을 확률
- p(add, prev) = 이 영역에 측정기가 부착된 새가 prev마리일 때, add 마리의 새에 측정기가 새로 부착될 확률

# 생태학



https://www.acmicpc.net/problem/13257

- D[d][m] = d일이 지난후, m마리의 새에 측정기가 부착되어 있을 확률
- p(add, prev) = 이 영역에 측정기가 부착된 새가 prev마리일 때, add 마리의 새에 측정기가 새로 부착될 확률
- D[d][m] = p(0, m)*D[d-1][m] + p(1, m-1)*D[d-1][m-1] + p(2, m-2) * D[d-1][m-2] + ⋯ + p(c, m-c)*D[d-1][m-c]

# 생태학



https://www.acmicpc.net/problem/13257

- p(add, prev) = 이 영역에 측정기가 부착된 새가 prev마리일 때, add 마리의 새에 측정기가 새로 부착될 확률
- 하루에 잡은 새의 add마리는 측정기가 부착되지 않은 새, c-add는 측정기가 부착된 새
- 측정기가 부착되지 않은 새의 수: N – prev
- 측정기가 부착된 새의 수: prev

# 생태학

https://www.acmicpc.net/problem/13257

- p(add, prev) = 이 영역에 측정기가 부착된 새가 prev마리일 때, add 마리의 새에 측정기가 새로 부착될 확률
- 하루에 잡은 새의 add마리는 측정기가 부착되지 않은 새, c-add는 측정기가 부착된 새
- 측정기가 부착되지 않은 새의 수: N – prev
- 측정기가 부착된 새의 수: prev
- 측정기가 부착되지 않은 새에서 add마리를 잡아야 하고
- 측정기가 부착된 새에서 c-add 마리를 잡아야 한다.
- C(add, N-prev) * C(c-add, prev)

# 생태학



https://www.acmicpc.net/problem/13257

- p(add, prev) = 이 영역에 측정기가 부착된 새가 prev마리일 때, add 마리의 새에 측정기가 새로 부착될 확률
- 하루에 잡은 새의 add마리는 측정기가 부착되지 않은 새, c-add는 측정기가 부착된 새
- 측정기가 부착되지 않은 새의 수: N – prev
- 측정기가 부착된 새의 수: prev
- 측정기가 부착되지 않은 새에서 add마리를 잡아야 하고
- 측정기가 부착된 새에서 c-add 마리를 잡아야 한다.
- C(N-prev, add) * C(prev, c-add)
- 그냥 새를 c마리 잡는 경우의 수: C(N, c)

# 생태학

https://www.acmicpc.net/problem/13257

- https://gist.github.com/Baekjoon/204291804d54d45b6caabf1007173e36

# 랜덤 소트

https://www.acmicpc.net/problem/1521

- 랜덤 소트는 어떤 순열이 주어졌을 때, i<j이면서 A[i] > A[j]인 임의의 쌍을 교환하는 것이다.
- 입력으로 주어진 순열을 오름차순으로 정렬할 때, 필요한 교환의 횟수의 기댓값을 구하는 문제

# 랜덤 소트

https://www.acmicpc.net/problem/1521

- D[A] = 순열 A를 정렬하는데 필요한 교환 횟수의 기댓값

# 랜덤 소트



https://www.acmicpc.net/problem/1521

```
double go(vector<int> a) {
    if (d.count(a)) return d[a];
    double ans = 0.0; int tot = 0;
    for (int i=0; i<n-1; i++) {
        for (int j=i+1; j<n; j++) {
            if (a[i] > a[j]) {
                swap(a[i], a[j]);
                ans += go(a);
                tot += 1;
                swap(a[i], a[j]);
            }
        }
    }
    if (tot > 0) ans = ans / tot + 1;
    return d[a] = ans;
}
```

# 랜덤 소트

https://www.acmicpc.net/problem/1521

- https://gist.github.com/Baekjoon/53094af37c4e341225be5072f75b1342

# 복권 + 은행

https://www.acmicpc.net/problem/13258

- 은행은 계좌를 가지고 있는 사람들에게 잔고 1원당 티켓을 1개씩 지급한다
- 모든 티켓을 지급한 후에는 티켓 하나를 랜덤하게 고른다
- 모든 티켓이 당첨될 확률은 같다. 당첨된 사람의 계좌에 J원이 즉시 추가된다
- 스타트링크 은행에 계좌를 가지고 있는 사람의 수와 잔고가 주어졌을 때, C주가 지난 후 강호의 통장 잔고의 기댓값을 구하는 문제

# 복권 + 은행



https://www.acmicpc.net/problem/13258

- w주가 지난 후에 은행에 예금된 총 잔고에 w * J 원이 추가된다
- 가장 처음에 예금된 총 잔고가 total원이었다면, w주 후에는 total + w * J가 된다
- w주가 지난 후에 총 티켓의 개수는 total + w * J개가 필요하고
- w주에 강호가 복권에 t번 당첨된 상태라면
- 잔고는 A[0] + t*J원이다.
- w주에 복권에 당첨될 확률은 (A[0] + t*J) / (total + w*J)

# 복권 + 은행



https://www.acmicpc.net/problem/13258

- D[w][win] = w주에 강호가 win번 이겼을 때 강호 잔고의 기댓값
- w주에 복권에 당첨될 확률 p = (A[0] + t*J) / (total + w*J)
- D[w][win] = p * D[w+1][win+1] + (1-p) * D[w+1][win]

# 복권 + 은행



https://www.acmicpc.net/problem/13258

- https://gist.github.com/Baekjoon/2e5faa6bf62a842e02523a6ce8c9fd44

# DP 최적화

# Knuth Optimization



Knuth Optimization

- 점화식: D[i][j] = min(D[i][k] + D[k][j]) + C[i][j] (i < k < j)
- 여기서 C[i][j]: 비용
- Quadrangle inequality: C[a][c] + C[b][d] ≤ C[a][d] + C[b][c] (a ≤ b ≤ c ≤ d)
- Monotonicity: C[b][c] ≤ C[a][d] (a ≤ b ≤ c ≤ d)
- P[i][j] = D[i][j]가 최소 되기 위한 가장 작은 k 일 때
- P][j-1] ≤ P][j] ≤ P][j]

# Knuth Optimization

Knuth Optimization

- O(N^3)을 O(N^2)으로 줄일 수 있다

# 파일 합치기

https://www.acmicpc.net/problem/11066

- 각 장이 쓰여진 파일을 합쳐서 최종적으로 소설의 완성본이 들어있는 한 개의 파일을 만든
- 이 과정에서 두 개의 파일을 합쳐서 하나의 임시파일을 만들고, 이 임시파일이나 원래의 파일을 계속 두 개씩 합쳐서 소설의 여러 장들이 연속이 되도록 파일을 합쳐나가고, 최종적으로는 하나의 파일로 합친다
- 두 개의 파일을 합칠 때 필요한 비용(시간 등)이 두 파일 크기의 합이라고 가정할 때, 최종적인 한 개의 파일을 완성하는데 필요한 비용의 총 합

# 파일 합치기



https://www.acmicpc.net/problem/11066

- 연속된 파일만 합칠 수 있다
- 파일은 2개의 연속된 파일을 합치는 것이다
- N^3 다이나믹을 생각해볼 수 있다.

# 파일 합치기



https://www.acmicpc.net/problem/11066

- D[i][j] = i번째 장부터 j번째 장까지 합쳤을 때, 필요한 최소 비용

- i번째 장부터 k번째 장까지 합친 파일과 k+1번째 장부터 j번째 장까지 합치면 된다
- D[i][j] = D[i][k] + D[k+1][j] + 합치는 비용

# 파일 합치기



https://www.acmicpc.net/problem/11066

- C/C++: https://gist.github.com/Baekjoon/bb044e22a3ef51475bfe57da1cf0dfe0

# 파일 합치기



https://www.acmicpc.net/problem/11066

- D[i][j] = i번째 장부터 j번째 장까지 합쳤을 때, 필요한 최소 비용

- i번째 장부터 k번째 장까지 합친 파일과 k+1번째 장부터 j번째 장까지 합치면 된다
- D[i][j] = D[i][k] + D[k+1][j] + 합치는 비용
- 점화식: D[i][j] = min(D[i][k] + D[k][j]) + C[i][j] (i < k < j)
- 같은 형식이다

# 파일 합치기



https://www.acmicpc.net/problem/11066

- Quadrangle inequality: $C[a][c] + C[b][d] \leq C[a][d] + C[b][c]$ ($a \leq b \leq c \leq d$)
- Monotonicity: $C[b][c] \leq C[a][d]$ ($a \leq b \leq c \leq d$)
- 도 성립한다

# 파일 합치기



https://www.acmicpc.net/problem/11066

- C/C++: https://gist.github.com/Baekjoon/980b7bed3e6d7a0ebe886e7c1dfe0a72

# 문자열 자르기

https://www.acmicpc.net/problem/13260

- 길이가 N인 문자열을 두 조각으로 자르는데 필요한 비용은 N이다.
- 문자열을 잘라야 하는 위치가 주어졌을 때, 문자열을 자르는 비용의 최소값을 구하는 문제
- thisisastringofchars를, 3, 8, 10번 문자 뒤에서 잘라야 하는 경우

# 문자열 자르기

https://www.acmicpc.net/problem/13260

```
thisisastringofchars       (문자열)
thi sisastringofchars      (비용: 20)
thi sisas tringofchars     (비용: 17)
thi sisas tr ingofchars    (비용: 12)
                           총: 49.
```

# 문자열 자르기



https://www.acmicpc.net/problem/13260

```
thisisastringofchars       (문자열)
thisisastr ingofchars      (비용: 20)
thisisas tr ingofchars     (비용: 10)
thi sisas tr ingofchars    (비용: 8)
                           총: 38.
```

# 문자열 자르기



https://www.acmicpc.net/problem/13260

- D[i][j] = i번째 위치부터 j번째 위치까지를 자르는 비용
- D[i][j] = D[i][k] + D[k][j] + A[j]-A[i]

# 문자열 자르기



https://www.acmicpc.net/problem/13260

- https://gist.github.com/Baekjoon/0a99247b7d270aca2ff4a2bf6f47a6c8

# 문자열 자르기

https://www.acmicpc.net/problem/13260

- D[i][j] = i번째 위치부터 j번째 위치까지를 자르는 비용
- D[i][j] = D[i][k] + D[k][j] + A[j]-A[i]
- 점화식: D[i][j] = min(D[i][k] + D[k][j]) + C[i][j] (i < k < j)
- 같은 형식이다

# 문자열 자르기

https://www.acmicpc.net/problem/13260

- Quadrangle inequality: $C[a][c] + C[b][d] \leq C[a][d] + C[b][c]$ $(a \leq b \leq c \leq d)$
- Monotonicity: $C[b][c] \leq C[a][d]$ $(a \leq b \leq c \leq d)$
- 도 성립한다

# 문자열 자르기

https://www.acmicpc.net/problem/13260

- https://gist.github.com/Baekjoon/a7a84a83424c1c6ceb5a1bfce94f539f

# 행렬 곱셈 순서



https://www.acmicpc.net/problem/11049

- 크기가 N×M인 행렬 A와 M×K인 B를 곱할 때 필요한 곱셈 연산의 수는 총 N×M×K번
- 행렬 N개를 곱하는데 필요한 곱셈 연산의 수는 행렬을 곱하는 순서에 따라 다르다
- A의 크기가 5×3이고, B의 크기가 3×2, C의 크기가 2×6인 경우
- (AB)C = 5×3×2 + 5×2×6 = 30 + 60 = 90
- A(BC) = 3×2×6 + 5×3×6 = 36 + 90 = 126

# 행렬 곱셈 순서

https://www.acmicpc.net/problem/11049

- D[i][j] = i번째 행렬부터 j번째 행렬까지 곱했을 때, 곱셈 연산의 최소값
- 행렬의 순서를 바꿀 수 없다

| | | | | |
|---|---|---|---|---|
| i | | k | | j |

- i와 j 사이의 어딘가(k)에서 행렬을 나눠서 곱셈을 해야 한다
- (i~k까지 곱한 행렬) × (k+1~j까지 곱한 행렬)
- D[i][k] + D[k+1][j] + 행렬 곱셈에서 필요한 연산 횟수

# 행렬 곱셈 순서

https://www.acmicpc.net/problem/11049

- D[i][j] = i번째 행렬부터 j번째 행렬까지 곱했을 때, 곱셈 연산의 최소값
- A[i] = i번째 행렬의 크기 (A[i][0] x A[i][1])
- D[i][j] = Min(D[i][k]+D[k+1][j]+A[i][0]*A[k][1]*A[j][1])

# 행렬 곱셈 순서



https://www.acmicpc.net/problem/11049

```
int go(int x, int y) {
    if (d[x][y]) return d[x][y];
    if (x == y) return 0;
    if (x+1 == y) {
        return a[x][0]*a[x][1]*a[y][1];
    }
    int &ans = d[x][y];
    ans = -1;
```

# 행렬 곱셈 순서



https://www.acmicpc.net/problem/11049

```
    for (int k=x; k<=y-1; k++) {
        int t1 = go(x,k);
        int t2 = go(k+1,y);
        if (ans == -1 || ans > t1+t2+a[x][0]*a[k][1]*a[y][1]) {
            ans = t1+t2+a[x][0]*a[k][1]*a[y][1];
        }
    }
    return ans;
}
```

# 행렬 곱셈 순서



https://www.acmicpc.net/problem/11049

- C/C++
  - https://gist.github.com/Baekjoon/45c10837dadd4a2c29eb
- Java
  - https://gist.github.com/Baekjoon/f9995c7e91eea4b300b6

# 행렬 곱셈 순서



https://www.acmicpc.net/problem/11049

- Monotonicity: C[b][c] ≤ C[a][d] (a ≤ b ≤ c ≤ d)
- 가 성립하지 않는다.
- 예를 들어, M1,M2,M3,M4의 크기가 2x3, 3x2, 2x10, 10x1 인 경우에
- M1M2M3의 정답은 (M1M2)M3 인데
- M1M2M3M4의 정답은 M1(M2(M3M4)) 이다

# Divide & Conquer Optimization



Divide & Conquer Optimization

- 점화식: D[i][j] = min(D[i-1][k] + C[k][j]) (k < j)
- 이면서
- P[i][j] = D[i][j]가 최소가 되는 k 일 때
- P[i][j] ≤ P[i][j+1]
- 또는
- Quadrangle inequality
- C[a][c] + C[b][d] ≤ C[a][d] + C[b][c] (a ≤ b ≤ c ≤ d)
- C[a][c] + C[b][d] ≥ C[a][d] + C[b][c] (a ≤ b ≤ c ≤ d)
- 중 하나를 만족

# Divide & Conquer Optimization



Divide & Conquer Optimization

- N = 200인 경우에 D[i][j]에서 i ≤ 3까지를 모두 다 구해놓았다고 치자
- D[4][100]을 구했다고 하자.
- 그럼 P[4][100]도 구할 수 있다.

# Divide & Conquer Optimization



Divide & Conquer Optimization

- N = 200인 경우에 D[i][j]에서 i ≤ 3까지를 모두 다 구해놓았다고 치자
- D[4][100]을 구했다고 하자.
- 그럼 P[4][100]도 구할 수 있다.
- D[4][1] ~ D[4][99]를 구할 때, k는 1 ~ P[4][100]에 있을 것이고
- D[4][101] ~ D[4][200]를 구할 때, k는 P[4][100] ~ n에 있을 것이고

# Divide & Conquer Optimization



Divide & Conquer Optimization

- go(n, l, r, pl, pr)을 D[n][l~r]을 구하는 함수이고, k는 pl~pr에 있다
    1. l == r일 때 예외 처리
    2. m = (l+r) / 2, D[n][m]을 구한다. 이 때, k는 pl ~ pr까지 모두 순회 한다
    3. d(n, l, m-1, pl, P[n][m])
    4. d(n, m+1, r, P[n][m], pr)

# Divide & Conquer Optimization



Divide & Conquer Optimization

- O(KN^2)을 O(KNlgN)으로 줄일 수 있다

# 탈옥



https://www.acmicpc.net/problem/13261

- L개의 칸으로 이루어져 있는 감옥이 있다
- i번 방에 들어있는 죄수의 탈옥력은 C[i]
- 간수는 최대 G명까지 고용할 수 있다
- i번 방에 들어있는 죄수의 탈옥 위험도는
- C[i] * (i번 방을 감시하는 간수가 감시하는 죄수의 수)

# 탈옥



https://www.acmicpc.net/problem/13261

- D[i][j] = i명의 죄수가 j번 감옥까지 감시하고 있을 때, 탈옥 위험도의 최소값

# 탈옥



https://www.acmicpc.net/problem/13261

- D[i][j] = i명의 죄수가 j번 감옥까지 감시하고 있을 때, 탈옥 위험도의 최소값
- i번째 죄수가 k+1번 감옥부터 j번 감옥까지 감시하고 있다고 하자

# 탈옥



https://www.acmicpc.net/problem/13261

- D[i][j] = i명의 죄수가 j번 감옥까지 감시하고 있을 때, 탈옥 위험도의 최소값
- i번째 죄수가 k+1번 감옥부터 j번 감옥까지 감시하고 있다고 하자
- D[i][j] = D[i-1][k] + Cost[k+1][j] $(0 \le k \le j)$

- Cost[k+1][j] = (Sum[j] – Sum[k]) * (j-k)

# 탈옥

https://www.acmicpc.net/problem/13261

- https://gist.github.com/Baekjoon/e011d9e1457a94eab5213f9629e86734

# 탈옥



https://www.acmicpc.net/problem/13261

- $D[i][j] = D[i-1][k] + Cost[k+1][j]$ $(0 \le k \le j)$
- $P[i][j]$ = $D[i][j]$가 최소가 되는 k
- $P[i][j] \le P[i][j+1]$ 이다

# 탈옥

https://www.acmicpc.net/problem/13261

- https://gist.github.com/Baekjoon/b5f5f8349a7bc2b4ace54690499af23f

# 수열의 OR 점수

https://www.acmicpc.net/problem/13262

- 크기가 N인 수열 A와 정수 K가 주어진다.
- 수열을 K개의 그룹으로 나눠야 한다.
- 그룹은 연속되어야 하고, 모든 원소는 한 그룹에 포함되어 있어야 한다.
- 그룹의 점수는 그룹에 속한 수를 모두 OR한 값
- 수열의 점수는 그룹의 점수를 모두 합한 값

# 수열의 OR 점수

https://www.acmicpc.net/problem/13262

- D[i][j] = i개의 수를 j개의 그룹으로 나누었을 때, 그룹 점수의 최대값

# 수열의 OR 점수



https://www.acmicpc.net/problem/13262

- D[i][j] = j개의 수를 i개의 그룹으로 나누었을 때, 점수의 최대값
- D[i][j] = max(D[i-1][k-1] + Cost(k, j))

# 수열의 OR 점수



https://www.acmicpc.net/problem/13262

- D[i][j] = j개의 수를 i개의 그룹으로 나누었을 때, 점수의 최대값
- D[i][j] = max(D[i-1][k-1] + Cost(k, j))
- P[i][j] = D[i][j]가 최대가 되는 k 일 때
- P[i][j] ≤ P[i][j+1]를 만족한다.

# 수열의 OR 점수

https://www.acmicpc.net/problem/13262



- https://gist.github.com/Baekjoon/d727bebe336395362d8aae22c75adedf

# Convex Hull Optimization



Convex Hull Optimization

- D[i] = min(D[j] + B[j]*A[i]) (j < i)
- B[j] ≥ B[j+1], A[i] ≤ A[i+1]

# Convex Hull Optimization



Convex Hull Optimization

- O(N^2)을 O(N)으로 줄일 수 있다

# Convex Hull Optimization

Convex Hull Optimization

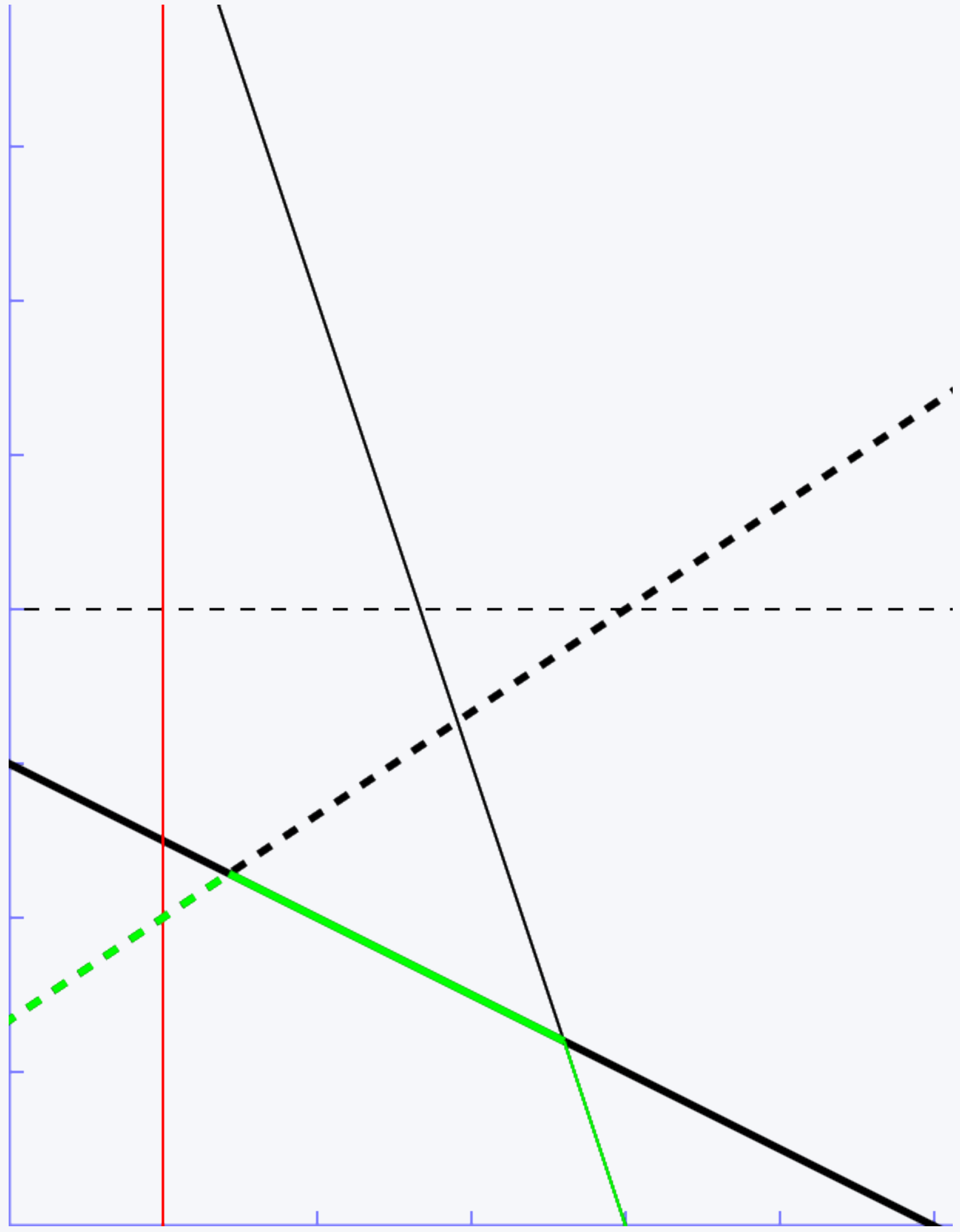
- 선분 y = A[i]*x + B[i]가 여러 개 있다.
- 쿼리: x좌표가 주어졌을 때, f(x)중 가장 작은 값을 찾는 문제

# Convex Hull Optimization



Convex Hull Optimization

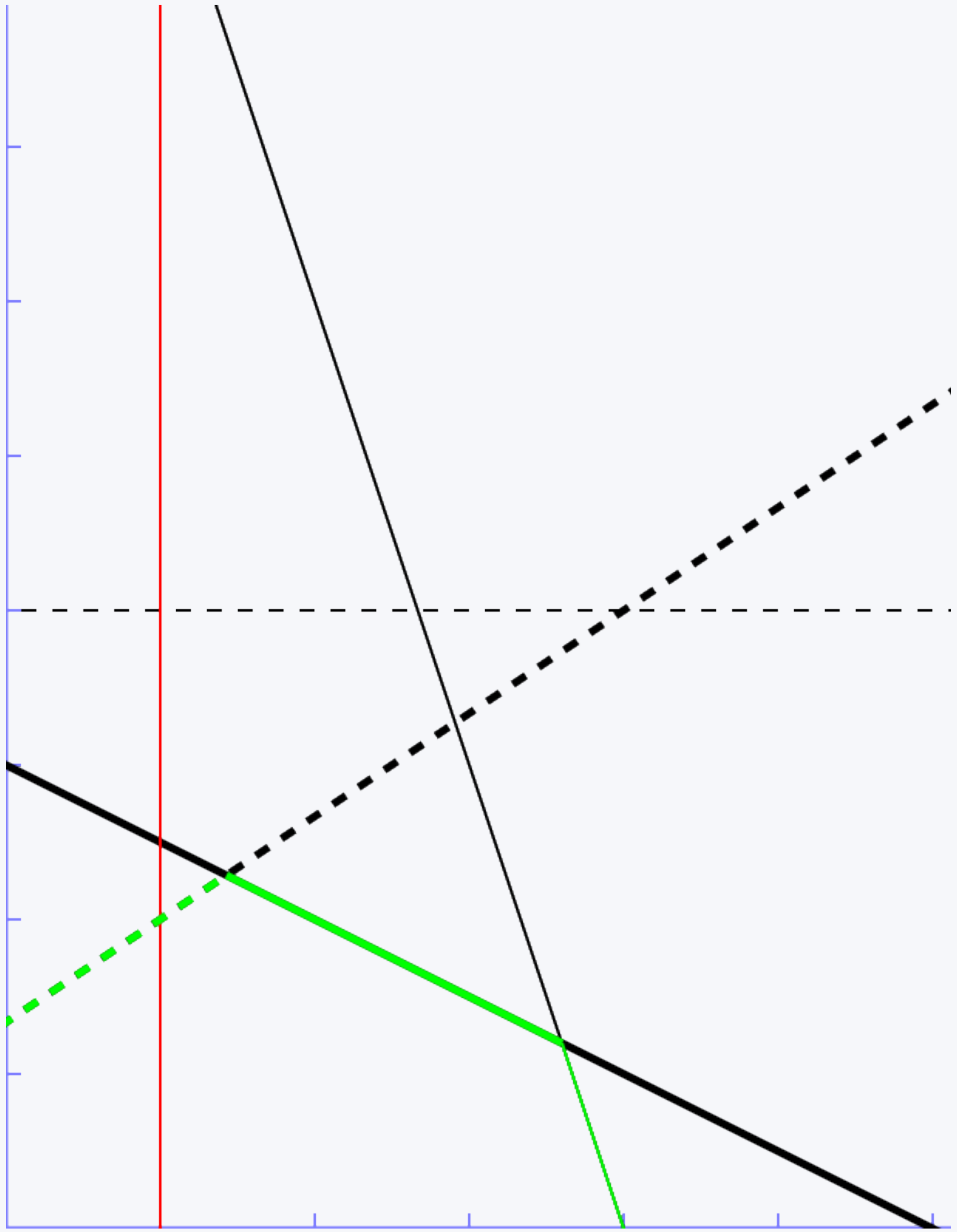
- y=4
- y=2/3x + 4
- y=-3x+12
- y=1/2x+3
- 이 있을 때
- 최소값은 2이다



# Convex Hull Optimization

Convex Hull Optimization

- 선분의 기울기는 2/3, -1/2, -3 순서로 줄어든다



# Convex Hull Optimization



Convex Hull Optimization

- 모든 선분을 기울기가 감소하게 정렬한다
- 그 다음 하나씩 합친다
- 새로운 선분을 추가할 때
- 일부 선분은 더 이상 정답이 될 수 없기 때문에 제거되어야 한다
- 스택을 사용한다

# Convex Hull Optimization



Convex Hull Optimization

- 스택의 가장 윗 두 개 선분을 l1, l2라고 하고 새로 추가할 선분을 l3라고 하면
- l1-l3의 교차점이 l1-l2의 교차점보다 왼쪽에 있으면 l2가 필요 없다

# Convex Hull Optimization



Convex Hull Optimization

- $D[i] = \min(D[j] + B[j]*A[i])$ $(j < i)$
- $B[j] \geq B[j+1]$, $A[i] \leq A[i+1]$
- $f(i) = \min(A[i]*B[j] + D[j])$
- $f(x = A[i]) = \min(B[j]*x + D[j])$

# Convex Hull Optimization



Convex Hull Optimization

- D[0]을 넣는다
- A[1]에서 최소값을 찾는다
- D[1]을 넣는다
- A[2]에서 최소값을 넣는다
- D[2]를 넣는다
- …

# Convex Hull Optimization



Convex Hull Optimization

- D[i] = min(D[j] + B[j]*A[i]) (j < i)
- B[j] $\geq$ B[j+1], A[i] $\leq$ A[i+1]

- D[i] = max(D[j] + B[j]*A[i]) (j < i)
- B[j] $\leq$ B[j+1], A[i] $\leq$ A[i+1]

# 나무 자르기

https://www.acmicpc.net/problem/13263

- 나무의 높이: A[i]
- i번 나무에 전기톱을 사용할 때 마다 그 나무의 높이는 1만큼 감소
- 전기톱은 사용할 때 마다 충전
- 완전히 잘려진 나무의 번호 중 최댓값이 i이면, 전기톱을 충전하는 비용은 B[i]
- 완전히 잘려진 나무가 없다면 전기톱은 충전할 수가 없다
- 모든 나무를 완전히 자르는데 필요한 충전 비용의 최소값

# 나무 자르기

https://www.acmicpc.net/problem/13263

- D[i] = min(D[j], B[j]*A[i]) (j < i)
- A[i] ≤ A[i+1]
- B[i] ≥ B[i+1]

# 나무 자르기



https://www.acmicpc.net/problem/13263

- https://gist.github.com/Baekjoon/7727c72f2b4322bec54b1f59ceec9c5e
- Convex Hull Optimization 사용
- https://gist.github.com/Baekjoon/6cfc671b61895b1807d43effa579f614

# 땅따먹기



https://www.acmicpc.net/problem/6171

- N개의 땅을 사려고 한다 ($1 \leq N \leq 50000$)
- 땅은 W[i] * H[i]이다
- 여러 개의 땅을 살 때는 (해당 땅 중 W[i]의 최대값) * (해당 땅 중 H[i]의 최대값)이 가격이다
- 땅을 모두 사는 비용의 최소값을 구하는 문제

# 땅따먹기



https://www.acmicpc.net/problem/6171

- 어떤 직사각형 A의 너비와 높이가 다른 직사각형 B의 너비와 높이보다 크거나 같으면
- 직사각형 B는 필요가 없다

# 땅따먹기

https://www.acmicpc.net/problem/6171

- 따라서 모든 직사각형의 높이는 증가하게, 그러면서 너비는 감소하게 만들 수 있다.

# 땅따먹기



https://www.acmicpc.net/problem/6171

- 그 다음, 직사각형은 연속해서 나누는 것이 좋다.

# 땅따먹기

https://www.acmicpc.net/problem/6171

- D[i] = i번째 땅까지 샀을 때, 구매 비용의 최소값

# 땅따먹기



https://www.acmicpc.net/problem/6171

- D[i] = i번째 땅까지 샀을 때, 구매 비용의 최소값
- D[i] = min(D[j-1] + A[i].h * A[j].w)

# 땅따먹기



https://www.acmicpc.net/problem/6171

- https://gist.github.com/Baekjoon/36ce084f0d0356915cbe16786e46624f

# 특공대



https://www.acmicpc.net/problem/4008

- 1~N까지 병사 N명을 여러 개의 그룹으로 나눈다.
- 이 때, i~j까지 병사로 이루어진 그룹의 전투력은 x = i번 전투력 ~ j번 전투력까지의 합
- 그룹의 조정된 전투력은 $Ax^2 + Bx + C$ 이다
- 그룹을 적절히 나누어서 조정된 전투력을 최대로 하는 문제

# 특공대

https://www.acmicpc.net/problem/4008

- sum(i, j)를 x[i] + ⋯ + x[j] 로
- adjust(i, j)를 A*sum(i, j)^2 + B*sum(i, j) + C로 한다
- D[i] =

# 특공대

https://www.acmicpc.net/problem/4008



- sum(i, j)를 x[i] + ⋯ + x[j] 로
- adjust(i, j)를 A*sum(i, j)^2 + B*sum(i, j) + C로 한다
- D[i] = max(D[j] + adjust(j+1, i)) (0 ≤ j < i)

# 특공대



https://www.acmicpc.net/problem/4008

- sum(i, j)를 x[i] + ⋯ + x[j] 로
- adjust(i, j)를 A*sum(i, j)^2 + B*sum(i, j) + C로 한다
- D[i] = max(D[j] + adjust(j+1, i)) (0 ≤ j < i)
- sum(i, j) = s[j] – s[i-1] 이라고 한다면
- D[i] = D[j] + adjust(j+1, i)
- D[i] = D[j] + A*sum(j+1,i)^2 + B*sum(j+1,i) + C
- D[i] = D[j] + A*(S[i]-S[j])^2 + B*(S[i]-S[j]) + C
- D[i] = D[j] + A*(S[i]^2 – 2S[i]S[j] + S[j]^2) + B*(S[i]-S[j]) + C
- D[i] = (A*S[i]^2 + B*S[i] + C) + D[j] – 2*A*S[j]S[i] + A*S[j]^2 – B*S[j]

# 특공대

https://www.acmicpc.net/problem/4008

- D[i] = (A*S[i]^2 + B*S[i] + C) + D[j] – 2*A*S[j]S[i] + A*S[j]^2 – B*S[j]
- (A*S[i]^2 + B*S[i] + C)는 그냥 상수이다. 따라서,
- D[i] = max(D[j] – 2*A*S[j]S[i] + A*S[j]^2 – B*S[j]) + (A*S[i]^2 + B*S[i] + C)
- max 안을 S[i]에 대한 다항식으로 나타낼 수 있다.
- (-2*A*S[j])S[i] + (A*S[j]^2 – B*S[j] + D[j])

- i < j라고 하면, 기울기는 증가한다

# 특공대

https://www.acmicpc.net/problem/4008

- https://gist.github.com/Baekjoon/2fc91f3a0526be29c2ab3543cc4feb6d