다이나믹 프로그래밍 4

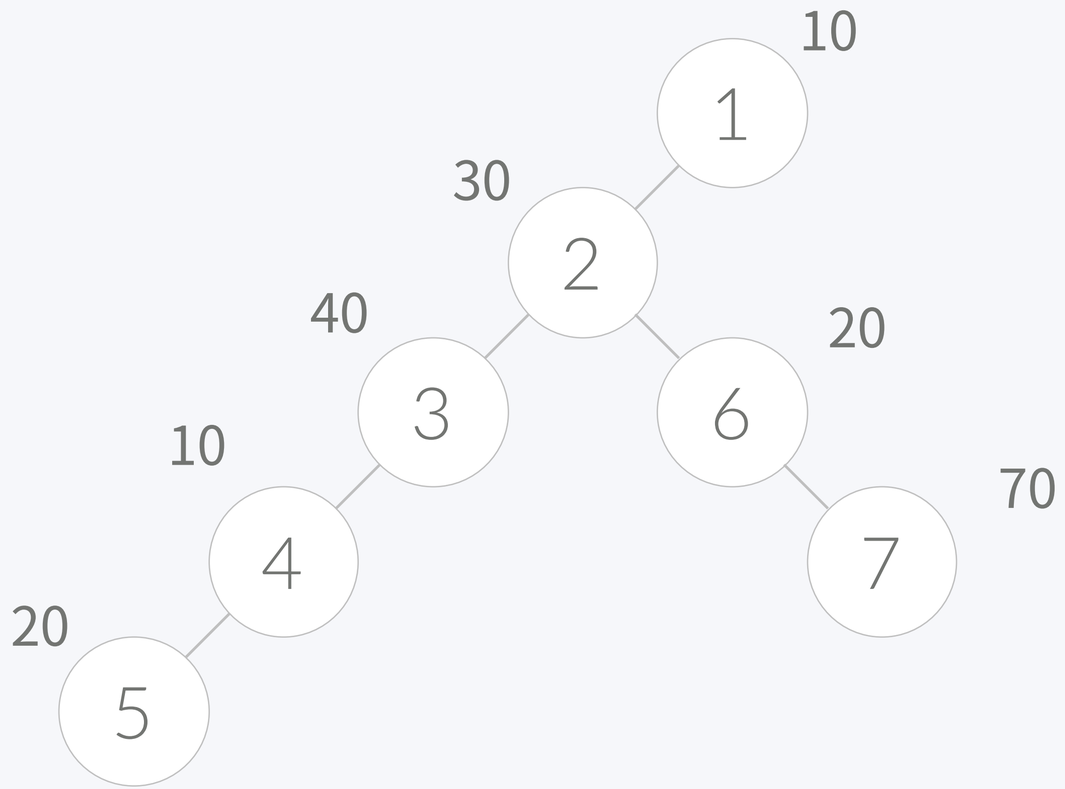
최백준 choi@startlink.io

문제 풀기

트리의 독립집합

https://www.acmicpc.net/problem/2213

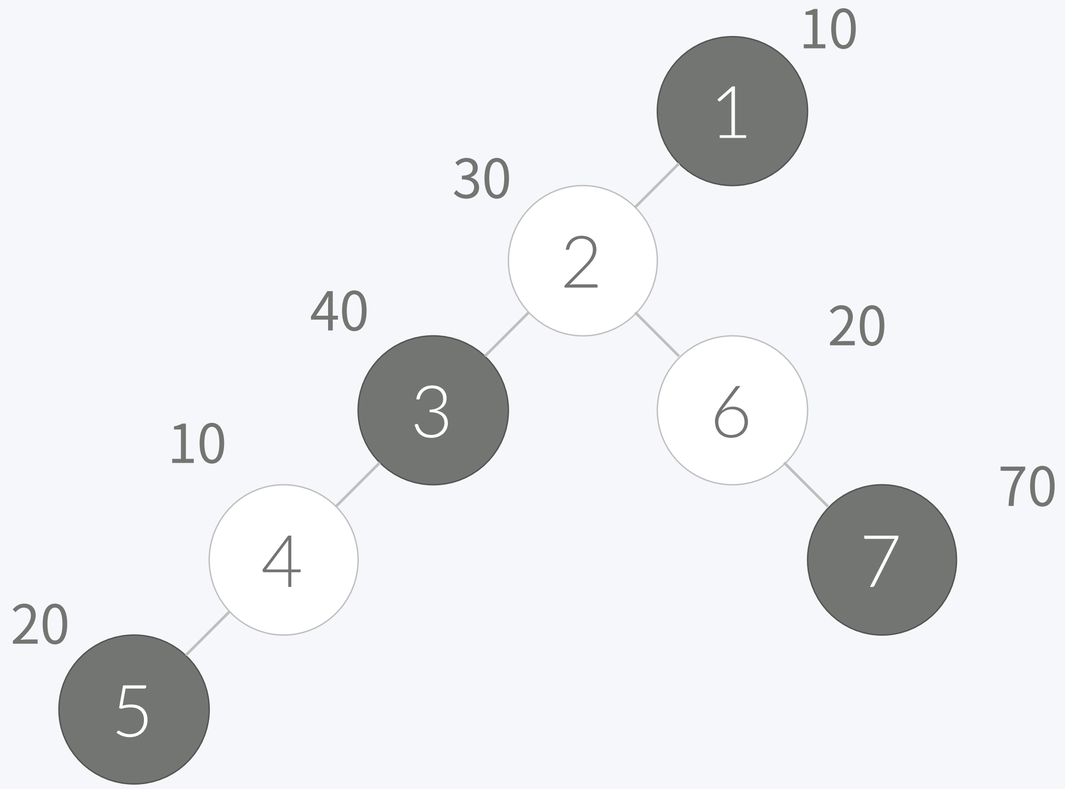
- 트리의 독립집합을 구하는 문제



트리의 독립집합

https://www.acmicpc.net/problem/2213

- 독립집합에 포함되어 있는 정점 끼리는 인접하지 않아야 함



트리의 독립집합

https://www.acmicpc.net/problem/2213

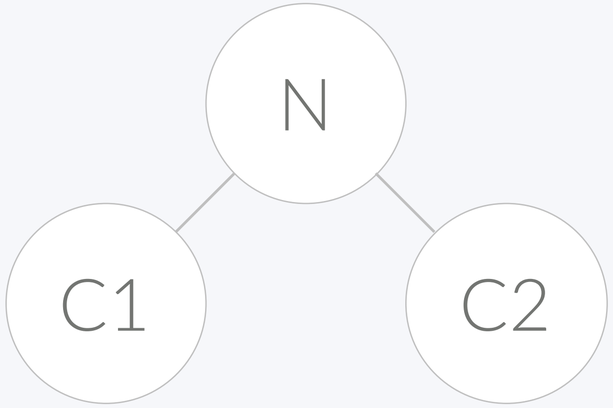
- D[N][0] = N번 정점을 독립집합에 포함시키지 않았을 때, 최대 크기
- D[N][1] = N번 정점을 독립집합에 포함시켰을 때, 최대 크기

- 트리에 루트가 없기 때문에, 1을 루트라고 가정하고 문제를 푼다

트리의 독립집합

https://www.acmicpc.net/problem/2213

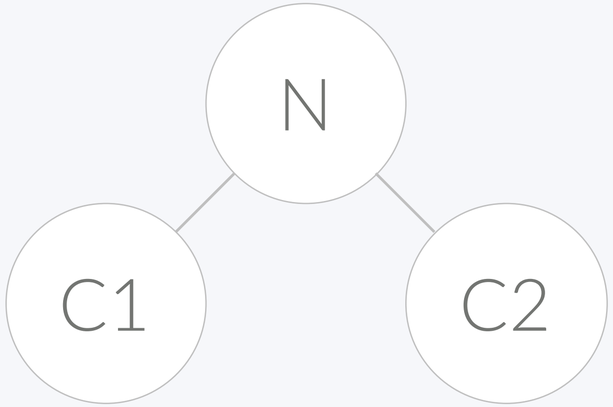
- D[N][0] = N번 정점을 독립집합에 포함시키지 않았을 때, 최대 크기
- N번 정점을 독립집합에 포함시키지 않았기 때문에, 자식을 포함시켜도 되고, 안 포함시켜도 된다
- D[N][0] = Σmax(D[C][0], D[C][1])
- 여기서 C는 N의 자식 정점



트리의 독립집합

https://www.acmicpc.net/problem/2213

- D[N][1] = N번 정점을 독립집합에 포함시켰을 때, 최대 크기
- N번 정점을 독립집합에 포함시켰기 때문에, 자식을 포함시키면 안된다
- D[N][1] = ΣD[C][0] + A[N]
- 여기서 C는 N의 자식 정점



트리의 독립집합

https://www.acmicpc.net/problem/2213

- 최대값을 구하고, 어떻게 선택해야 최대값을 만드는지도 구해야 한다
- 일단 답을 구한 다음, 트리를 순회하면서 어떤 값이 최대값인지를 선택한다

트리의 독립집합

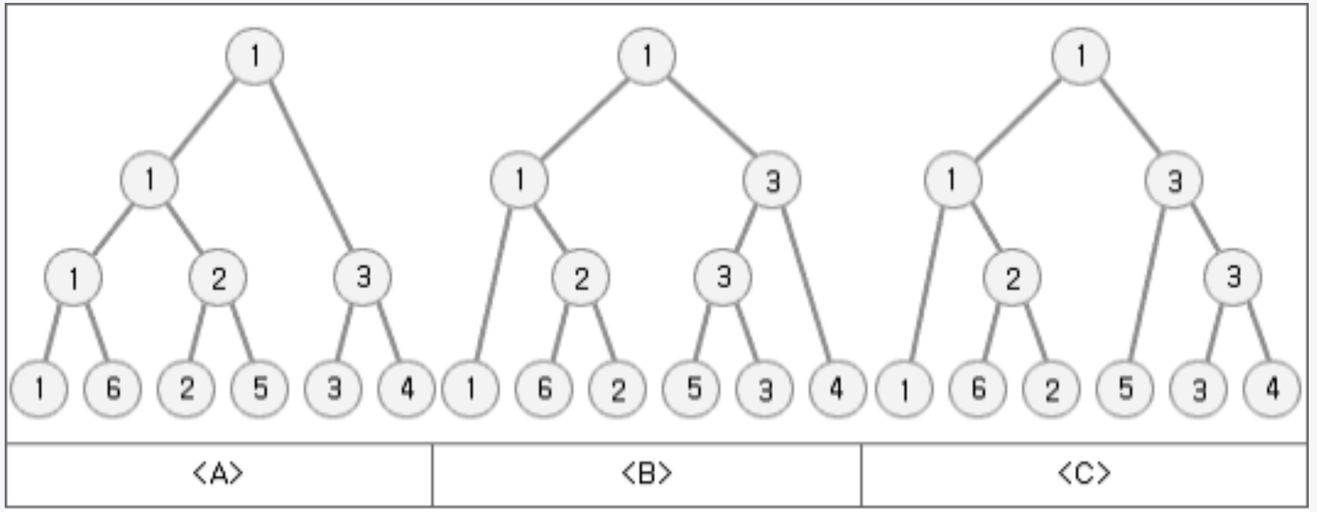
https://www.acmicpc.net/problem/2213

- C/C++: https://gist.github.com/Baekjoon/3fd1210c7b988f44d7e1

토너먼트 만들기

https://www.acmicpc.net/problem/2262

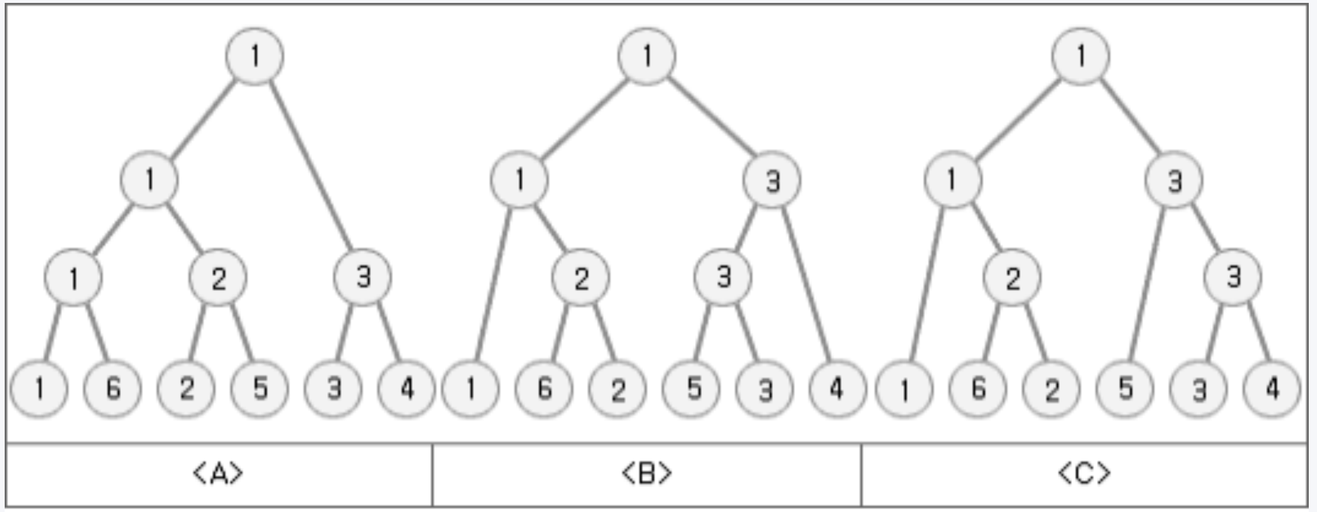
- 랭킹 차이의 합이 최소가 되어야 함
- <A>의 경우는 각 시합이 (1 6), (2 5), (3 4), (1 2), (1 3)으로 랭킹 차이의 합이 5+3+1+1+2=12가 된다. 반면에 <B>는 11이, <C>는 10이 된다.
- D[i][j] = i번부터 j번 사람 까지 토너먼트를 적절히 만들었을 때, 랭킹 차이의 합의 최소값



토너먼트 만들기

https://www.acmicpc.net/problem/2262

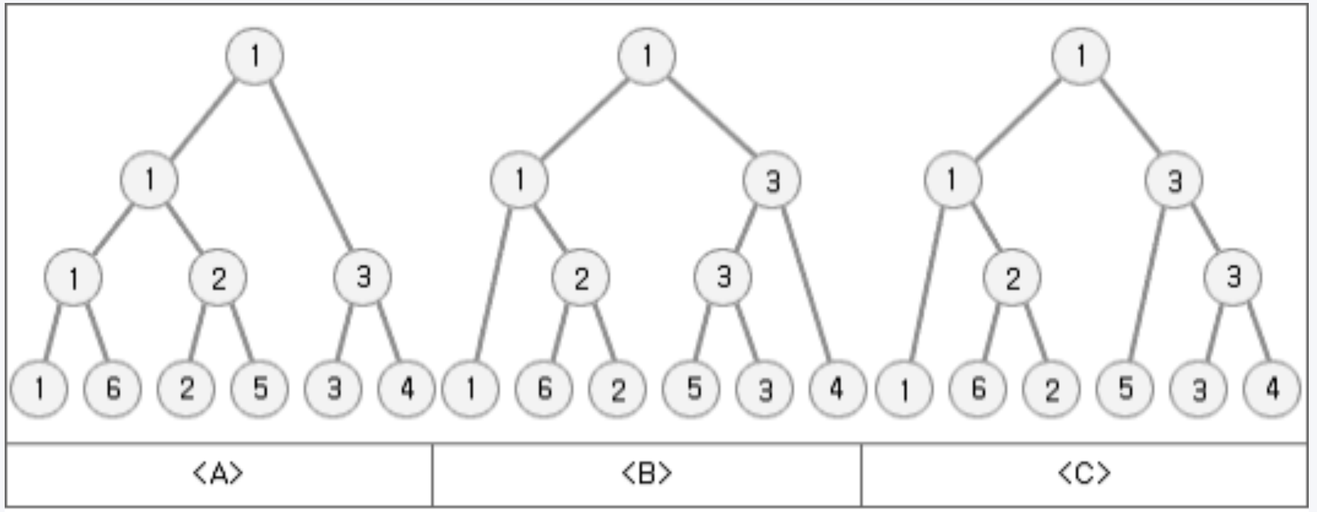
- 토너먼트의 승자는 항상 정해져 있음
- i번 사람부터 j번 사람까지 토너먼트를 한다면, 항상 랭킹이 낮은 사람(숫자가 낮은 사람)이 승리하게 됨
- Winner[i][j] = min(A[i]~A[j])



토너먼트 만들기

https://www.acmicpc.net/problem/2262

- D[i][j] = i번부터 j번 사람 까지 토너먼트를 적절히 만들었을 때, 랭킹 차이의 합의 최소값
- 토너먼트는 항상 두 사람이 한다.
- D[i][j] = MIN(D[i][k] + D[k+1][j] + 랭킹차이)



토너먼트 만들기

https://www.acmicpc.net/problem/2262

- C/C++: https://gist.github.com/Baekjoon/d322829b0277ef9379a3

택배

https://www.acmicpc.net/problem/1866

- 물품 N개
  - A[1], A[2], … , A[N]
- 트럭을 이용할 때, 한 칸 움직이는데 비용이 T라고 하면
  - 비용: A[i] * T
  - 트럭에는 1개만 운송할 수 있음
- 헬리콥터는 한 번 이용할 때 비용이 H
  - 물건, 거리 제한이 없음
  - 헬리콥터로 물건을 X에 내린 다음에, X에서 트럭을 이용해서 배송하는 것도 가능

택배

https://www.acmicpc.net/problem/1866

- 배송 위치: 10, 40, 30, 50, 10
- 트럭을 이용하는 비용: 10
- 헬리콥터를 이용하는 비용: 200
- 모두 트럭을 이용하는 경우
- 10*10 + 40*10 + 30*10 + 50*10 + 10*10 = 1400

택배

https://www.acmicpc.net/problem/1866

- 배송 위치: 10, 40, 30, 50, 10
- 트럭을 이용하는 비용: 10
- 헬리콥터를 이용하는 비용: 200
- 10은 트럭으로, 나머지는 헬리콥터를 이용하는 경우
- 트럭: 10*10 + 10*10
- 헬리콥터: 200
- 이제 40에 물건이 3개 있음. 30과 50으로 트럭을 이용해 이동
  - 10*10 + 10*10
- 총: 10*10 + 10*10 + 200 + 10*10 + 10*10 = 600

택배

https://www.acmicpc.net/problem/1866

- 먼저, 정렬을 한 다음
- D[N] = 1~N번 까지 물건을 배송할 때 최소값

1. N번 물건을 트럭으로 운송하는 경우
2. N번 물건을 헬리콥터로 운송하는 경우
3. N, N-1번 물건을 헬리콥터로 운송하는 경우
4. N, N-1, N-2번 물건을 헬리콥터로 운송하는 경우
5. …
6. N, N-1, …, 1번 물건을 헬리콥터로 운송하는 경우

택배

https://www.acmicpc.net/problem/1866

- 물건 N개를 모두 헬리콥터를 이용해서 운송한다면
- 어느 위치로 운송하는 것이 최소값일까?
- 가운데 위치!
- 관련 문제 2141번 우체국
- https://www.acmicpc.net/problem/2141

택배

https://www.acmicpc.net/problem/1866

- 먼저, 정렬을 한 다음
- D[N] = 1~N번 까지 물건을 배송할 때 최소값

1. N번 물건을 트럭으로 운송하는 경우
2. **N번 물건을 헬리콥터로 운송하는 경우**
3. **N, N-1번 물건을 헬리콥터로 운송하는 경우**
4. **N, N-1, N-2번 물건을 헬리콥터로 운송하는 경우**
5. **…**
6. **N, N-1, …, 1번 물건을 헬리콥터로 운송하는 경우**

택배

https://www.acmicpc.net/problem/1866

- 먼저, 정렬을 한 다음
- D[N] = 1~N번 까지 물건을 배송할 때 최소값

1. N번 물건을 트럭으로 운송하는 경우
2. M번부터 N번 물건을 헬리콥터로 운송하는 경우

택배

https://www.acmicpc.net/problem/1866

- 먼저, 정렬을 한 다음
- D[N] = 1~N번 까지 물건을 배송할 때 최소값

- N번 물건을 트럭으로 운송하는 경우
  - D[N-1] + A[N]*T
- M번부터 N번 물건을 헬리콥터로 운송하는 경우
  - 헬리콥터의 도착 위치 x = A[(M+N)/2]
  - D[M-1] + H + (x에서 트럭으로 배송하는 비용)

택배

https://www.acmicpc.net/problem/1866

- 시간 복잡도 계산
- 정렬 : O(NlgN)
- DP: O(N^3)
  - O(N) = N
    - O(N) = M
      - O(N) = M~N 까지 비용 계산
- 시간 초과
- C/C++: https://gist.github.com/Baekjoon/18cd8537a3042a68a827

택배

https://www.acmicpc.net/problem/1866

- 줄일 수 있는 부분
  - 헬리콥터를 이용한 다음, 트럭으로 배송하는 시간
- i~j까지를 헬리콥터를 이용한 다음, 트럭으로 배송하는 경우
  - 헬리콥터의 도착 위치: A[x] (x = (i+j)/2)
  - (A[j]-A[x])*T
  - (A[j-1]-A[x])*T
  - …
  - (A[x+1]-A[x])*T
  - (A[x]-A[x])*T
  - (A[x]-A[x-1])*T
  - …
  - (A[x]-A[i])*T

택배

https://www.acmicpc.net/problem/1866

- i~j까지를 헬리콥터를 이용한 다음, 트럭으로 배송하는 경우
  - 헬리콥터의 도착 위치: A[x] (x = (i+j)/2)
  - ((A[x]~A[j] 까지 합) – A[x]*(j-x+1))*T
  - **(A[j]-A[x])*T**
  - **(A[j-1]-A[x])*T**
  - …
  - **(A[x+1]-A[x])*T**
  - **(A[x]-A[x])*T**
  - (A[x]-A[x-1])*T
  - …
  - (A[x]-A[i])*T

택배

https://www.acmicpc.net/problem/1866

- i~j까지를 헬리콥터를 이용한 다음, 트럭으로 배송하는 경우
  - 헬리콥터의 도착 위치: A[x] (x = (i+j)/2)
  - ((A[x]~A[j] 까지 합) – A[x]*(j-x+1))*T
  - (A[j]-A[x])*T
  - (A[j-1]-A[x])*T
  - …
  - (A[x+1]-A[x])*T
  - **(A[x]-A[x])*T**
  - **(A[x]-A[x-1])*T**
  - …
  - **(A[x]-A[i])*T**

택배

https://www.acmicpc.net/problem/1866

- Prefix Sum을 이용해 O(N) 처리를 해놓으면
- 합을 O(1)만에 구할 수 있음

택배

https://www.acmicpc.net/problem/1866

- C/C++: https://gist.github.com/Baekjoon/03eca9438e826dcca026

가로등 끄기

https://www.acmicpc.net/problem/2315

- D[i][j] = i~j까지 가로등을 껐을 때, 낭비되는 전력의 최소값

가로등 끄기

https://www.acmicpc.net/problem/2315

- D[i][j][where] = i~j까지 가로등을 껐을 때, 낭비되는 전력의 최소값
  - where = 0이면 마징가는 i에
  - where = 1이면 마징가는 j에
- 마징가가 이동할 수 있는 위치
  - i-1
  - j+1
- 이 때, i~j까지 가로등은 모두 꺼져있는 상태이다.
- 즉, 1~i-1, j+1~N 까지 가로등이 켜져있다.

가로등 끄기

https://www.acmicpc.net/problem/2315

- 왼쪽 가로등을 끄러 가는 경우
- go(left-1, right, 0) + (x[now]-x[left-1])*(s[n]-s[right]+s[left-1])
- 오른쪽 가로등의 경우
- go(left, right+1, 1) + (x[right+1]-x[now])*(s[n]-s[right]+s[left-1])

가로등 끄기

https://www.acmicpc.net/problem/2315

- C/C++: https://gist.github.com/Baekjoon/1fec5373c8ac574f176f

사수아탕

https://www.acmicpc.net/problem/2419

- 사탕은 $x_1, x_2, \cdots, x_n$에 있다.
  - $x_1 \le x_2 \le \cdots \le x_n$
- 수아는 0에 있다.
- 사탕 바구니 k개를 $t_1, t_2, \cdots, t_k$ 라는 시간에 찾았다면
- 총 먹은 사탕의 개수는
- $(m-t_1) + (m-t_2) + \cdots + (m-t_k) = km - (t_1 + t_2 + \cdots + t_k)$
  - $t_1, t_2, \cdots, t_k \le m$
- 여기서 k가 정해져있다면, km도 정해지기 때문에
- $(t_1 + t_2 + \cdots + t_k)$를 최소화 해야 함

사수아탕

https://www.acmicpc.net/problem/2419

- 수아는 항상 왼쪽 또는 오른쪽으로 가장 가까운 사탕 바구니로 간다
- 사탕바구니를 그냥 지나치는 경우는 없다

사수아탕

https://www.acmicpc.net/problem/2419

- L[i][j][k] = 수아가 x_i에서 시작해서, k개의 사탕 바구니를 찾을 때, 못 먹는 사탕의 최소 개수 (i~j에는 사탕 바구니를 이미 찾음)
- R[i][j][k] = L과 같지만 수아가 x_j에서 시작함

사수아탕

https://www.acmicpc.net/problem/2419

- L[i][j][k]
- x_i에서 시작해서 k개 사탕 바구니를 찾아야 함
- 그런데 i, i+1, …, j 에는 사탕 바구니가 없음
- 따라서 x_{i-1} 이나 x_{j+1}로 가야 함
- x_{i-1}로 가는 경우
  - L[i-1][j][k-1]
  - 시간: $x_i - x_{i-1}$
- x_{j+1}로 가는 경우
  - R[i][j+1][k-1]
  - 시간: $x_{j+1} - x_i$

사수아탕

https://www.acmicpc.net/problem/2419

- L[i][j][k]
  - min
  - $L[i-1][j][k-1] + k*(x_i - x_{i-1})$
  - $R[i][j+1][k-1] + k*(x_{j+1} - x_i)$
- R[i][j][k]
  - min
  - $L[i-1][j][k-1] + k*(x_j - x_{i-1})$
  - $R[i][j+1][k-1] + k*(x_{j+1} - x_j)$

사수아탕

https://www.acmicpc.net/problem/2419

- L[i][j][0] = R[i][j][0] = 0;

사수아탕

https://www.acmicpc.net/problem/2419

- 문제를 쉽게 풀기 위해서
- 수아의 처음 위치 0을 x_s로 추가하면 편함
- 이제 다음을 구해줘야 함
- $\max(km - L(s, s, k))$ $(0 \leq k < n)$

사수아탕

https://www.acmicpc.net/problem/2419

- C/C++: https://gist.github.com/Baekjoon/0f98946df205dfd8643b
- 메모리를 N^3만큼 사용하는데
- N^2만큼 사용하게 바꿀 수 있다
- k는 항상 k-1만 참조하기 때문에, k와 k-1만 저장
- C/C++: https://gist.github.com/Baekjoon/6766216b2a404ff77527

Sequence

https://www.acmicpc.net/problem/2291

- N, M, K가 주어졌을 때
- $a_1 \le a_2 \le \cdots \le a_N$ 이고
- $a_1+a_2+\cdots+a_N = M$인 수열 중에서
- 사전 순으로 K번째인 수열 구하기

Sequence

https://www.acmicpc.net/problem/2291

- D[N][M][L] = 길이가 N이고, 합이 M이며, a_1이 L인 수열의 개수
- D[1][i][i] = 1
- D[N][M][L]의 앞에 K 라는 새로운 수 (L ≥ K)가 추가되면
- 합은 M+K, 첫 수는 L이 된다.
- D[N+1][M+K][K] += D[N][M][L]

Sequence

https://www.acmicpc.net/problem/2291

```
for (int i=1; i<=n-1; i++) {
    for (int j=1; j<=m; j++) {
        for (int k=1; k<=m; k++) {
            for (int l=k; l<=m; l++) {
                d[i+1][j+l][l] += d[i][j][k];
            }
        }
    }
}
```

Sequence

https://www.acmicpc.net/problem/2291

- D[N][M][L] = 길이가 N이고, 합이 M이며, a1이 L인 수열의 개수
- D[1][i][i] = 1
- D[N][M][L]에서 2번째 수가 K라면 (M-K ≥ 1, K ≥ L)
- D[N][M][L] += D[N-1][M-L][K]

Sequence

https://www.acmicpc.net/problem/2291

```
for (int i=2; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        for (int k=1; k<=j; k++) {
            for (int l=k; l<=j; l++) {
                if (j-k >= 1) {
                    d[i][j][k] += d[i-1][j-k][l];
                }
            }
        }
    }
}
```

Sequence

https://www.acmicpc.net/problem/2291

- 사전 순으로 K번째를 찾아야 하기 때문에
- 수열의 앞 부터 정답을 찾아야 함
- C/C++: https://gist.github.com/Baekjoon/e110c245a48b14e006a7

Sequence

https://www.acmicpc.net/problem/2291

- D[N][M][L] = 길이가 N, 합이 M이고, L로 시작하는 수열
- D[4][9][1] = 5
  - D[3][8][1] = 3
    - D[2][7][1] = 1
      - D[1][6][6] = 1
    - D[2][7][2] = 1
      - D[1][5][5] = 1
    - D[2][7][3] = 1
      - D[1][4][4] = 1
  - D[3][8][2] = 2
    - D[2][6][2] = 1
      - D[1][4][4] = 1
    - D[2][6][3] = 1
      - D[1][3][3] = 1
  - D[3][8][3] = 0

Sequence

https://www.acmicpc.net/problem/2291

- 이 문제는 2차원으로도 풀 수 있다

Sequence

https://www.acmicpc.net/problem/2291

- D[N][M][L] = 길이가 N, 합이 M이고, L로 시작하는 수열
- D[4][9][1] = 5 **= D[4][9]**
  - D[3][8][1] = **3 = D[3][8]**
    - D[2][7][1] = 1
      - D[1][6][6] = 1
    - D[2][7][2] = 1
      - D[1][5][5] = 1
    - D[2][7][3] = 1
      - D[1][4][4] = 1
  - D[3][8][2] = 2 **= D[3][4]**
    - D[2][6][2] = 1
      - D[1][4][4] = 1
    - D[2][6][3] = 1
      - D[1][3][3] = 1
  - D[3][8][3] = 0

Sequence

https://www.acmicpc.net/problem/2291

- D3[N][M][L] = D2[N][M-N*(L-1)]
- 항상 수열의 시작을 1로 고정하자
- 여기서 길이가 N이고 합이 M인 수열의 앞에 L을 추가한다면
- 수열의 모든 수에 L-1을 더해야 한다.

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

팰린드롬의 개수

https://www.acmicpc.net/problem/1204

- 팰린드롬: 왼쪽에서 부터 읽을 때와 오른쪽에서 부터 읽을 때 같은 문자열 (공백 무시)
- N개의 단어가 주어진다 ($1 \le N \le 50$, $1 \le$ 단어의 길이 ≤ 15, 단어는 중복되지 않음)
- 이 단어를 이용해서 문자열 S를 만들어야 한다
- 각 단어를 여러 번 사용해도 되고, 사용하지 않는 단어가 있어도 된다
- 단어와 단어 사이에는 공백을 하나 집어넣어야 한다
- 같은 단어를 여러 번 사용해도 공백을 집어넣어야 한다
- 문자열 S중에서 팰린드롬이면서 길이가 K이하인 것의 개수를 세는 문제 ($1 \le K \le 100$)
- 빈 문자열은 팰린드롬이 아니고, 이는 공백을 포함한다
- S는 공백으로 시작하거나 끝날 수 없다

팰린드롬의 개수

https://www.acmicpc.net/problem/1204

- N = 2, K = 4
- 단어: z, zz인 경우
- 5가지가 가능하다
- z
- zz
- zz z
- z zz
- z z

팰린드롬의 개수

https://www.acmicpc.net/problem/1204

- 팰린드롬을 양 끝에서부터 중앙으로 만들어야 한다
- 왼쪽 절반과 오른쪽 절반에는 대응하는 글자가 있어야 한다
- 예를 들어, 왼쪽 절반이 abcd이고, 오른쪽 절반이 fedcba인 경우에
- 왼쪽 절반은 이제 ef로 시작해야 한다
- 더 많은 글자를 가지고 있는 것이 왼쪽인지 오른쪽인지를 알아야 하고
- 추가할 수 있는 글자의 개수를 알고 있어야 한다

팰린드롬의 개수

https://www.acmicpc.net/problem/1204

- 사용할 수 있는 글자의 개수: 0~100개
- 추가해야 하는 곳: 2개 (왼쪽 또는 오른쪽)
- 넘어가는 문자열: 50*15개
- 101 * 2 * 50 * 15 = 151500 이므로
- 다이나믹으로 풀 수 있다

팰린드롬의 개수

https://www.acmicpc.net/problem/1204

- 넘어가는 문자열이 팰린드롬인 경우: 팰린드롬을 만든 것이다. 정답에 1을 더함
- 더 이상 사용할 수 글자가 없는 경우: 0을 리턴

팰린드롬의 개수

https://www.acmicpc.net/problem/1204

- 넘어가는 문자열의 모든 부분 문자열에 대해서, 일치하는 단어가 있는지를 확인해야 한다
- 일치하는 단어가 있으면 추가를 하고, 다음 상태를 확인한다

팰린드롬의 개수

https://www.acmicpc.net/problem/1204

- C/C++: https://gist.github.com/Baekjoon/c4c664fceede991b88c2