

# 발전소

<https://www.acmicpc.net/problem/1102>

- $dp[i]$  = 발전소의 상태를  $i$ 로 만들기 위해 필요한 최소 비용
- $i$  : 발전소를 이진수로 나타낸 상태(1 : 켜져 있음)
- $j$  : 상태  $i$ 에서 켜져 있는 발전소이고  $k$  발전소를 켜서  $i$  상태로 추가한다.
- $k$  : 기존에 꺼져 있는 발전소이고  $j$  발전소에 의하여 켜져  $i$ 상태에 추가된다.
- $dp[i] = \min(dp[i \& \sim(1 \ll k)] + A[j][k])$ 
  - ①  $j$ 와  $k$ 가 달라야 한다.
  - ②  $j$ 가  $i$ 에 속해야 한다 :  $(i \& (1 \ll j)) \neq 0$
  - ③  $k$ 가  $i$ 에 속해야 한다 :  $(i \& (1 \ll k)) \neq 0$
- 초깃값 : 켜져있는 발전소를 비트값으로 표시 한다.  
ex) start : YNN  $\rightarrow 001_{(2)}$   
 $dp[start] = 0$
- 비트가  $P$ 개 이상 켜져 있는 것 중 최솟값을 답으로 정한다.

# 발전소

<https://www.acmicpc.net/problem/1102>

```
int solve(int i) {
    if (i == start) return 0;

    if (dp[i] != -1) return dp[i];

    dp[i] = 1e9;
    for (int j = 0; j < N; ++j) {
        for (int k = 0; k < N; ++k) {
            if (j != k && (i & (1 << j)) != 0 && (i & (1 << k)) != 0) {
                dp[i] = min(dp[i], solve(i & ~(1 << k)) + A[j][k]);
            }
        }
    }
    return dp[i];
}
```

Top - Down

```
dp[start] = 0;
for (int i = 0; i < (1 << N); ++i) {
    for (int j = 0; j < N; ++j) {
        for (int k = 0; k < N; ++k) {
            if (j != k && (i & (1 << j)) != 0 && (i & (1 << k)) != 0) {
                dp[i] = min(dp[i], dp[i & ~(1 << k)] + A[j][k]);
            }
        }
    }
}
```

Bottom - Up

