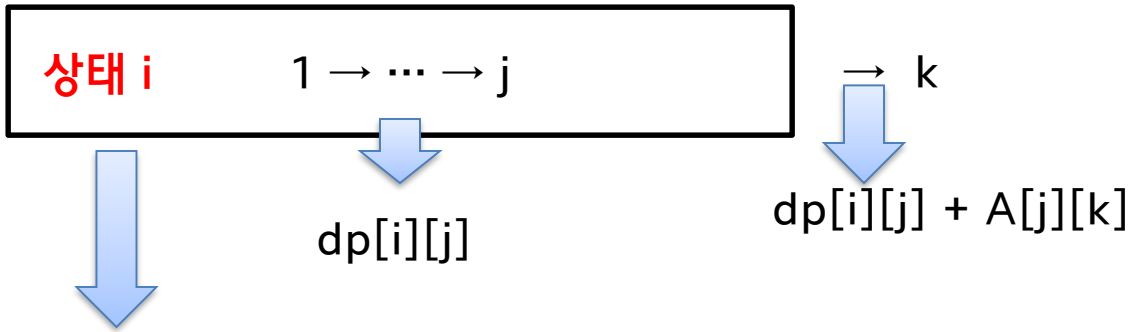
외판원 순회

https://www.acmicpc.net/problem/2098

- dp[i][j] = 도시를 방문한 상태가 i이고, 마지막 방문 도시가 j 일 때 최솟값
- 시작 도시는 1로 고정한다. (1 → 2 → 3 → 1 과 2 → 3 → 1 → 2 는 같다.)
- 1 →4 까지 정답이 1 →2 →3 →4 라고 하면 1 →3 →2 →4는 방문한 집합은 같기 때문에 추후에 고려할 필요가 없는 값이 된다.
  즉, 1 → 5로 갈 때 4에서 가는 경우라면
  1 → 2 → 3 → 4 → 5가 1 → 3 → 2 → 4 → 5 보다 무조건 최솟값이 된다.

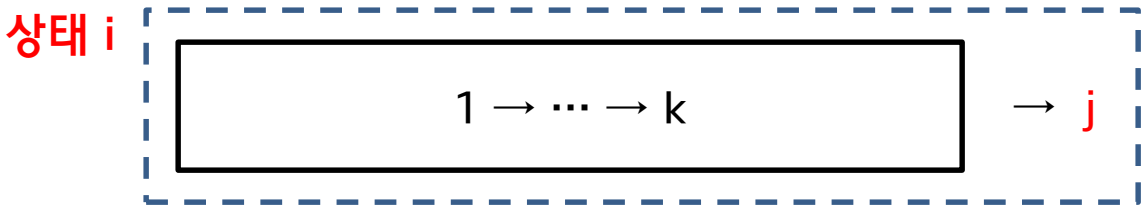


- dp[i | (1<<k)][k] 로 갱신할 수 있다.
  ① s & (1<<j) != 0 : j가 집합 s에 들어가 있어야 한다.
  ② s & (1<<k) == 0 : k가 집합 s에 없어야 한다.
  ③ A[j][k] != 0 : j → k로 이동가능해야 한다.

외판원 순회

https://www.acmicpc.net/problem/2098

- dp[i][j] = 도시를 방문한 상태가 i이고, 마지막 방문 도시가 j 일 때 최솟값



- dp[i][j] = min(dp[s&~(1<<j)][k] + A[k][j]
  ① i & (1<<k) != 0 : k가 집합 s에 속한다.
  ② i & (1<<j) ! = 0 : j가 집합 s에 속한다.
  ③ A[k][j] != 0 : k와 j는 연결되어 있다.
- 초깃값 : dp[(1<<0)][0] = 0, 나머지 MAX
- 정답 : min(dp[(1<<N) -1][i] + A[i][0]) (i=0~N-1)

외판원 순회

https://www.acmicpc.net/problem/2098

```
dp = vii((1 << N) + 10, vi((20), 1e9));
dp[(1 << 0)][0] = 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 && w[j][k] != 0) {
                                dp[i | (1 << k)][k] = min(dp[i | (1 << k)][k], dp[i][j] + w[j][k]);
                        }
                }
        }
}
```

```
dp = vii((1 << N) + 10, vi((20), 1e9));
dp[(1 << 0)][0] = 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)) && (i &(1 << k) && w[k][j])) {
                                dp[i][j] = min(dp[i][j], dp[i & ~(1 << j)][k] + w[k][j]);
                        }
                }
        }
}
```

외판원 순회

https://www.acmicpc.net/problem/2098

```
int solve(int i, int j) {
        if (i == (1 << 0) && j == 0) return 0;

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

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