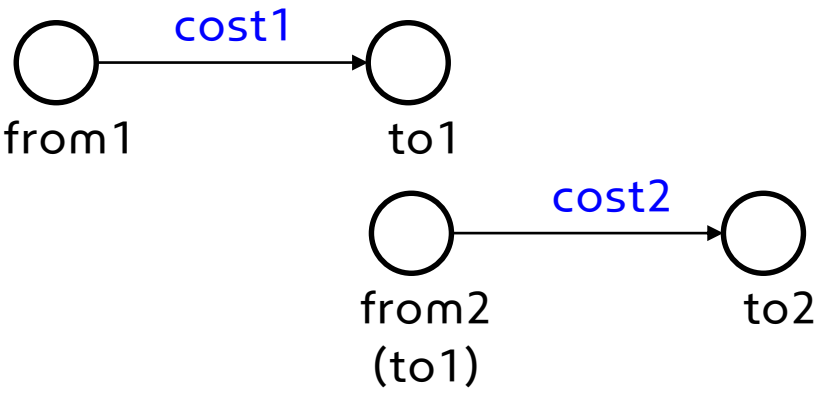
최단 경로 – 벨만포드 + SPFA

[shortest path]

- 벨만 포드의 성능을 향상시킨 알고리즘으로 최악의 경우에는 벨만 포드의 시간복잡도와 같지만 평균적으로 O(E)의 시간 복잡도를 가진다. (벨만 포드 : O(VE))

```
for (int i = 1; i <= N; i++) {
        for (int j = 0; j < M; j++) {
                int u = A[j].from;
                int v = A[j].to;
                int w = A[j].cost;
                if (dist[u] != INF && dist[v] > dist[u] + w) {
                        dist[v] = dist[u] + w;
                        if (i == N) {
                                negative_cycle = true;
                        }
                }
        }
}
```



← ① dist[to1] 값이 update 되지 않았다고 가정

← ② ①에서 update가 되지 않으면 ②에서도 update가 되지 않는다.

Gaussian

최단 경로 – 벨만포드 + SPFA

shortest path

- 벨만 포드는 모든 간선에 대하여 update를 진행하고 SPFA는 update 된 정점과 연결된 간선만 update를 진행한다.
  ☞ 인접 리스트 구현 필요
- 바뀐 정점은 큐를 이용하여 관리하고 큐에 들어가 있는지 아닌지는 배열을 이용하여 체크한다.

```
while (!q.empty()) {
        int from = q.front();
        ck[from] = false;
        q.pop();

        for (int i = 0; i < A[from].size(); ++i) {
                Edge e = A[from][i];
                int to = e.to;
                int cost = e.cost;
                if (dist[to] > dist[from] + cost) {
                        dist[to] = dist[from] + cost;
                        if (ck[to] == false) {
                                q.push(to);
                                ck[to] = true;
                                cnt[to]++;
                                if (cnt[to] >= N) {
                                        printf("-1\n");
                                        return 0;
                                }
                        }
                }
        }
}
```

```
dist[1] = 0;
queue<int> q;
q.push(1);
ck[1] = true;
```