쿠키 마을

용감한 쿠키가 이 마을에서 저 마을로 여행을 가려고 합니다. 이 때 가장 짧은 경로의 거리와 그 경로를 구하시오.

문제

마을의 개수 n과 간선 목록의 개수 m이 주어집니다.
n은 1000 이하, m은 1,000,000 이하입니다.

마을의 번호는 1부터 n까지 할당되어 있습니다.
거리 w는 1 이상의 값을 가집니다.
간선의 목록은 a b w 의 형태로, 출발점 a, 도착점 b, 거리 w 순으로 표현됩니다.

간선 목록이 다 입력된 후, a b 의 형태로 출발지 a와 목적지 b가 입력됩니다.
각 입력마다 최소 거리와 출발지와 목적지까지 거치는 마을을 출력하면 됩니다.
경로가 없어서 도착할 수 없는 경우 0만 출력하면 됩니다.

0이 입력될 경우 프로그램을 종료하면 됩니다.

입출력 예시

5 10 // n m 5 1 3 // a → b = w 1 5 5 4 5 3 1 4 1 1 2 1 2 1 9 2 4 2 4 3 2 3 4 4 2 3 3 // 간선 입력 종료 5 3 // 출발지, 도착지 2 1 1 4 0 // 프로그램 종료	6 5 1 4 3 // 5 3 에 대한 답 (거리 6, 경로 5→1→4→3) 8 2 4 5 1 // 2 1 에 대한 답 (거리 8, 경로 2→4→5→1) 1 1 4 // 1 4 에 대한 답 (거리 1, 경로 1→4)

C / C++ 를 사용하시는 학생 분들은 아래의 폼을 참고해서 작성해 주셔야 기본적인 컴파일 에러를 방지할 수 있습니다.

또한 C 언어의 경우 표준 컴파일러에서는 scanf_s 또는 printf_s 등과 같이 "_s"를 붙이는 경우 컴파일 에러가 발생하기 때문에 "_s"를 제거한 scanf / printf 등의 함수를 사용하시기 바랍니다.

C:

```
#include <stdio.h>

int main() {
    /* TODO */

    return 0;
}
```

C++:

```
#include <iostream>
using namespace std;

int main() {
    /* TODO */

    return 0;
}
```

Floyd의 알고리즘 (1)

- 알고리즘:

```
void floyd(int n, const number W[][], number D[][]) {
  int i, j, k;
  D = W;
  for(k=1; k <= n; k++)
    for(i=1;  i <= n; i++)
      for(j=1; j <= n; j++)
        D[i][j] = minimum(D[i][j], D[i][k]+D[k][j]);
}
```

$d^{k}_{ij} \leftarrow \min\{d^{k-1}_{ij}, d^{k-1}_{ik} + d^{k-1}_{kj}\};$

- 모든 경우를 고려한 분석:
  - ✓ 단위연산: for-*j* 루프안의 지정문
  - ✓ 입력크기: 그래프에서의 정점의 수 *n*

$$T(n) = n \times n \times n = n^3 \in \Theta(n^3)$$

Floyd의 알고리즘 (2)

- 문제: 가중치 포함 그래프의 각 정점에서 다른 모든 정점까지의 최단 거리를 계산하고, 각각의 최단경로를 구하라.
- 입력: 가중치 포함 방향성 그래프 *W*와 그 그래프에서의 정점의 수 *n*.
- 출력: 최단경로의 길이가 포함된 배열 *D*, 그리고 다음을 만족하는 배열 *P*.

$$P[i][j] = \begin{cases} v_i\text{에서 } v_j \text{ 까지 가는 최단경로의 중간에 놓여 있는 정점이 최소한 하나는 있는 경우} \rightarrow \text{그 놓여 있는 정점 중에서 가장 큰 인덱스} \\ \text{최단경로의 중간에 놓여 있는 정점이 없는 경우} \rightarrow 0 \end{cases}$$

Floyd의 알고리즘 (3)

- 알고리즘:

```
void floyd2(int n, const number W[][],
       number D[][], index P[][]) {
  index i, j, k;
  for(i=1; i <= n; i++)
    for(j=1; j <= n; j++)
       P[i][j] = 0;
  D = W;
  for(k=1; k<= n; k++)
    for(i=1; i <= n; i++)
       for(j=1; j<=n; j++)
         if (D[i][k] + D[k][j] < D[i][j]) {
            P[i][j] = k;
            D[i][j] = D[i][k] + D[k][j];
         }
}
```


Floyd의 알고리즘 (4)

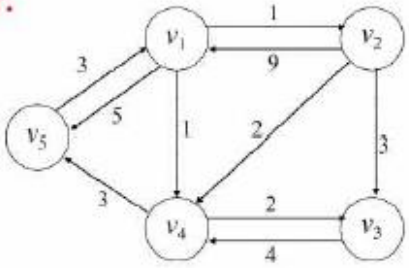
- 앞의 예를 가지고 *D*와 *P*를 구해 보시오.

$P[i][j]$	1	2	3	4	5
1	0	0	4	0	4
2	5	0	0	0	4
3	5	5	0	0	4
4	5	5	0	0	0
5	0	1	4	1	0

Floyd의 알고리즘 (5)

- **최단경로 상에 놓여 있는 정점을 출력하라.**
- 알고리즘:

```
void path(index q,r) {
 if (P[q][r] != 0) {
    path(q,P[q][r]);
    cout << " v" << P[q][r];
    path(P[q][r],r);
 }
}
```



- 위의 P를 가지고 path(5,3)을 구해 보시오.

```
path(5,3) = 4
    path(5,4) = 1
        path(5,1) = 0
        v1
        path(1,4) = 0
    v4
    path(4,3) = 0
```

$P[i][j]$	1	2	3	4	5
1	0	0	4	0	4
2	5	0	0	0	4
3	5	5	0	0	4
4	5	5	0	0	0
5	0	1	4	1	0

결과: v1 v4.

즉, v_5에서 v_3으로 가는 최단경로는 v_5, v_1, v_4, v_3이다.