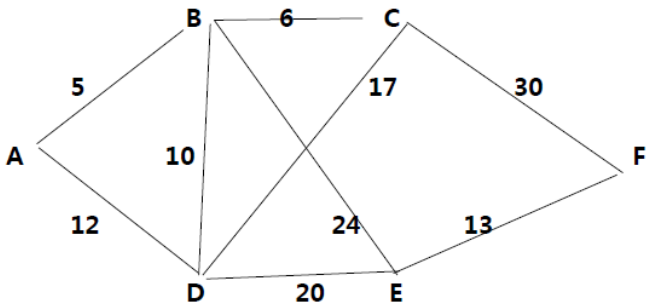
# Lab10 Prim's Algorithm (Minimal Spanning Tree)

1. Input data 는 다음 그래프들을 사용할 것 (**Adjacency Matrix 로 저장**)




2. 참고

1) main()

- print weighted Graph()
- call prim(A) // Starting vertex A

// 알고리즘 강의노트 참조

**3. Output 은 다음과 같다. (정점 A 에서 시작)**

1) Weighted Graph 를 다음과 같이 출력할 것

| | **A** | **B** | **C** | **D** | **E** | **F** |
|---|---|---|---|---|---|---|
| **A** | 0 | 5 | 100 | 12 | 100 | 100 |
| **B** | 5 | 0 | 6 | 10 | 24 | 100 |
| **C** | 100 | 6 | 0 | 17 | 100 | 30 |
| **D** | 12 | 10 | 17 | 0 | 20 | 100 |
| **E** | 100 | 24 | 100 | 20 | 0 | 13 |
| **F** | 100 | 100 | 30 | 100 | 13 | 0 |

2) Minimal Spanning Tree (Greedy algorithm)

**MST: (A,B) → (B,C) → (C,D) → (D,E) → (E,F)**

**Weight: 61**

3) Revised Prim's Algorithm (A 에서 시작, 모든 후보군 함께 고려함)

**MST: (A,B) → (B,C) → (B,D) → (D, E) → (E, F)**

**Weight: 54**

*** Extra 5 Points (Kruskal's Algorithm)**

**1. 위 그래프를 데이터로 이용할 것 (Adjacency List 로 저장)**

2. Sorting algorithm 은 임의의 알고리즘 사용가능

3. Kruskal's 알고리즘 (강의 노트 참조)

4. 실행결과 (Output)

**Original Data:**

A 5 B, B 6 C, A 12 D, B 10 D, C 17 D
B 24 E, D 20 E, C 30 F, E 13 F

**Sorted Data:**

A 5 D, B 6 C, B 10 D A 12 D E 13 F,
C 17 D, D 20 E, B 24 E, C 30 F

**Kruskal's MST:**

1) Edge1: A 5 B
2) Edge2: B 6 C
3) Edge3: B 10 D
4) Edge4: E 13 F
5) Edge5: D 20 E.

**Total Cost: 54**