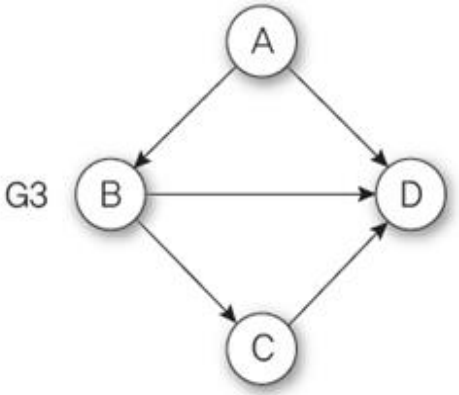
자료구조
그래프구조

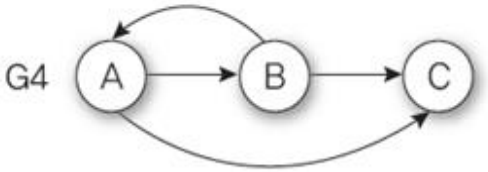
B711063 민경호

그래프란?

- 정점과 간선으로 이루어진 자료구조
- 정점(Vertex) : 데이터가 저장되는 노드
- 간선(Edge) : 노드들을 연결하는 선

그래프의 종류
방향 그래프





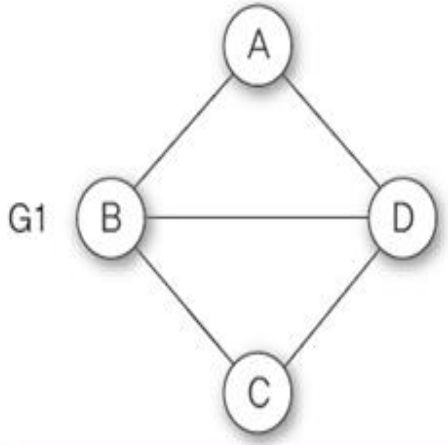
V(G3) = {A, B, C, D}
E(G3) = {<A, B>, <A, D>, <B, C>, <B, D>, <C, D>}

V(G4) = {A, B, C}
E(G4) = {<A, B>, <A, C>, <B, A>, <B, C>}

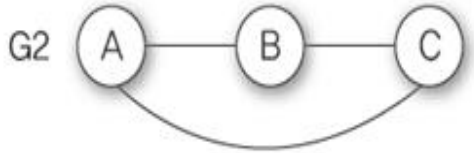
간선에 방향이 있는 그래프

Vi->Vj를 <Vi, Vj>로 표현

그래프의 종류
무방향 그래프



V(G1) = {A, B, C, D}
E(G1) = {(A, B), (A, D), (B, C), (B, D), (C, D)}

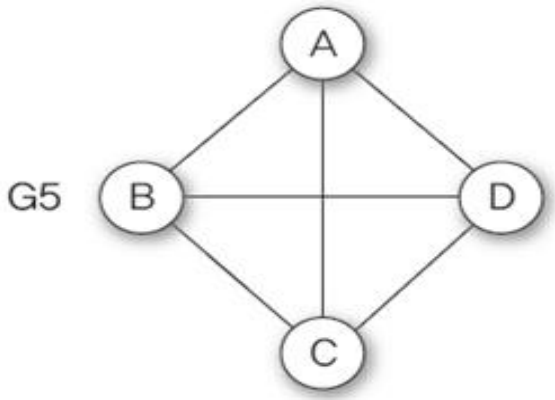


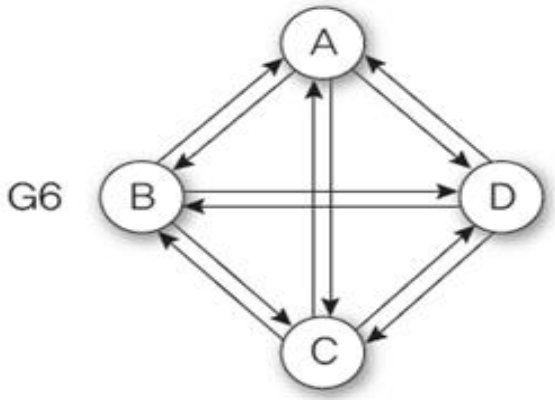
V(G2) = {A, B, C}
E(G2) = {(A, B), (A, C), (B, C)}

두 정점을 연결하는 간선에 방향이 없는 그래프

(Vi, Vj)로 표현

그래프의 종류
완전 그래프



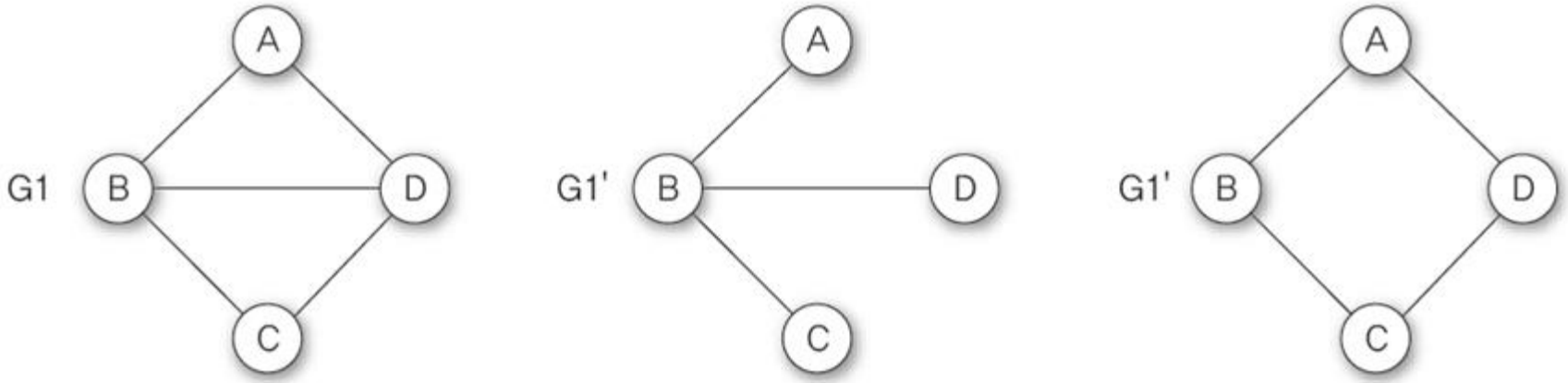


무방향 : n(n-1)/2

방향 : n(n-1)

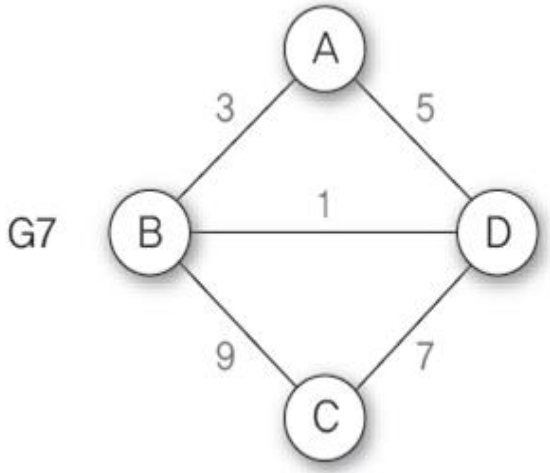
각 정점에서 다른 모든 정점이 연결된, 최대로 많은 간선 수를 가진 그래프

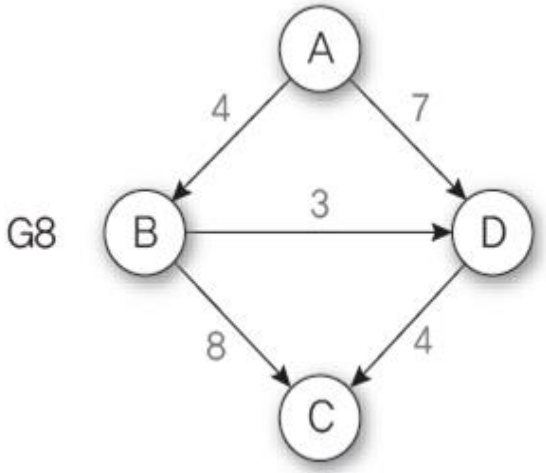
그래프의 종류
부분 그래프



원래 그래프에서 정점이나 간선을 일부만 제외하여 만든 그래프

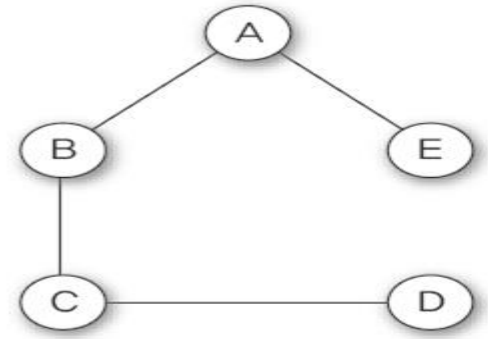
그래프의 종류
가중치 그래프



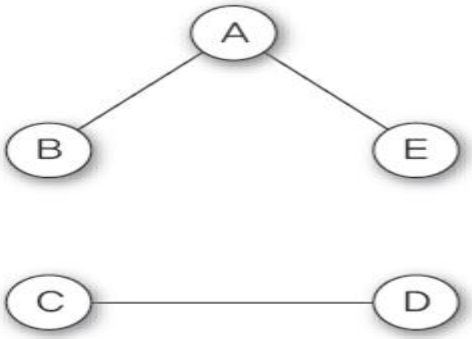


간선에 가중치를 할당한 그래프
=네트워크

그래프의 종류
연결 그래프 / 단절그래프



(a) 연결 그래프



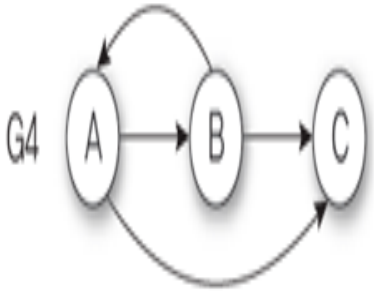
(b) 단절 그래프

연결그래프 : 동떨어진 정점없이 모든 정점들 사이에 경로가 있는 그래프

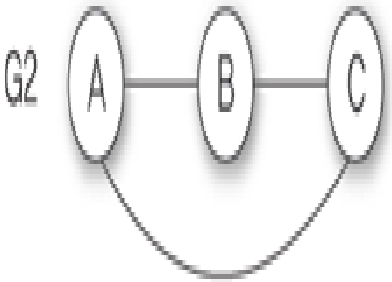
단절그래프 : 연결되지 않은 정점이 있는 그래프

그래프 구현방법

인접행렬 방식



	A	B	C
A	0	1	1
B	1	0	1
C	0	0	0



	A	B	C
A	0	1	1
B	1	0	1
C	1	1	0

그래프의 두 정점을 연결한 간선의 유무를 행렬로 저장하는 방식

그래프 구현방법
인접행렬 방식

```
typedef char VtxData;                  // 그래프 정점에 저장할 데이터의 자료형
int adj[MAX_VTXS][MAX_VTXS];           // 인접 행렬
int vsize;                             // 전체 정점의 개수
VtxData vdata[MAX_VTXS];               // 정점에 저장할 데이터 배열
```

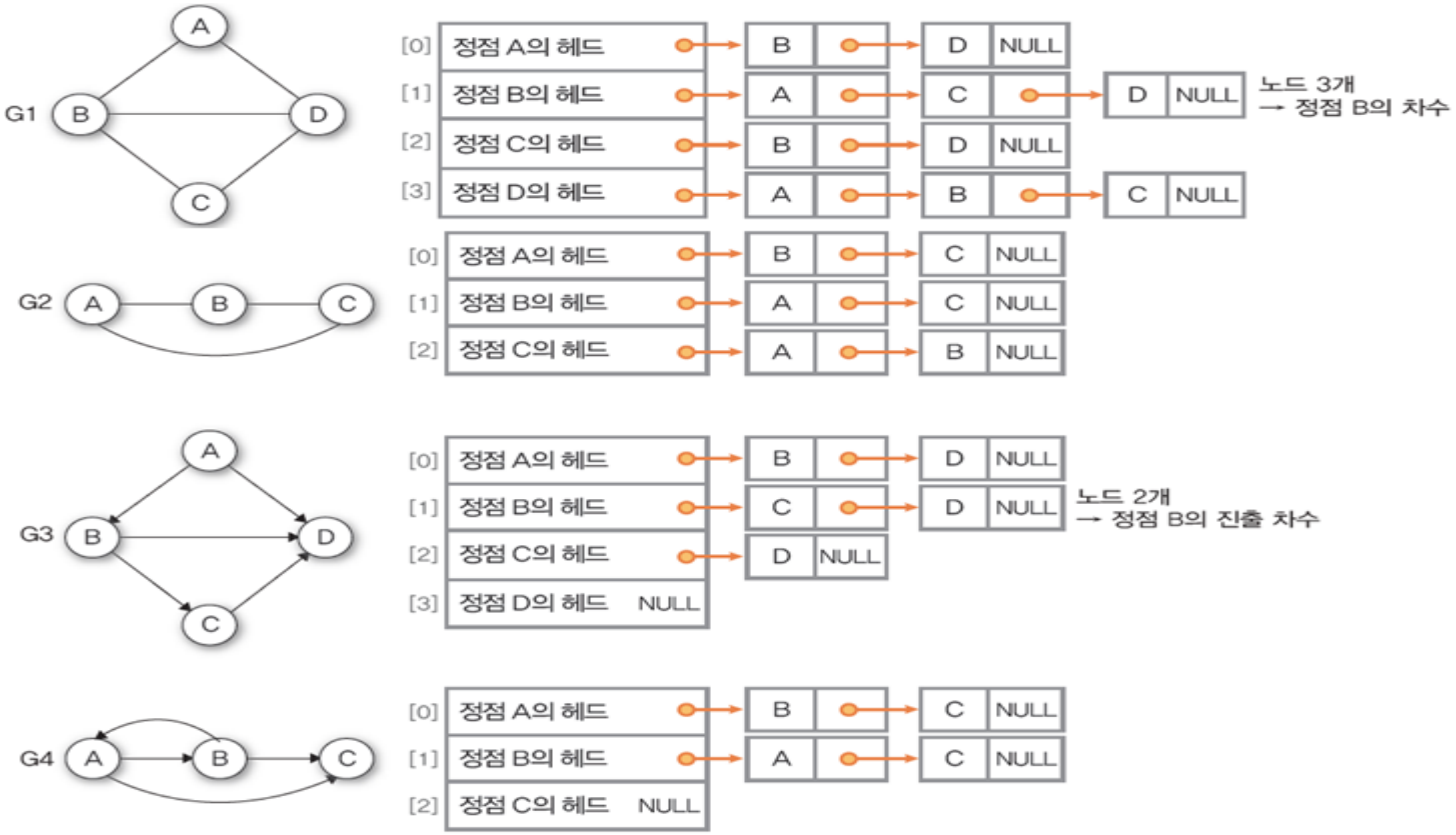
```
void init_graph() {
  int i, j;
  vsize=0;
  for(i=0 ; i<MAX_VTXS; i++)
  for(j=0 ; j<MAX_VTXS; j++)
    adj[i][j] = 0;
}
void insert_vertex( char name ) {
  if (is_full_graph())
    error("Error: 정점 개수 초과\n");
  else
    vdata[vsize++] = name;
}
```

그래프 구현방법

인접행렬 방식

```
void insert_edge(int u, int v, int val)
{
  adj[u][v] = val;
}
void insert_edge2(int u, int v, int val)
{
  adj[u][v] = adj[v][u] = val;
}
```

그래프 구현방법
인접리스트 방식

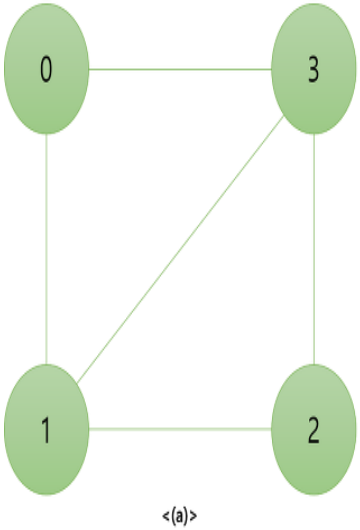


```
//노드를 정의할 구조체
typedef struct GraphNode {
	int vertex; //정점
	struct GraphNode* link;
}GraphNode;

//노드가 저장되는 리스트를 정의할 구조체
typedef struct GraphType {
	int n; //정점의 개수
	GraphNode* adj_list[MAX_VERTICES];
}GraphType;
```

```
//그래프 초기화
void init(GraphType* g) {
	int v;
	g->n = 0;
	for (v = 0; v < MAX_VERTICES; v++) {
		g->adj_list[v] = NULL;
	}
}

//정점 삽입 연산
void insert_vertex(GraphType* g, int v) {
	if (((g->n) + 1) > MAX_VERTICES) {
		cout << "그래프 최대 정점 개수 초과!\n";
		return;
	}
	g->n++;
}

//간선 삽입 연산, v를 u의 인접 리스트에 삽입함
void insert_edge(GraphType* g, int u, int v) {
	GraphNode* node;
	//정점 u의 번호나 정점 v의 번호가 그래프 정점의 개수 이상일 때
	//그래프의 정점의 개수가 n이면 간선의 개수는 n-1이다.
	if (u >= g->n || v >= g->n) {
		cout << "그래프 정점 번호 오류!\n";
	}

	//새로운 노드 동적 생성, node에는 시작 메모리 주소가 저장
	node = (GraphNode*)malloc(sizeof(GraphNode));
	node->vertex = v;
	node->link = g->adj_list[u];
	g->adj_list[u] = node;
}
```

인접행렬, 인접리스트의 장/단점

<인접행렬>



	[0]	[1]	[2]	[3]
[0]	0	1	0	1
[1]	1	0	1	1
[2]	0	1	0	0
[3]	1	1	1	0

<인접리스트>