자료구조
-그래프-

https://youtu.be/7d86AY2kbss

엄 시 우

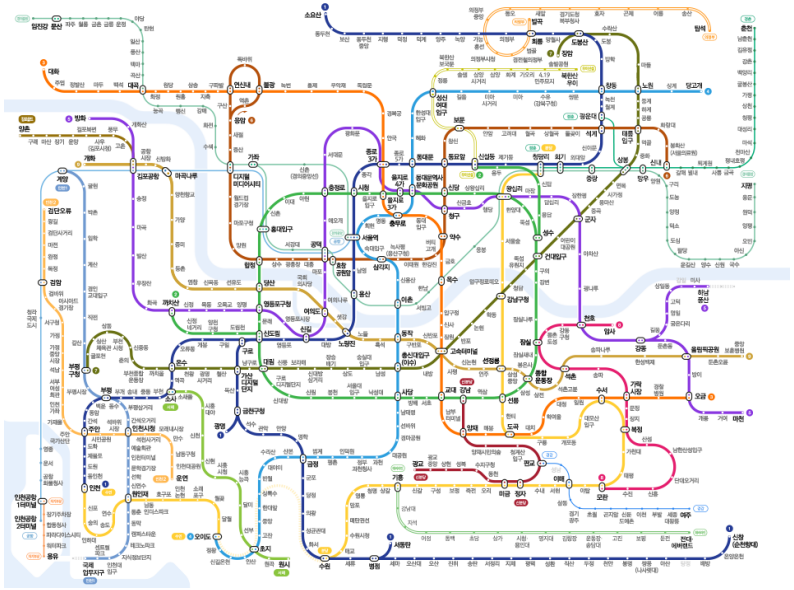
HSU 한성대학교 HANSUNG UNIVERSITY

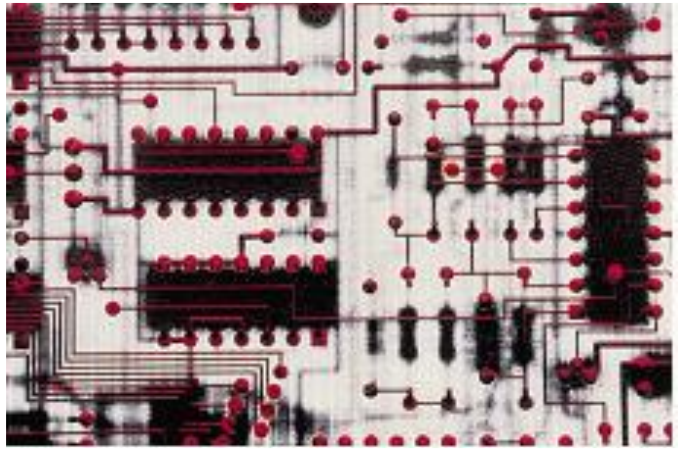
CryptoCraft LAB
https://crypto.modoo.at

그래프

- 그래프(graph) : 연결되어 있는 객체간의 관계를 표현하는 자료구조
  - 아주 일반적인 자료구조

- 그래프의 예 : 전기회로, 지도, 지하철 노선도 등등!





그래프 용어

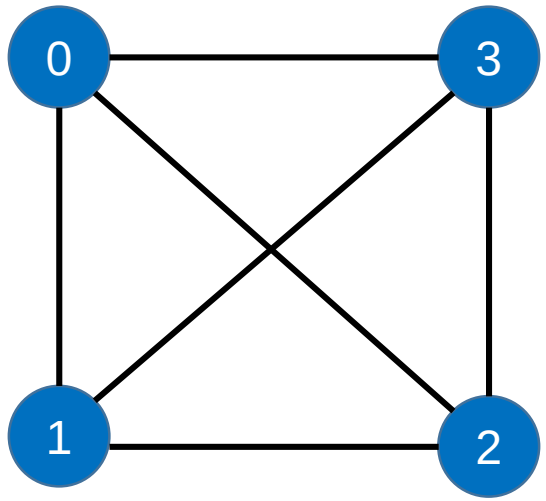
- 그래프는 (V, E)로 표시된다.
  - V는 정점(vertices)들의 집합
  - E는 간선(edge)들의 집합

- 예제 그래프
  - 정점은 각 지하철역을 의미한다.
  - 간선은 지하철역을 이어주는 선로를 의미한다.
  - 간선에는 다음역까지 걸리는 시간과 같은 데이터가 저장 될 수 있다.

그래프의 용어

- 인접 정점 : 간선에 의해 연결된 정점
- 차수 : 정점에 연결된 다른 정점의 개수
- 경로 : 정점의 나열로 표현
  - EX) 단순경로 : 0, 1, 2, 3

- 경로의 길이 : 경로를 구성하는데 사용된 간선의 수
- 완전그래프 : 모든 정점이 연결되어 있는 그래프



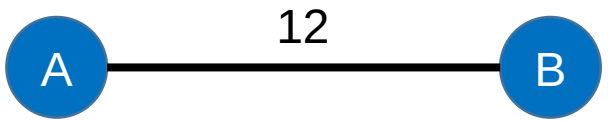
완전그래프

그래프의 종류

- 간선의 종류에 따라 그래프를 무방향 그래프와 방향 그래프로 구분
  - 무방향 그래프 : 간선을 통해서 양방향으로 갈수 있음을 나타낸다. (A, B)와 같이 정점의 쌍으로 표현
  - (A,B) = (B,A)

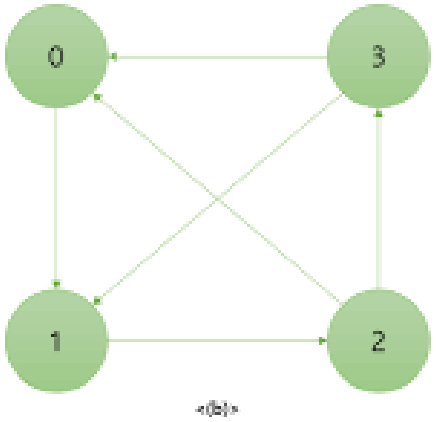
  - 방향 그래프 : 방향성이 존재하는 간선으로 도로의 일방통행길과 마찬가지로 한쪽 방향으로만 갈 수 있음을 나타낸다. <A,B>로 표현
  - <A,B> ≠ <B,A>

- 가중치 그래프, 네트워크 : 간선에 비용이나 가중치가 할당된 그래프



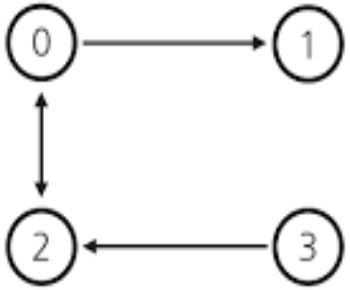
그래프 표현 방법

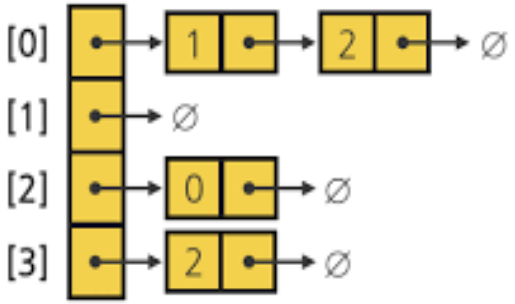
- 그래프를 표현하는 2가지 방법
  - 인접행렬 : 2차원 배열을 사용하여 표현
  - 인접리스트 : 연결리스트를 사용하여 표현



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

인접행렬

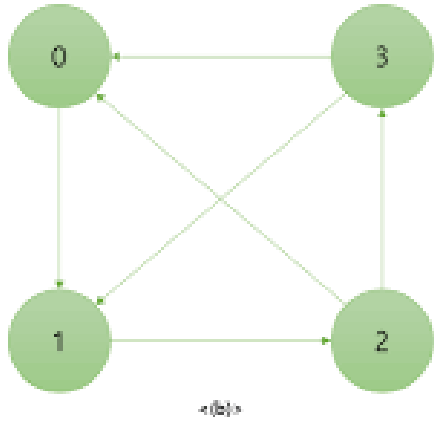




인접리스트

그래프의 표현 방법 - 인접행렬

- 인접행렬 방법
  - 간선 (I,j)가 그래프에 존재하면 M[i][j] = 1
  - 간선 (I,j)가 그래프에 존재하지 않으면 M[i][j] = 0



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

그래프의 표현 방법 - 인접행렬

```c
#define MAX_VERTICES 50
typedef struct GraphType {
    int n;
    int adj_mat[MAX_VERTICES][MAX_VERTICES];
} GraphType;
```

```c
//그래프 초기화
void init(GraphType* g){
    int r, c;
    g->n = 0;
    for(r=0; r<MAX_VERTICES; r++){
        for(c=0; c<MAX_VERTICES; c++){
            g->adj_mat[r][c] = 0;
        }
    }
}
```

```c
//정점 삽입 연산
void insert_vertex(GraphType* g, int v){
    if(((g->n)+1) > MAX_VERTICES) {
        printf("그래프 정점의 개수 초과\n");
    } else {
        g->n++;
        printf("%d 정점 추가 성공\n", v);
    }
}
```

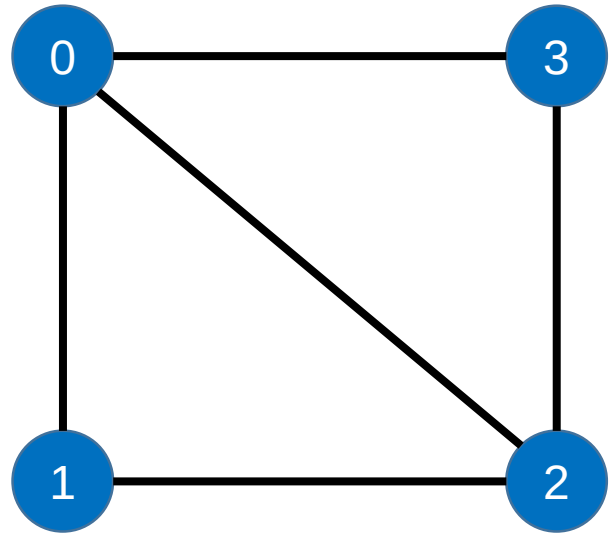
그래프의 표현 방법 - 인접행렬

```c
//간선 삽입 연산
void insert_edge(GraphType* g, int start, int end){
    if(start>= g->n || end>= g->n){
        printf("그래프 정점 번호 오류\n");
    } else {
        g->adj_mat[start][end] = 1;
        g->adj_mat[end][start] = 1;
        printf("(%d, %d) 간선 추가 완료\n", start, end);
    }
}

//행렬 출력 함수
void print_adj_mat(GraphType* g){
    for(int i=0; i<g->n; i++){
        for(int j=0; j<g->n; j++){
            printf("%2d ", g->adj_mat[i][j]);
        }
        printf("\n");
    }
}
```

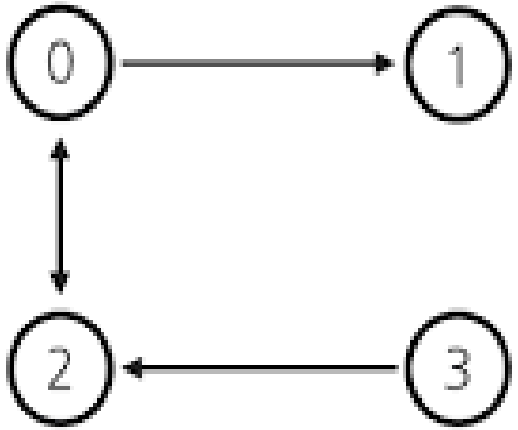
그래프의 표현 방법 - 인접행렬

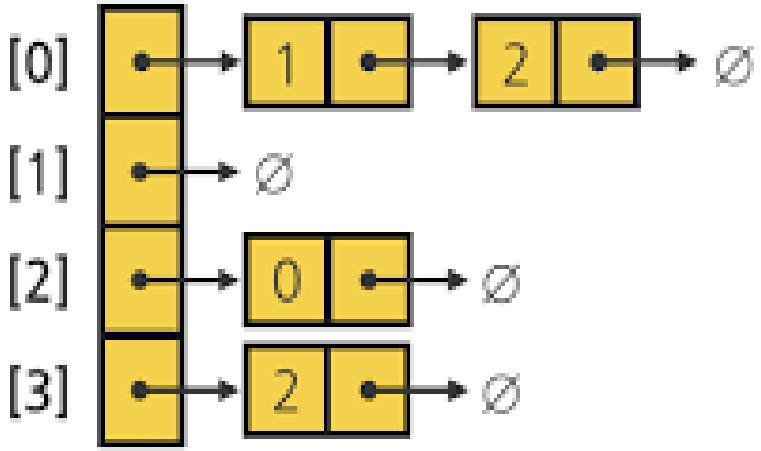
```
int main()
{
    GraphType *g;
    g = (GraphType*)malloc(sizeof(GraphType));
    init(g);
    for(int i=0; i<4; i++){
        insert_vertex(g, i);
    }

    insert_edge(g, 0, 1);
    insert_edge(g, 0, 2);
    insert_edge(g, 0, 3);
    insert_edge(g, 1, 2);
    insert_edge(g, 2, 3);

    printf("\n");
    print_adj_mat(g);

    free(g);
    return 0;
}
```

```
0 정점 추가 성공
1 정점 추가 성공
2 정점 추가 성공
3 정점 추가 성공
(0, 1) 간선 추가 완료
(0, 2) 간선 추가 완료
(0, 3) 간선 추가 완료
(1, 2) 간선 추가 완료
(2, 3) 간선 추가 완료

 0  1  1  1
 1  0  1  0
 1  1  0  1
 1  0  1  0
```



그래프의 표현 방법 – 인접리스트





그래프의 표현 방법 – 인접리스트

```c
typedef struct GraphNode {
    int vertex;
    struct GraphNode* link;
} GraphNode;

typedef struct GraphType {
    int n;
    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;
    }
}
```

```c
//정점 삽입 연산
void insert_vertex(GraphType* g, int v){
    if(((g->n)+1) > MAX_VERTICES) {
        printf("그래프 정점의 개수 초과\n");
    } else {
        g->n++;
        printf("%d 정점 추가 성공\n", v);
    }
}
```

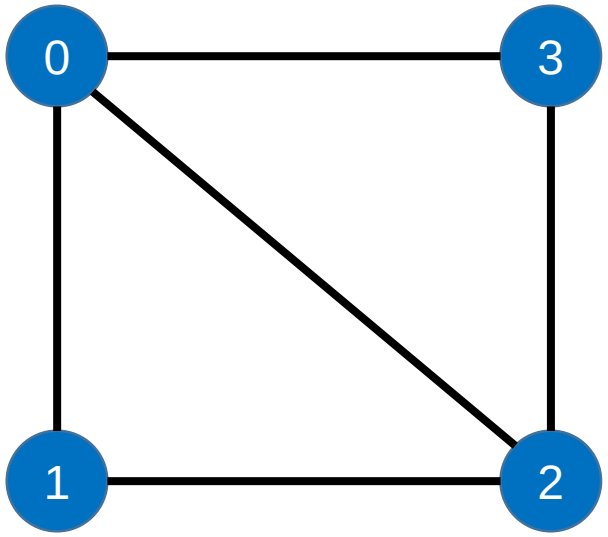
```c
//간선 삽입 연산
void insert_edge(GraphType* g, int u, int v){
    GraphNode* node;
    if(u>= g->n || v>= g->n){
        printf("그래프 정점 번호 오류\n");
    } else {
        node = (GraphNode*)malloc(sizeof(GraphNode));
        node->vertex = v;
        node->link = g->adj_list[u];
        g->adj_list[u] = node;
    }
}
```

그래프의 표현 방법 – 인접리스트

```c
int main()
{
    GraphType *g;
    g = (GraphType*)malloc(sizeof(GraphType));
    init(g);
    for(int i=0; i<4; i++){
        insert_vertex(g, i);
    }

    insert_edge(g, 0, 1);
    insert_edge(g, 1, 0);
    insert_edge(g, 0, 2);
    insert_edge(g, 2, 0);
    insert_edge(g, 0, 3);
    insert_edge(g, 3, 0);
    insert_edge(g, 1, 2);
    insert_edge(g, 2, 1);
    insert_edge(g, 2, 3);
    insert_edge(g, 3, 2);

    printf("\n");
    print_adj_mat(g);

    free(g);
    return 0;
}
```

```c
//행렬 출력 함수
void print_adj_mat(GraphType* g){
    for(int i=0; i<g->n; i++){
        GraphNode* p = g->adj_list[i];
        printf("정점 %d의 인접 리스트 ", i);
        while(p!=NULL){
            printf("->%d ", p->vertex);
            p = p->link;
        }
        printf("\n");
    }
}
```

```
0 정점 추가 성공
1 정점 추가 성공
2 정점 추가 성공
3 정점 추가 성공

정점 0의 인접 리스트 ->3 ->2 ->1
정점 1의 인접 리스트 ->2 ->0
정점 2의 인접 리스트 ->3 ->1 ->0
정점 3의 인접 리스트 ->2 ->0
```



Q & A