CSE2010 자료구조론

Week 9: Graph 2

ICT융합학부 한진영

그래프 ADT

·객체: 정점의 집합과 간선의 집합

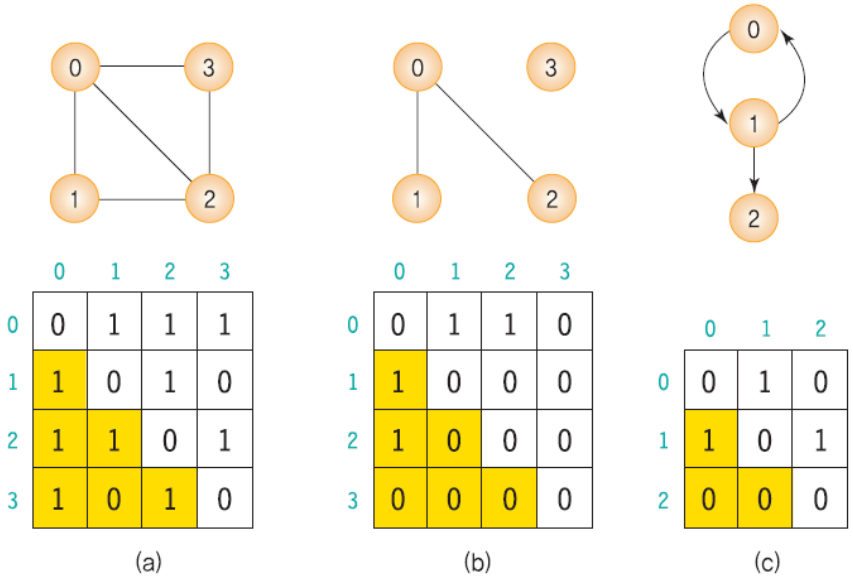
·연산:

- create_graph() ::= 그래프를 생성한다.
- init(g) ::= 그래프 g를 초기화한다.
- insert_vertex(g,v) ::= 그래프 g에 정점 v를 삽입한다.
- insert_edge(g,u,v) ::= 그래프 g에 간선 (u,v)를 삽입한다.
- delete_vertex(g,v) ::= 그래프 g의 정점 v를 삭제한다.
- delete_edge(g,u,v) ::= 그래프 g의 간선 (u,v)를 삭제한다.
- is_empty(g) ::= 그래프 g가 공백 상태인지 확인한다.
- adjacent(v) ::= 정점 v에 인접한 정점들의 리스트를 반환한다.
- destroy_graph(g) ::= 그래프 g를 제거한다.

* 그래프에 정점을 추가하려면 insert_vertex 연산 사용
* 그래프에 간선을 추가하려면 insert_edge 연산 사용

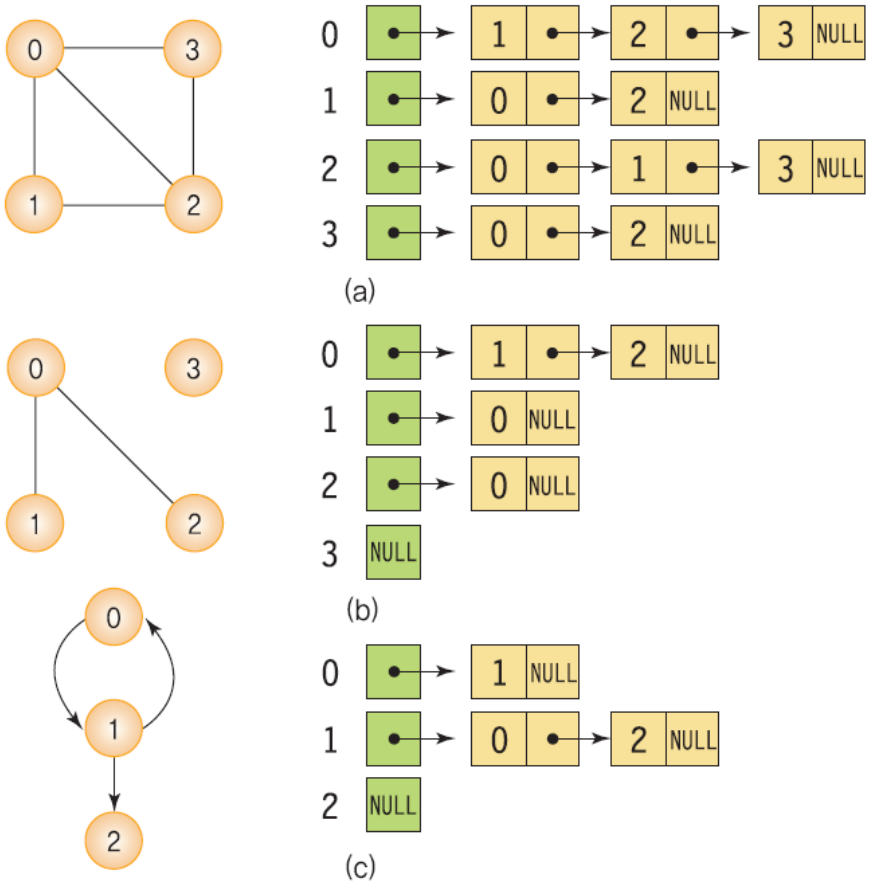
그래프 표현 방법(1)

- 인접행렬(adjacent matrix) 방법
  - if(간선 (i, j)가 그래프에 존재) M[i][j] = 1
  - 그렇지않으면 M[i][j] = 0
- 인접 행렬의 대각선 성분은 모두 0(자체 간선 불허)
- 무방향 그래프의 인접 행렬은 대칭



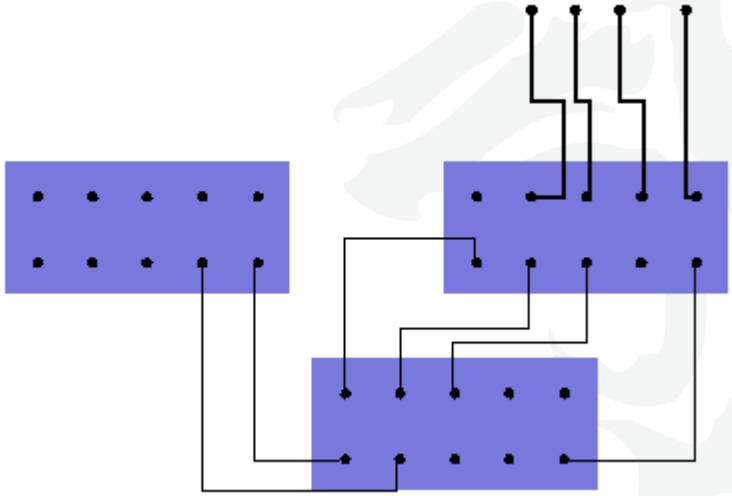
그래프 표현 방법(2)

- 인접리스트(adjacency list) 방법
  - 각 정점에 인접한 정점들을 연결리스트로 표현



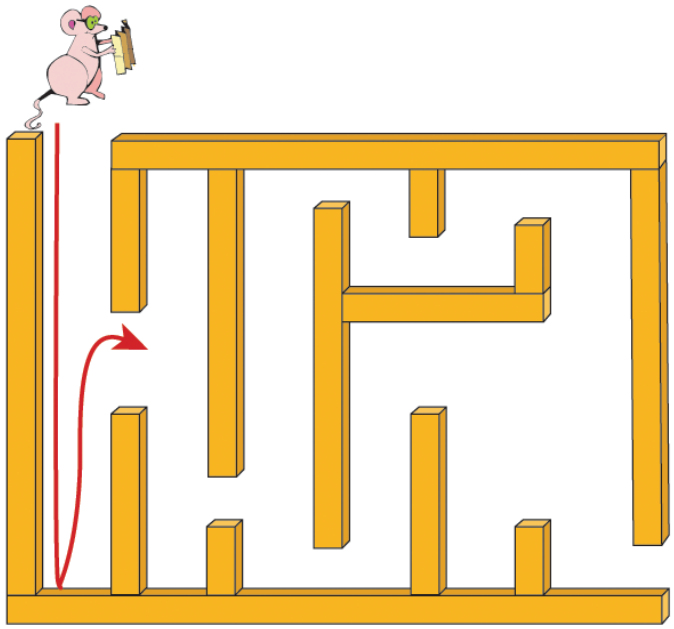
그래프 탐색

- 그래프의 가장 기본적인 연산
  - 하나의 정점으로부터 시작하여 차례대로 모든 정점들을 한번씩 방문
  - 많은 문제들이 단순히 그래프의 노드를 탐색하는 것으로 해결
    - (예) 도로망에서 특정 도시에서 다른 도시로 갈 수 있는지 여부
    - (예) 전자회로에서 특정 단자와 다른 단자가 서로 연결되어 있는지 여부



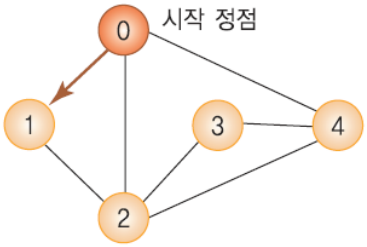
깊이 우선 탐색(DFS)

- 깊이 우선 탐색(DFS: depth-first search)
  - 한 방향으로 갈 수 있을 때까지 가다가 더 이상 갈 수 없게 되면 가장 가까운 갈림길로 돌아와서 이 곳으로부터 다른 방향으로 다시 탐색 진행
  - 되돌아가기 위해서는 스택 필요

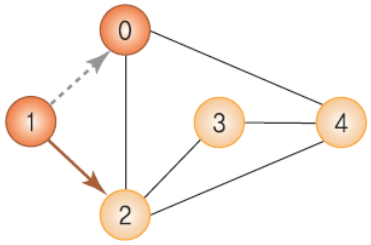


DFS 알고리즘

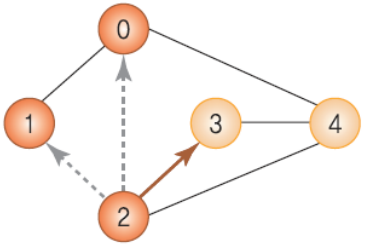
```
depth_first_search(v)
        v를 방문되었다고 표시;
        for all u ∈ (v에 인접한 정점) do
            if (u가 아직 방문되지 않았으면) then depth_first_search(u)
```



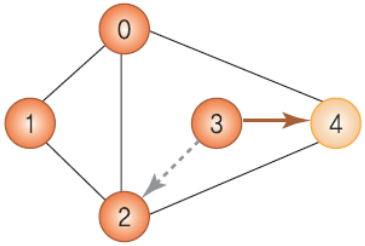
(a) 정점 1 방문



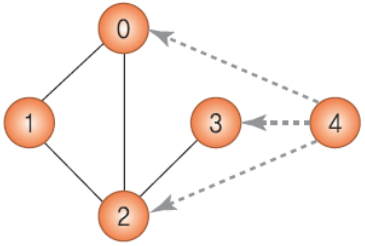
(b) 정점 2 방문



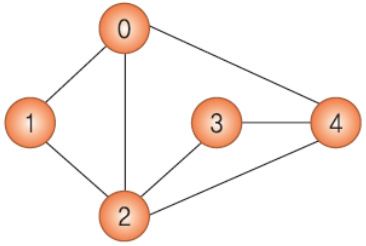
(c) 정점 3 방문



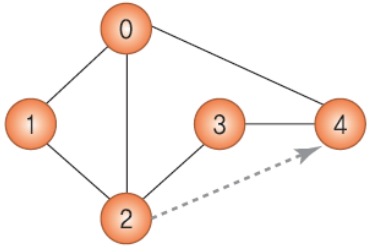
(d) 정점 4 방문



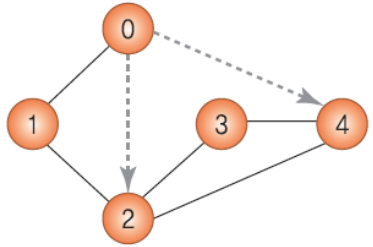
(e) 정점 3으로 backtracking



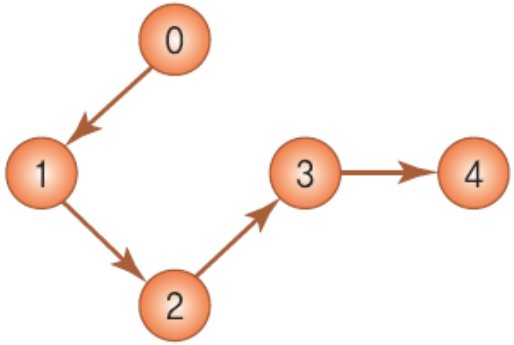
(f) 정점 2로 backtracking



(g) 정점 1로 backtracking



(h) 정점 0으로 backtracking(탐색 종료)



(i) 탐색 결과(방문 순서 0, 1, 2, 3, 4)

DFS 구현(1)

```
// 인접 행렬로 표현된 그래프에 대한 깊이 우선 탐색
void dfs_mat(GraphType *g, int v)
{
    int w;
    visited[v] = TRUE;          // 정점 v의 방문 표시
    printf("%d ", v);       // 방문한 정점 출력
    for(w=0; w<g->n; w++)              // 인접 정점 탐색
        if( g->adj_mat[v][w] && !visited[w] ) dfs_mat(g, w);  //정점 w에서 DFS 새로시작
}
```

DFS 구현(2)

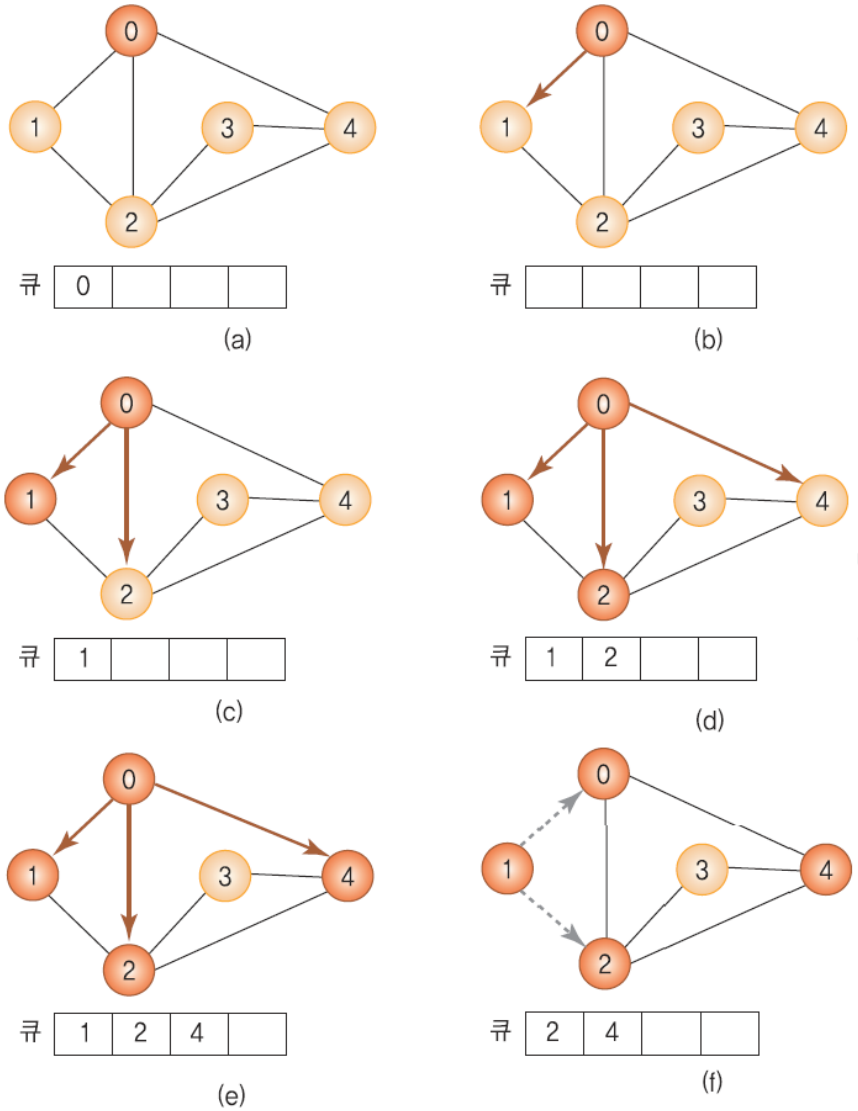
```
// 인접 리스트로 표현된 그래프에 대한 깊이 우선 탐색
void dfs_list(GraphType *g, int v)
{
   GraphNode *w;
   visited[v] = TRUE;              // 정점 v의 방문 표시
   printf("%d ", v);              // 방문한 정점 출력
   for(w=g->adj_list[v]; w; w=w->link)   // 인접 정점 탐색
     if(!visited[w->vertex]) dfs_list(g, w->vertex); //정점 w->vertex에서 DFS 새로시작
}
```

너비 우선 탐색(BFS)

- 너비 우선 탐색(BFS: breadth-first search)
  - 시작 정점으로부터 가까운 정점을 먼저 방문하고 멀리 떨어져 있는 정점을 나중에 방문하는 순회 방법
  - 큐를 사용하여 구현됨
- 너비우선탐색 알고리즘

```
breadth_first_search(v)
v를 방문되었다고 표시;
큐 Q에 정점 v를 삽입;
while (not is_empty(Q)) do
    큐 Q에서 정점 w를 삭제;
    for all u ∈ (w에 인접한 정점) do
        if (u가 아직 방문되지 않았으면) then    u를 큐 Q에 삽입;
                                                 u를 방문되었다고 표시;
```

BFS 예


0
1
3
4
2
큐 0
(a)
큐
(b)
큐 1
(c)
큐 1 2
(d)
큐 1 2 4
(e)
큐 2 4
(f)
큐 4 3
(g)
큐 3
(h)
큐
(i)
큐
(j) 탐색 결과(방문 순서 0, 1, 2, 4, 3)

BFS 구현(1) – 인접행렬

```
void bfs_mat(GraphType *g, int v)
{       int w;
        QueueType q;
        init(&q);                   // 큐 초기화
        visited[v] = TRUE;    // 정점 v 방문 표시
        printf("%d ", v);           // 정점 출력
        enqueue(&q, v);             // 시작 정점을 큐에 저장
        while(!is_empty(&q)){
        v = dequeue(&q);            // 큐에 정점 추출
        for(w=0; w<g->n; w++) // 인접 정점 탐색
                if(g->adj_mat[v][w] && !visited[w]){
                        visited[w] = TRUE;    // 방문 표시
                        printf("%d ", w);           // 정점 출력
                        enqueue(&q, w);         // 방문한 정점을 큐에 저장
                }
        }
}
```

BFS 구현(2) – 인접리스트

```
void bfs_list(GraphType *g, int v)
{       GraphNode *w;
        QueueType q;
        init(&q);          // 큐 초기화
        visited[v] = TRUE;   // 정점 v 방문 표시
        printf("%d ", v);       // 정점 v 출력
        enqueue(&q, v);             // 시작정점을 큐에 저장
        while(!is_empty(&q)){
                v = dequeue(&q);                    // 큐에서 정점 추출
                for(w=g->adj_list[v]; w; w = w->link) //인접 정점 탐색
                        if(!visited[w->vertex]){        // 미방문 정점 탐색
                        visited[w->vertex] = TRUE;   // 방문 표시
                        printf("%d ", w->vertex);     // 정점 출력
                        enqueue(&q, w->vertex); // 방문한 정점을 큐에 삽입
                }
        }
}
```

Week 9: Graph 2