컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 33강 - 그래프의 개념과 구현

Fast campus

학습 목표

그래프의 개념과 구현

1) 그래프의 개념에 대해서 이해합니다.
2) 그래프를 C언어를 이용하여 구현할 수 있습니다.

그래프의 개념과 구현

그래프의 개념과 구현

그래프(Graph)란 사물을 정점(Vertex)와 간선(Edge)으로 나타내기 위한 도구입니다.

그래프는 두 가지 방식으로 구현할 수 있습니다.

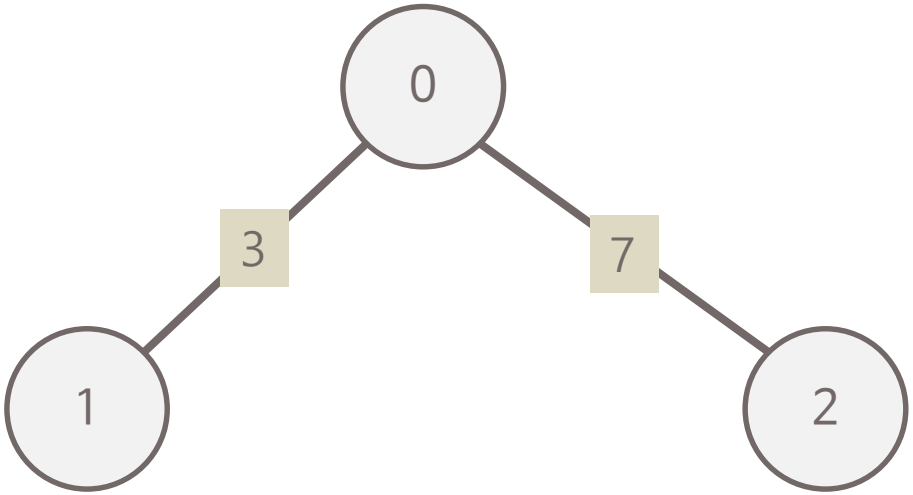
1) 인접 행렬(Adjacency Matrix): 2차원 배열을 사용하는 방식
2) 인접 리스트(Adjacency List): 리스트를 사용하는 방식

Fast campus

그래프의 개념과 구현

그래프의 개념과 구현

인접 행렬(Adjacency Matrix)에서는 그래프를 2차원 배열로 표현합니다.



0	3	7
3	0	무한
7	무한	0

그래프의 개념과 구현

무방향 비가중치 그래프와 인접 행렬

- 모든 간선이 방향성을 가지지 않는 그래프를 무방향 그래프라고 합니다.
- 모든 간선에 가중치가 없는 그래프를 비가중치 그래프라고 합니다.
- 무방향 비가중치 그래프가 주어졌을 때 연결되어 있는 상황을 인접 행렬로 출력할 수 있습니다.

그래프의 개념과 구현

무방향 비가중치 그래프와 인접 행렬

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int a[1001][1001];
int n, m;

int main(void) {
  scanf("%d %d", &n, &m);
  for (int i = 0; i < m; i++) {
    int x, y;
    scanf("%d %d", &x, &y);
    a[x][y] = 1;
    a[y][x] = 1;
  }
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
      printf("%d ", a[i][j]);
    }
    printf("\n");
  }
  system("pause");
}
```

그래프의 개념과 구현

방향 가중치 그래프와 인접 리스트

- 모든 간선이 방향을 가지는 그래프를 방향 그래프라고 합니다.
- 모든 간선에 가중치가 있는 그래프를 가중치 그래프라고 합니다.
- 무방향 비가중치 그래프가 주어졌을 때 연결되어 있는 상황을 인접 리스트로 출력할 수 있습니다.

그래프의 개념과 구현

방향 가중치 그래프와 인접 리스트 1) 연결 리스트 구조체 만들기

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

typedef struct {
  int index;
  int distance;
  struct Node *next;
} Node;
```

Fast campus

그래프의 개념과 구현

방향 가중치 그래프와 인접 리스트 2) 연결 리스트 삽입 함수

```
void addFront(Node *root, int index, int distance) {
  Node *node = (Node*)malloc(sizeof(Node));
  node->index = index;
  node->distance = distance;
  node->next = root->next;
  root->next = node;
}
```

Fast campus

그래프의 개념과 구현

방향 가중치 그래프와 인접 리스트 3) 연결 리스트 출력 함수

```
void showAll(Node *root) {
  Node *cur = root->next;
  while (cur != NULL) {
    printf("%d(거리: %d) ", cur->index, cur->distance);
    cur = cur->next;
  }
}
```

그래프의 개념과 구현

방향 가중치 그래프와 인접 리스트 4) 연결 리스트 사용해보기

```
int main(void) {
  int n, m;
  scanf("%d %d", &n, &m);
  Node** a = (Node**)malloc(sizeof(Node*) * (n + 1));
  for (int i = 1; i <= n; i++) {
    a[i] = (Node*)malloc(sizeof(Node));
    a[i]->next = NULL;
  }
  for (int i = 0; i < m; i++) {
    int x, y, distance;
    scanf("%d %d %d", &x, &y, &distance);
    addFront(a[x], y, distance);
  }
  for (int i = 1; i <= n; i++) {
    printf("원소 [%d]: ", i);
    showAll(a[i]);
    printf("\n");
  }
  system("pause");
  return 0;
}
```

배운 내용 정리하기

그래프의 개념과 구현

1) 인접 행렬은 모든 정점들의 연결 여부를 저장하여 $O(V^2)$의 공간을 요구하므로 공간 효율성이 떨어지지만 두 정점이 서로 연결되어 있는지 확인하기 위해 $O(1)$의 시간을 요구합니다.

2) 인접 리스트는 연결된 간선의 정보만을 저장하여 $O(E)$의 공간을 요구하므로 공간 효율성이 우수하지만 두 정점이 서로 연결되어 있는지 확인하기 위해 $O(V)$의 시간을 요구합니다.