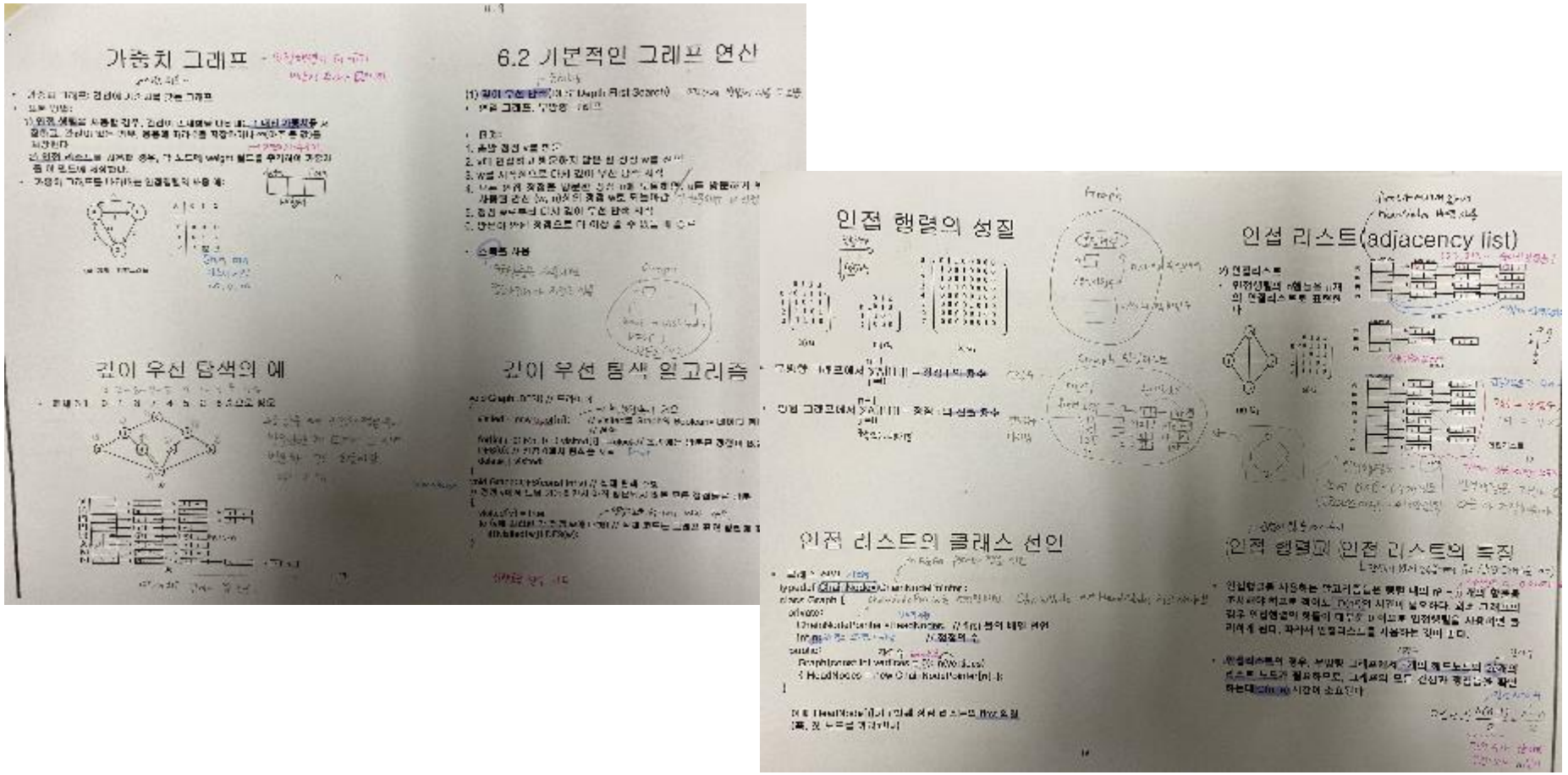
자료구조를 구조하자

2019.11.08

32153180 이상민

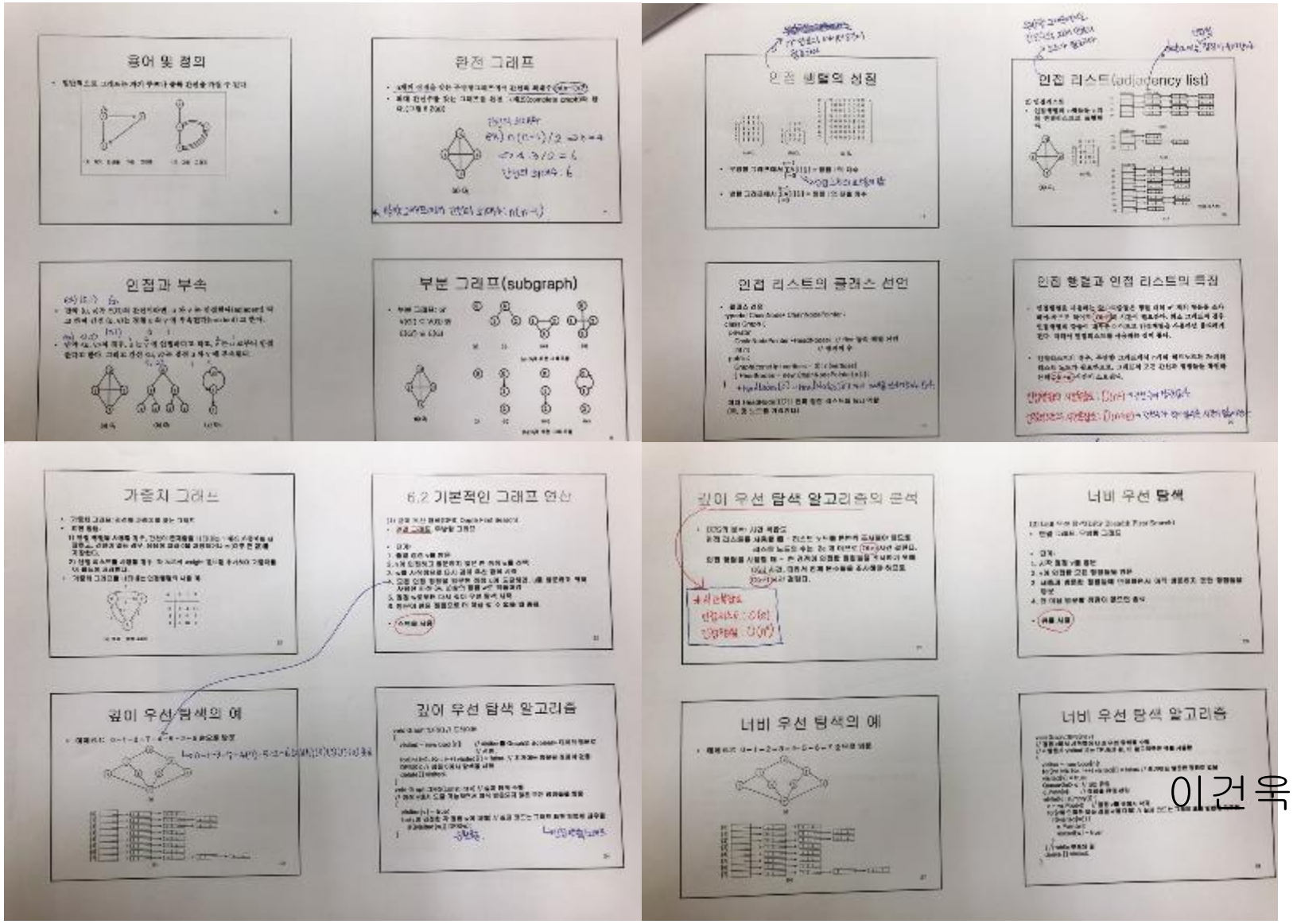
32162436 신창우 32163006 이건욱
32164420 조정민 32164959 허전진

학습 내용


가중치 그래프
6.2 기본적인 그레프 연산
깊이 우선 탐색의 예
깊이 우선 탐색 알고리즘
인접 행렬의 성질
인접 리스트의 클래스 선언
인접 행렬과 인접 리스트의 특징

신창우

학습 내용


용어 및 정의
완전 그래프
인접과 부속
부분 그래프(subgraph)
인접 행렬의 선정
인접 리스트(adjacency list)
인접 리스트의 클래스 선언
인접 행렬과 인접 리스트의 특징
가중치 그래프
6.2 기본적인 그래프 연산
깊이 우선 탐색 알고리즘의 분석
너비 우선 탐색
깊이 우선 탐색의 예
깊이 우선 탐색 알고리즘
너비 우선 탐색의 예
너비 우선 탐색 알고리즘

이건욱

학습 내용

```
bool Bst::Insert(const int &x)
{
    BstNode *p = root;
    BstNode *q = 0;

    while (p)
    {
        q = p;
        if (x == p->data)
        {
            cout << x << "가 이미 존재합니다" <<
            return false;
        }
        if (x < p->data)
            p = p->LeftChild;
        else
            p = p->RightChild;
    }
    p = new BstNode;
    p->LeftChild = p->RightChild = 0;
    p->data = x;

    if (!root)
        root = p;
    else if (x < q->data)
        q->LeftChild = p;
    else
        q->RightChild = p;
    return true;
}
```

```
bool Bst::Insert(const int &x)                  // 삽입 함수
{
    BstNode *p = root;
    BstNode *q = 0;                             // p를 뒤따라오는 노드

    while (p)
    {
        q = p;
        if (x == p->data)                       // 삽입하려는 x가 이미 존재할 경우
        {
            cout << x << "가 이미 존재합니다" << endl;
            return false;
        }
        if (x < p->data)
            p = p->LeftChild;
        else
            p = p->RightChild;
    }
    p = new BstNode;
    p->LeftChild = p->RightChild = 0;           // p의 왼쪽 자식, 오른쪽 자식 NULL
    p->data = x;

    if (!root)                                  // 빈 리스트
        root = p;
    else if (x < q->data)
        q->LeftChild = p;
    else
        q->RightChild = p;
    return true;
}
```

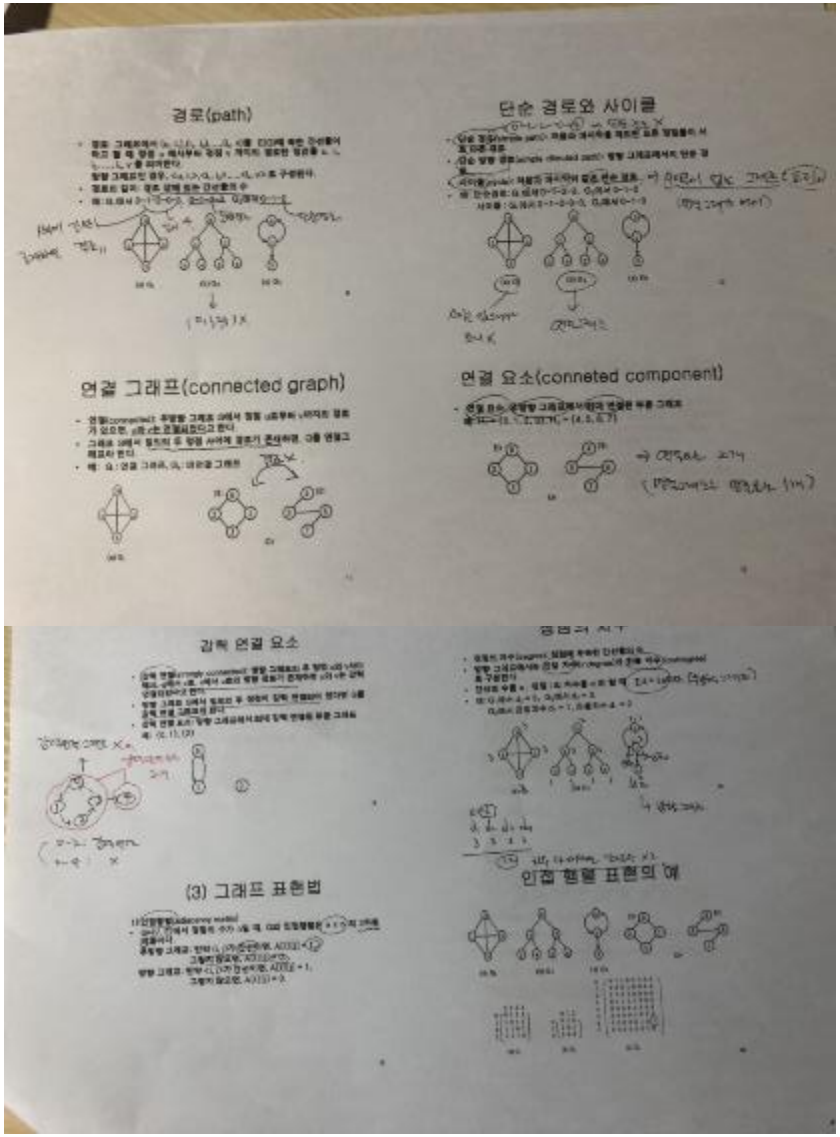
```
bool Bst::Delete(const int &x)
{
    BstNode *p = root;
    BstNode *q = 0;

    while (p)
    {
        if (x < p->data)
        {
            q = p;
            p = p->LeftChild;
        }
        else if (x > p->data)
        {
            q = p;
            p = p->RightChild;
        }
        else
            break;
    }

    if (!p)
    {
        cout << "삭제할 노드가 없습니다." << endl
        return false;
    }

    if (p->LeftChild == 0 && p->RightChild == 0)
    {
        if (x < q->data)
        {
```

```
bool Bst::Delete(const int &x)                  // 삭제 함수
{
    BstNode *p = root;
    BstNode *q = 0;                             // p를 뒤따라오는 노드

    while (p)                                   // 빈 리스트가 아닌 경우
    {
        if (x < p->data)                        // x가 p의 data보다 작은 경우
        {
            q = p;
            p = p->LeftChild;                   // 왼쪽 자식으로 이동
        }
        else if (x > p->data)                   // x가 p의 data보다 큰 경우
        {
            q = p;
            p = p->RightChild;                  // 오른쪽 자식으로 이동
        }
        else                                  // x가 p의 data와 같은 경우
            break;
    }

    if (!p)                                     // 일치하는 값이 없는 경우
    {
        cout << "삭제할 노드가 없습니다." << endl;
        return false;
    }

    if (p->LeftChild == 0 && p->RightChild == 0)        // 삭제할 노드의 자식이 없는 경우(단말노드)
    {
        if (x < q->data)                        // x가 q의 data보다 작은 경우
        {
```

```
C:₩Windows₩system32₩cmd.exe
********** 이진탐색트리 **********
(1) 삽입 (2) 삭제 (3) 탐색 (4) 중위우선순회 (0) 종료
1
입력할 키 개수 : 10
1번째 : 5
2번째 : 80
3번째 : 25
4번째 : 67
5번째 : 16
6번째 : 24
7번째 : 1
8번째 : 99
9번째 : 86
10번째 : 45
4
중위우선순회 출력
1 5 16 24 25 45 67 80 86 99
3
탐색할 값 입력 : 45
성공!
3
탐색할 값 입력 : 68
실패!
2
삭제할 값 입력 : 24
2
삭제할 값 입력 : 99
4
중위우선순회 출력
1 5 16 25 45 67 80 86
0
계속하려면 아무 키나 누르십시오 . . .
```

조정민

학습 내용




```
else                    // 찾는 값의 노드가 존재할 경우.
{
    if (p->LeftChild == 0 && p->RightChild == 0)// 단말노드일 경우.
    {
        if (p == root)  // 루트노드일 경우.
            root = 0;   // 루트삭제.
        else if (p->key < q->key)   // p가 q의 왼쪽자식일 경우.
            q->LeftChild = 0;       // q의 왼쪽자식 삭제.
        else            // p가 q의 오른쪽자식일 경우.
        q->RightChild = 0;          // q의 오른쪽자식 삭제.
    }

    else if (p->LeftChild && p->RightChild == 0)
                        // 왼쪽 자식만을 갖는 비단말노드일 경우.
    {
        if (p == root)  // 루트노드일 경우.
            root = p->LeftChild;    // 루트에 p의 왼쪽자식 대입.
        else if (x < q->key)        // p가 q의 왼쪽자식일 경우.
            q->LeftChild = p->LeftChild;
                        // q의 왼쪽자식에 p의 왼쪽자식 연결.
        else            // p가 q의 오른쪽자식일 경우.
            q->RightChild = p->LeftChild;
                        // q의 오른쪽자식에 p의 왼쪽자식 연결.
    }
    else if (p->LeftChild == 0 && p->RightChild)
                        // 오른쪽 자식만을 갖는 비단말노드일 경우.
    {
        if (p == root)  // 루트노드일 경우.
            root = p->RightChild;   // 루트에 p의 오른쪽자식 대입.
        else if (x < q->key)        // p가 q의 왼쪽자식일 경우.
            q->LeftChild = p->RightChild;
                        // q의 왼쪽자식에 p의 오른쪽자식 연결.
        else            // p가 q의 오른쪽자식일 경우.
            q->RightChild = p->RightChild;
                        // q의 오른쪽자식에 p의 오른쪽자식 연결.
    }

    else                // 두 개의 자식을 갖는 노드일 경우.
    {
        r = p;          // r에 p를 저장해둠.
        q = p;          // q는 p의 선행 노드를 가리킴.
        p = p->RightChild;          // p는 p의 오른쪽자식으로 이동.

        while (p->LeftChild)
        // p는 가장 작은 값으로 이동하고 q는 p의 선행노드를 가리킴.
        {
            q = p;
            p = p->LeftChild;
        }

        r->key = p->key;// key값 저장.
```

허정진

참고자료

https://www.youtube.com/watch?v=fVcKN42YXXI

https://www.youtube.com/watch?v=_hxFgg7TLZQ