Data Structure

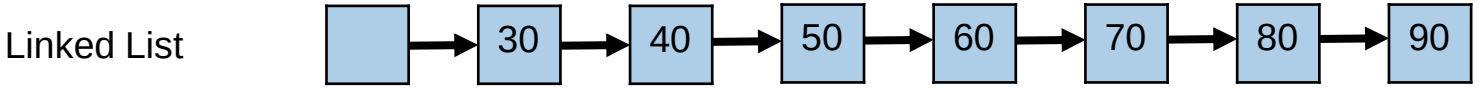
실습 6

0. 이번 주 실습 내용

- Skip List
  - Skip List 개념
  - Perfect Skip List
  - Randomized Skip List
  - Skip List 실습 (Randomized Skip List)
- Equivalence classes
  - Equivalence classes 개념
  - Equivalence classes 실습
- 과제: Reversing Linked List

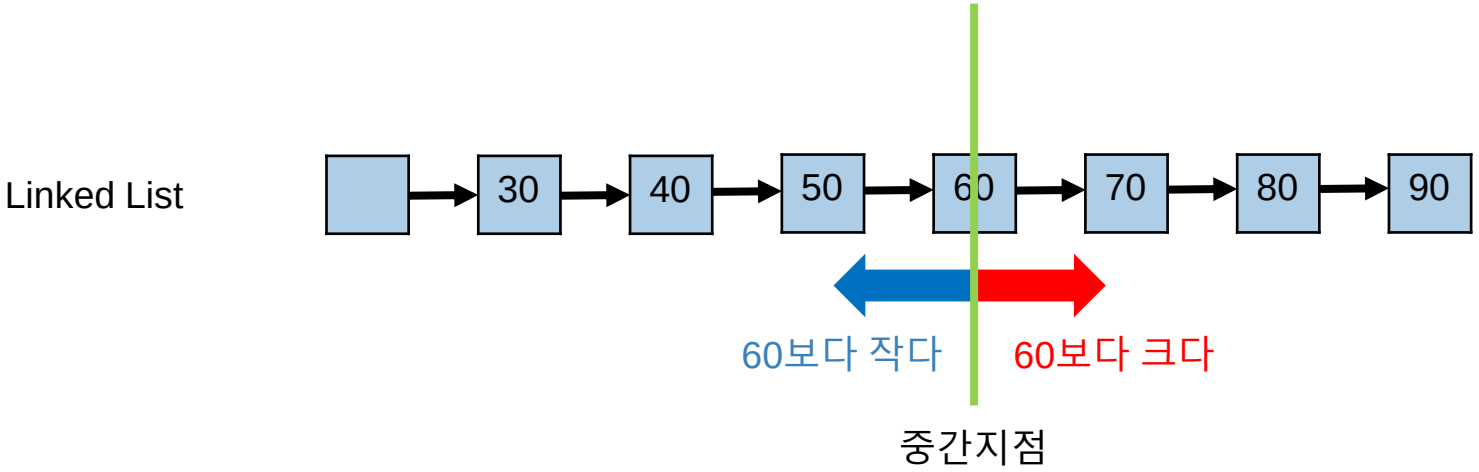
1. Skip List

- **어떻게 하면 리스트 안에서 자료를 빨리 찾을까?** Find(90)



1. Skip List

- **어떻게 하면 리스트 안에서 자료를 빨리 찾을까?** Find(90)



1. Skip List

- **어떻게 하면 리스트 안에서 자료를 빨리 찾을까?** Find(90)

1. Skip List

- 어떻게 하면 리스트 안에서 자료를 빨리 찾을까? Find(90)

1. Skip List

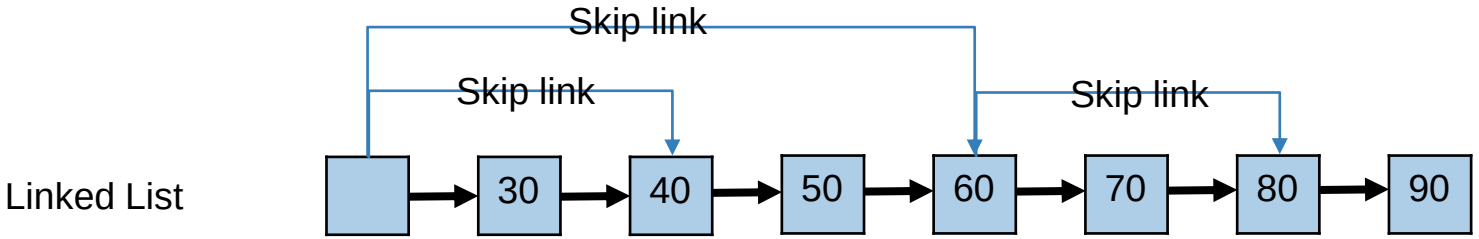
- **어떻게 하면 리스트 안에서 자료를 빨리 찾을까?** Find(90)

자료가 N개일 때 $\log_2 N$번의 비교로 검색가능

$$O(N) \quad \rightarrow \quad O(log\ N\ )$$

1. Skip List

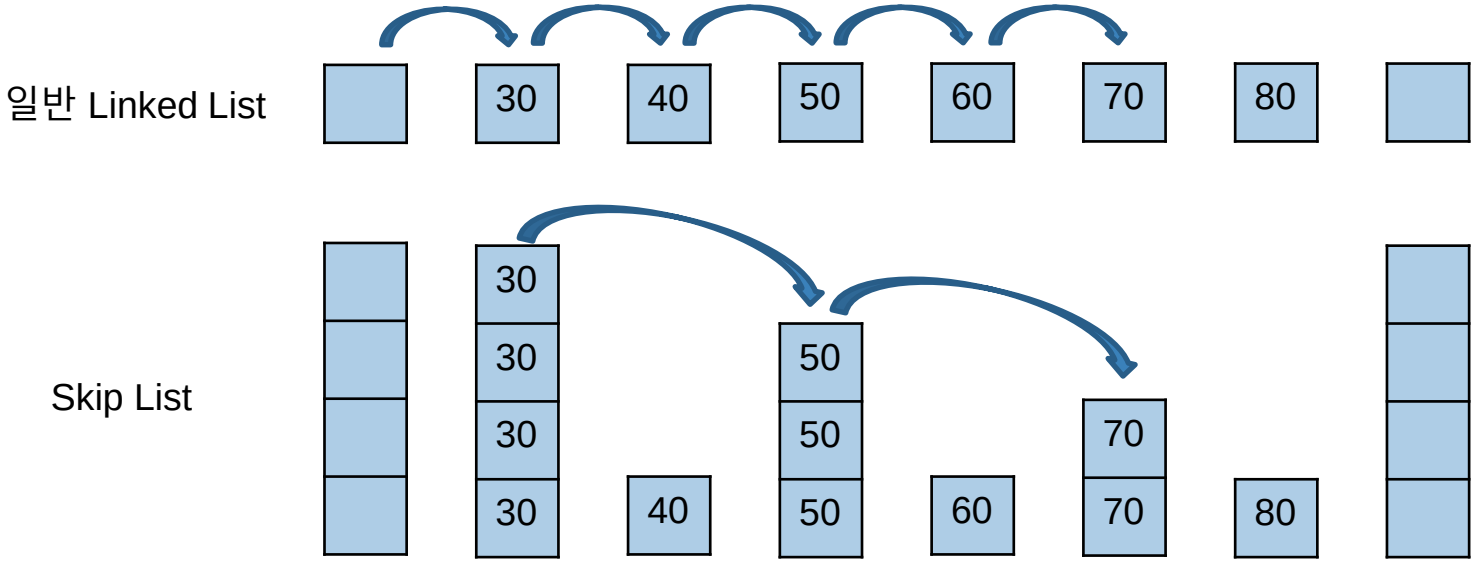
- **어떻게 하면 리스트 안에서 자료를 빨리 찾을까?** Find(90)



기본구현 아이디어: Skip link의 추가

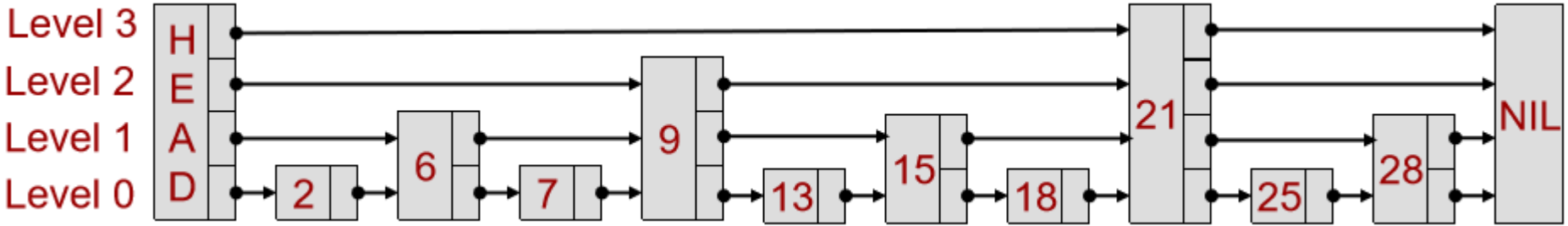
1. Skip List

- 정의: 빠른 탐색을 위해 Linked List을 변형한 자료구조



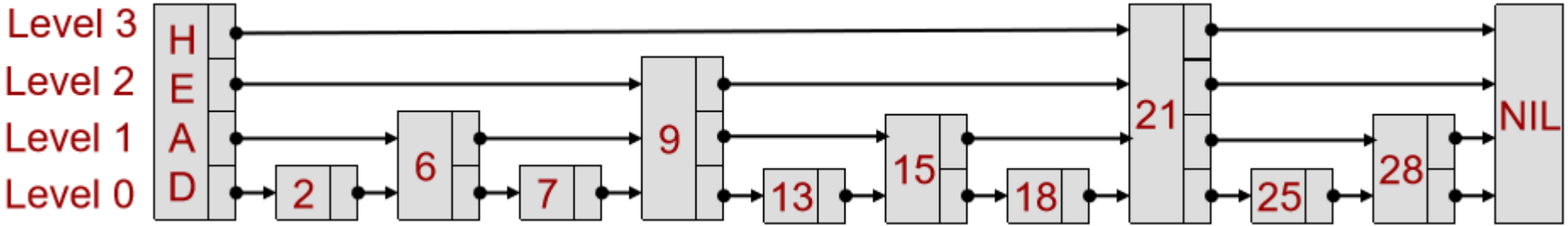
1. Perfect Skip List

- 현재 레벨의 원소 개수의 절반을 다음 레벨에 추가
  - 2번째 노드마다 레벨 1 추가
  - 4번째 노드마다 레벨 2 추가
  - 8번째 노드마다 레벨 3 추가



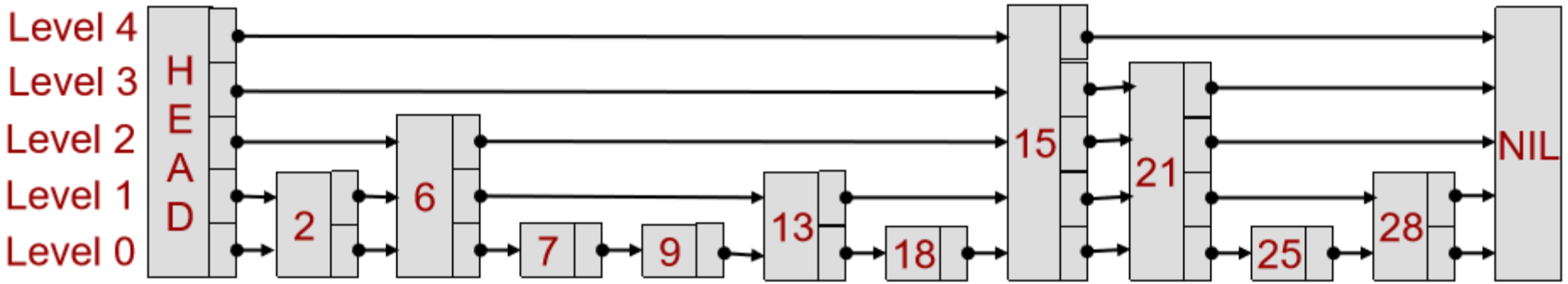
1. Perfect Skip List

- 규칙성을 유지하기 위해서 노드의 추가, 삭제가 불편

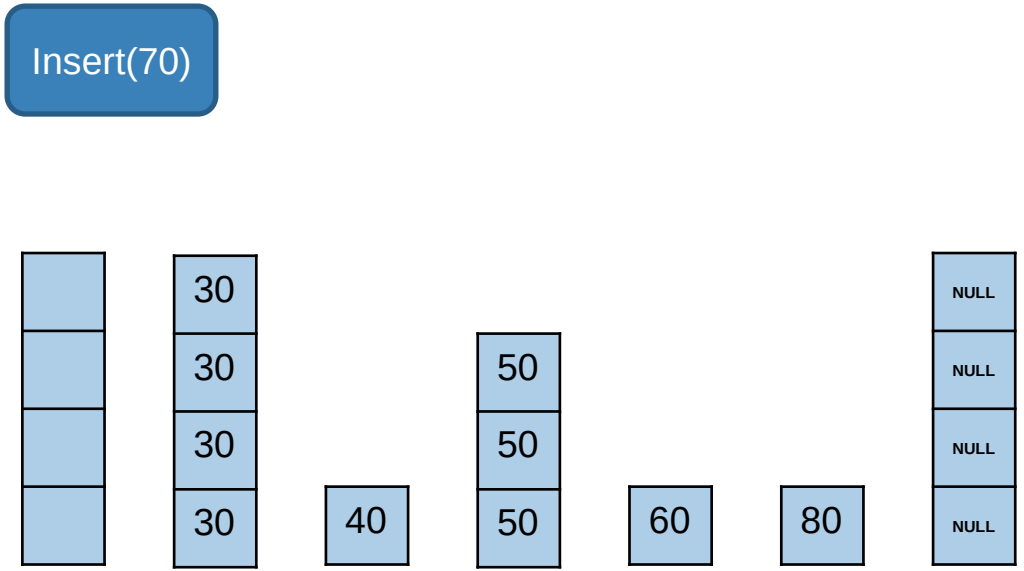


1. Randomized Skip List

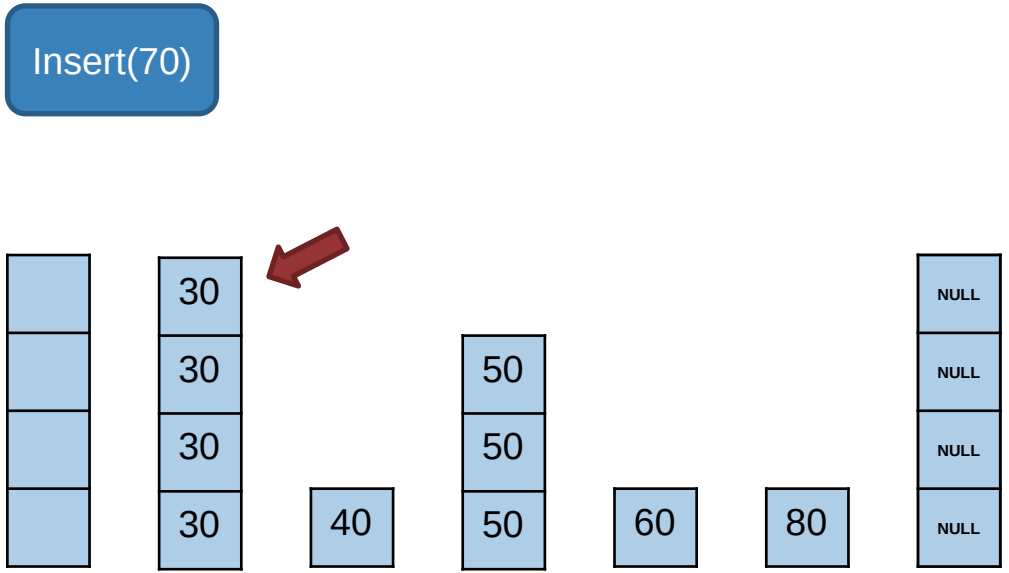
- 규칙없이 확률적으로 추가
- 레벨 별 원소의 개수에 집중



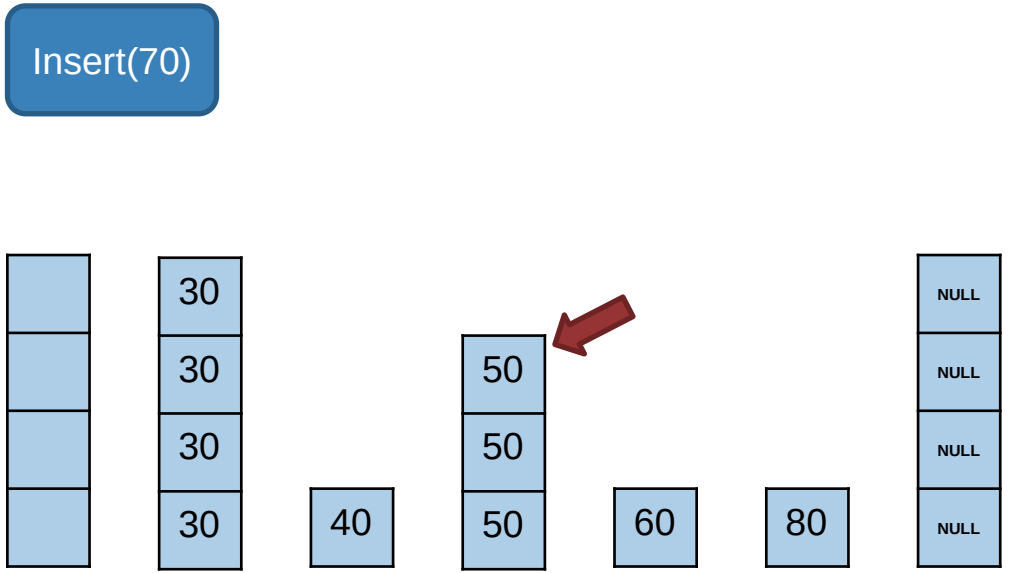
1. Randomized Skip List



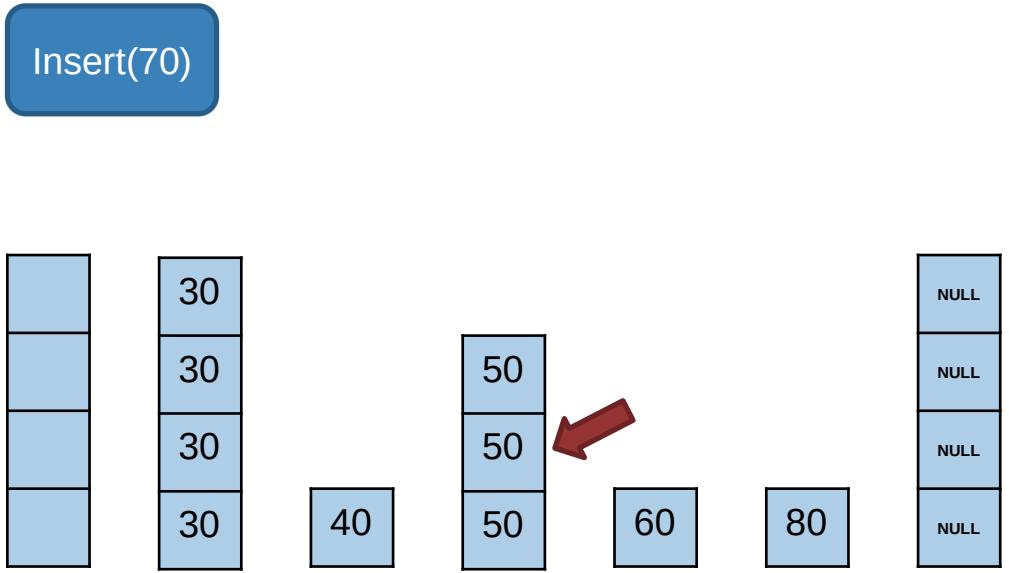
1. Randomized Skip List



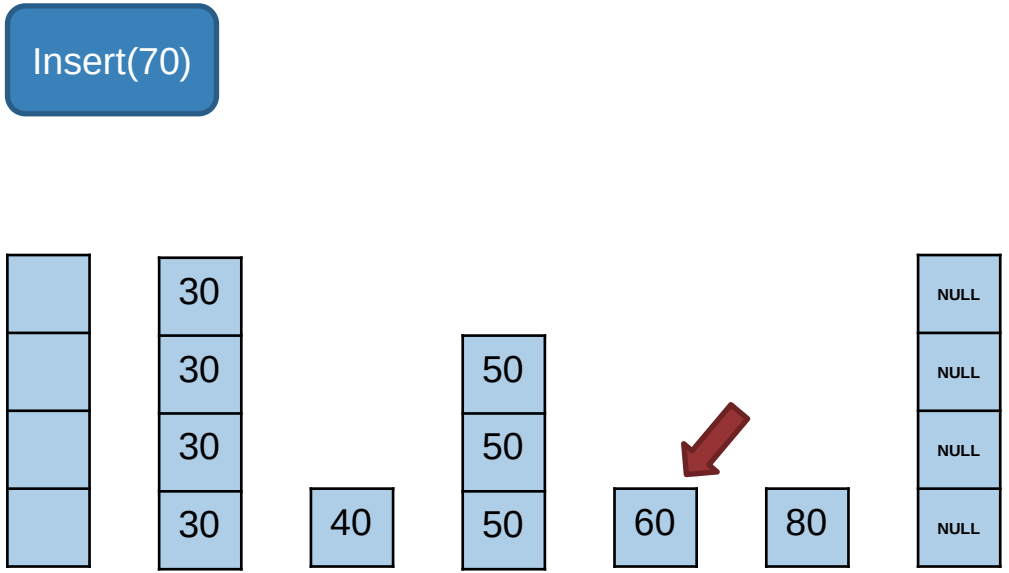
1. Randomized Skip List



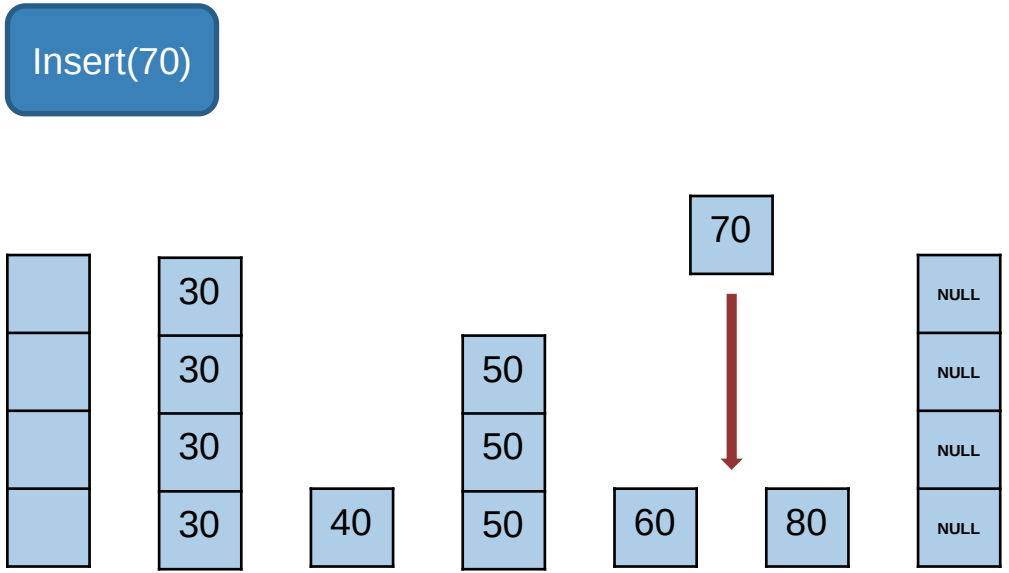
1. Randomized Skip List



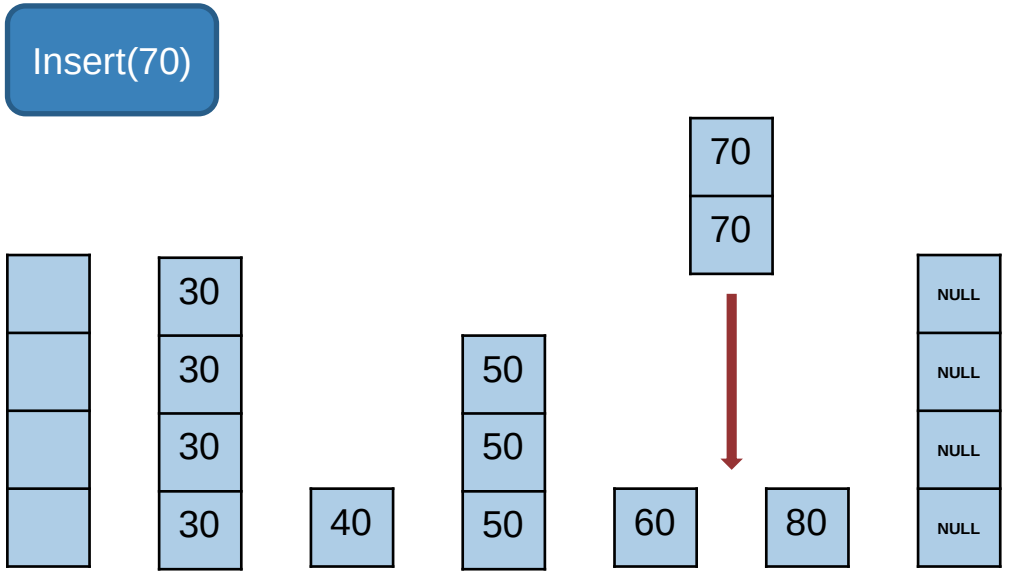
1. Randomized Skip List



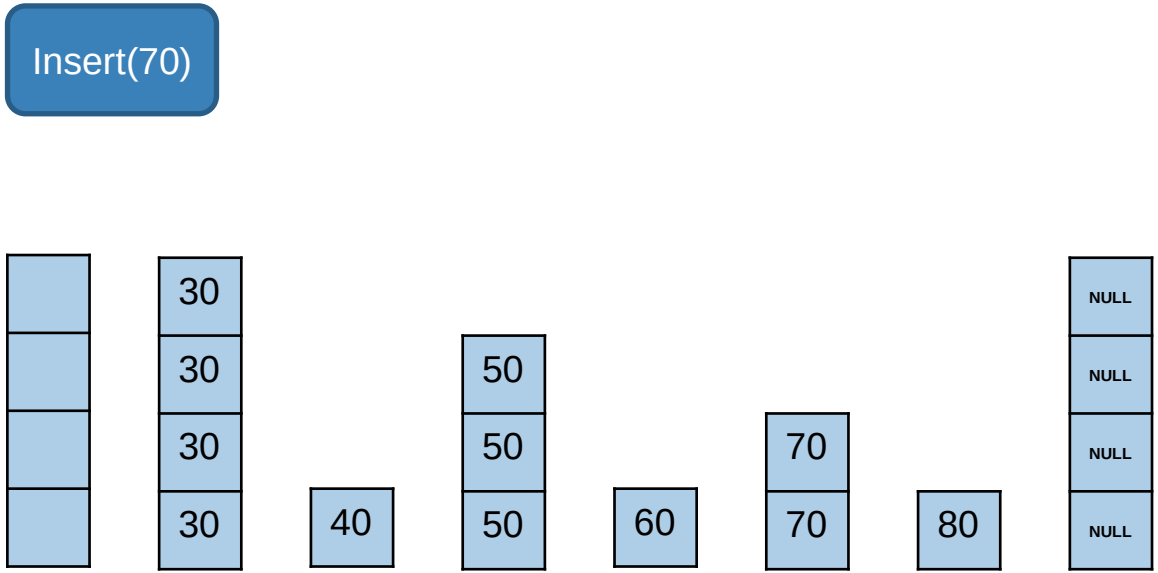
1. Randomized Skip List



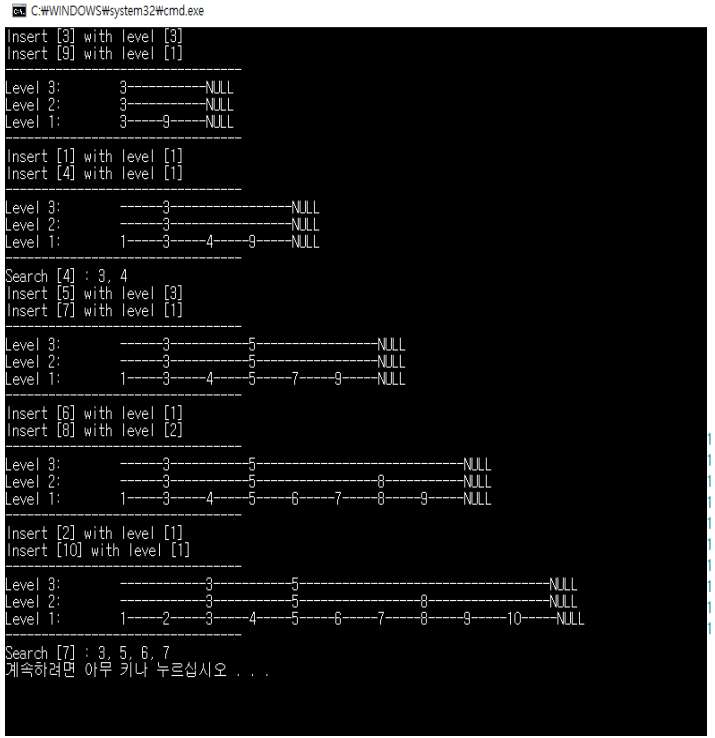
1. Randomized Skip List



1. Randomized Skip List



1. Randomized Skip List - 실습



- SkipNode 구조체 선언
- main 함수 정의

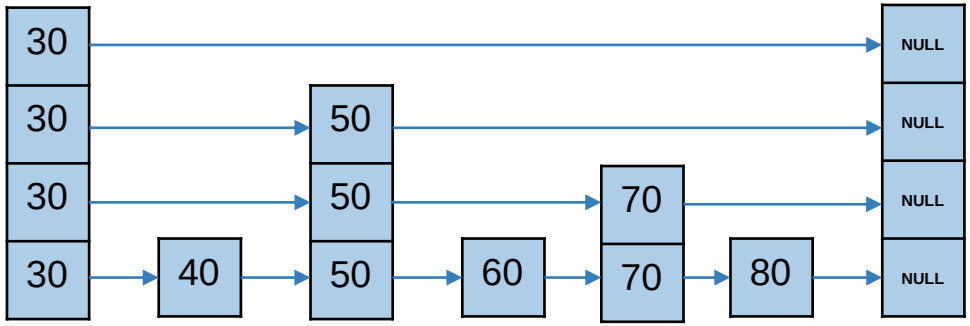
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX_LEVEL 3
#define MIN_DATA -9999

//SkipNode 구조체 선언
typedef struct SkipNode {
    int data;
    int level;
    struct SkipNode* next[MAX_LEVEL];
}SkipNode;

//SkipNode 관련 함수
void insertSkipNode(SkipNode** pHeadNode, int data);
void showSkipNode(SkipNode* pHeadNode);
void searchSkipNode(SkipNode* pHeadNode, int data);
```

```c
int main() {
    //랜덤함수 시드 초기화
    srand(time(NULL));

    //Skip List 생성
    SkipNode* SkipList = (SkipNode*)malloc(sizeof(SkipNode));
    SkipList->level = MAX_LEVEL;
    SkipList->data = MIN_DATA;
    for (int i = 0; i < SkipList->level; i++)
        SkipList->next[i] = NULL;

    insertSkipNode(&SkipList, 3);
    insertSkipNode(&SkipList, 9);
    showSkipNode(SkipList);

    insertSkipNode(&SkipList, 1);
    insertSkipNode(&SkipList, 4);
    showSkipNode(SkipList);

    searchSkipNode(SkipList, 4);

    insertSkipNode(&SkipList, 5);
    insertSkipNode(&SkipList, 7);
    showSkipNode(SkipList);

    insertSkipNode(&SkipList, 6);
    insertSkipNode(&SkipList, 8);
    showSkipNode(SkipList);

    insertSkipNode(&SkipList, 2);
    insertSkipNode(&SkipList, 10);
    showSkipNode(SkipList);

    searchSkipNode(SkipList, 7);

}
```

1. Randomized Skip List - 실습

- **show 함수 정의**

데이터를 List의 모든 node를 level별로 출력

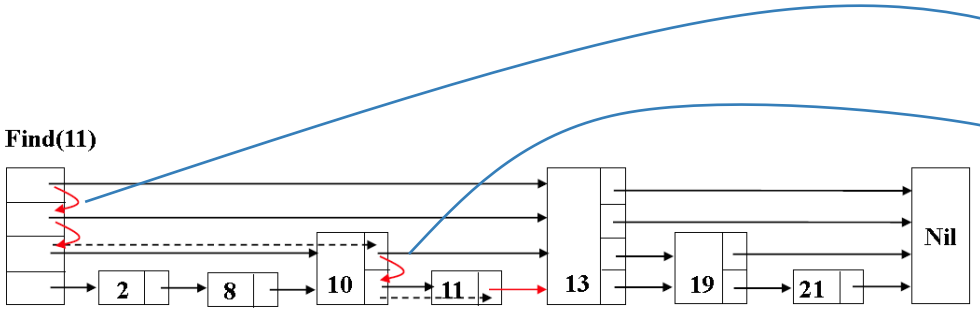



```
101  void showSkipNode(SkipNode* pHeadNode)
102  {
103      SkipNode* pTmpNode = NULL;
104      int i,j;
105
106      //Level 별로 끝에 도달할 때까지 데이터 출력
107      printf("--------------------------------₩n");
108      for (i = MAX_LEVEL; i >0;i--)
109      {
110          pTmpNode = pHeadNode->next[0];
111          printf("Level %d:₩t", i);
112          while (pTmpNode != NULL)
113          {
114              if (pTmpNode->level >= i)
115                  printf("%d-----", pTmpNode->data);
116              else
117                  printf("------");
118              pTmpNode = pTmpNode->next[0];
119          }
120          printf("NULL₩n");
121      }
122      printf("--------------------------------₩n");
123  }
```

1. Randomized Skip List - 실습

- **search 함수 정의**

찾고자 하는 데이터 값을 갖는 노드를 탐색하는 과정을 출력

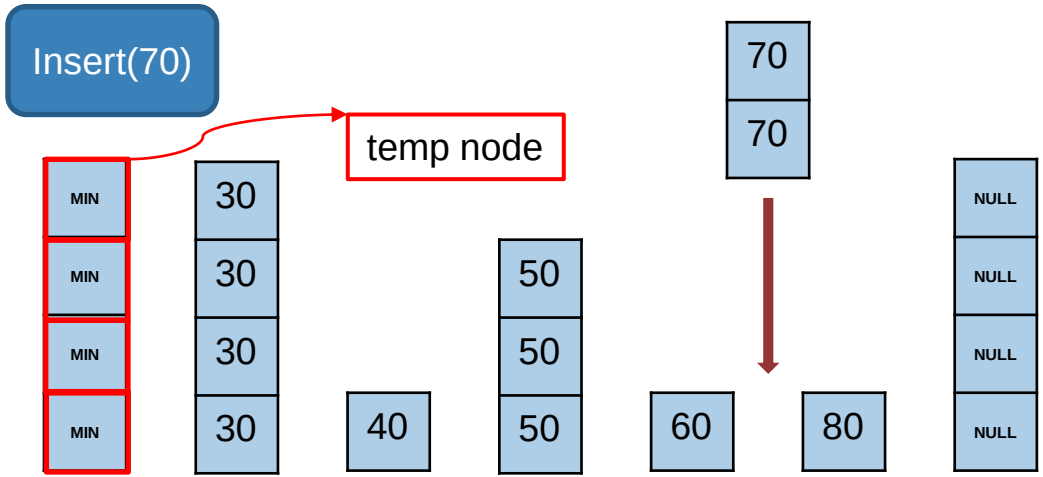



```
125  void searchSkipNode(SkipNode* pHeadNode, int data)
126  {
127      int pos = MAX_LEVEL-1;
128      SkipNode* pTmpNode = pHeadNode->next[pos];
129
130      //현재 Level에 속한 노드가 없거나 data가 찾고자 하는 data 보다 클 경우 Level 감소
131      while (pTmpNode==NULL || pTmpNode->data > data)
132          pTmpNode = pHeadNode->next[--pos];
133
134      printf("Search [%d] : ", data);
135      while (pTmpNode->data != data)
136      {
137          //탐색을 하다가 Node가 끝나거나 찾고자 하는 data가 없을 경우 Level 감소
138          if (pTmpNode->next[pos] == NULL || pTmpNode->next[pos]->data > data)
139              pos -= 1;
140          else
141          {
142              printf("%d, ", pTmpNode->data);
143              pTmpNode = pTmpNode->next[pos];
144              if (pTmpNode->data == data)
145              {
146                  printf("%d₩n", pTmpNode->data);
147                  return;
148              }
149          }
150      }
151      printf(",,, there is no %d₩n", data);
152      printf("--------------------------------₩n");
153  }
```

1. Randomized Skip List - 실습

- **insert 함수 정의**

1. Level별로 탐색을 하기 위한 temp 노드 생성
2. 탐색하는 과정과 유사하게 집어 넣을 직전 노드로 이동
3. Coin Flip을 통해 추가할 노드의 Level을 지정
4. 탐색한 temp 노드에 대하여 새로 추가할 노드와 연결

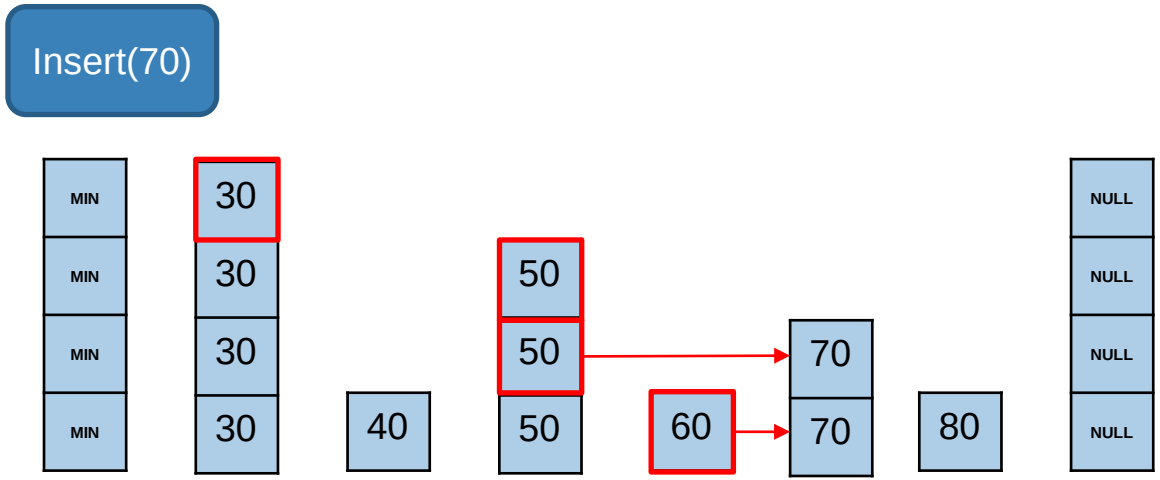



```
56  void insertSkipNode(SkipNode** pHeadNode, int data)
57  {
58      int level = 1, i, pos = MAX_LEVEL;
59      SkipNode* pTmpNode[MAX_LEVEL];
60      SkipNode* pNewNode;
61
62      //MAX_LEVEL에서 내려가면서 찾을 temp 노드 변수 초기화
63      for(i=0;i<MAX_LEVEL;i++)
64          pTmpNode[i] = *pHeadNode;
65
66      for (i = MAX_LEVEL - 1; i >= 0; i--)
67      {
68          //현재 Level에서 집어 넣을 직전 노드로 이동
69          
70
71
72
73
74
75      }
76
77      //추가할 노드의 최대 레벨을 계산(coin flip)
78      while (rand() % 2)
79      {
80          level++;
81          if (level >= MAX_LEVEL)
82              break;
83      }
84      //추가할 노드 동적 할당 및 초기화
85      pNewNode = (SkipNode*)malloc(sizeof(SkipNode));
86      pNewNode->level = level;
87      pNewNode->data = data;
88      for (i = 0; i < MAX_LEVEL; i++)
89          pNewNode->next[i] = NULL;
90
91      //추가할 노드의 Level만큼 앞 뒤 연결된 노드들을 연결
92      for (i = pNewNode->level - 1; i >= 0; i--)
93      {
94          
95
96      }
97
98
99      printf("Insert [%d] with level [%d]₩n", data,level);
100 }
```

1. Randomized Skip List - 실습

- **insert 함수 정의**

1. Level별로 탐색을 하기 위한 temp 노드 생성
2. 탐색하는 과정과 유사하게 집어 넣을 직전 노드로 이동
3. Coin Flip을 통해 추가할 노드의 Level을 지정
4. 탐색한 temp 노드에 대하여 새로 추가할 노드와 연결

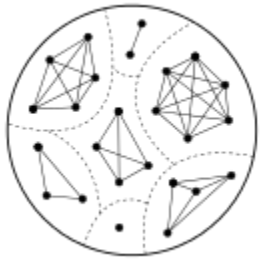



```
56  void insertSkipNode(SkipNode** pHeadNode, int data)
57  {
58      int level = 1, i, pos = MAX_LEVEL;
59      SkipNode* pTmpNode[MAX_LEVEL];
60      SkipNode* pNewNode;
61
62      //MAX_LEVEL에서 내려가면서 찾을 temp 노드 변수 초기화
63      for(i=0;i<MAX_LEVEL;i++)
64          pTmpNode[i] = *pHeadNode;
65
66      for (i = MAX_LEVEL - 1; i >= 0; i--)
67      {
68          //현재 Level에서 집어 넣을 직전 노드로 이동
69          
70
71
72
73
74
75      }
76
77      //추가할 노드의 최대 레벨을 계산(coin flip)
78      while (rand() % 2)
79      {
80          level++;
81          if (level >= MAX_LEVEL)
82              break;
83      }
84      //추가할 노드 동적 할당 및 초기화
85      pNewNode = (SkipNode*)malloc(sizeof(SkipNode));
86      pNewNode->level = level;
87      pNewNode->data = data;
88      for (i = 0; i < MAX_LEVEL; i++)
89          pNewNode->next[i] = NULL;
90
91      //추가할 노드의 Level만큼 앞 뒤 연결된 노드들을 연결
92      for (i = pNewNode->level - 1; i >= 0; i--)
93      {
94          
95
96      }
97
98
99      printf("Insert [%d] with level [%d]₩n", data,level);
100 }
```

2. Equivalence classes

- **정의: 연관이 있는 원소들을 한 집합으로 묶은 것 (엄밀한 정의는 이론수업 참조)**

- 그래프에서 연결된 노드들을 군집화 할 때 사용



- 구현 형태: Linked List로 구현
  - 본 실습에서는 이론수업PPT와 같이 Linked List의 배열을 사용

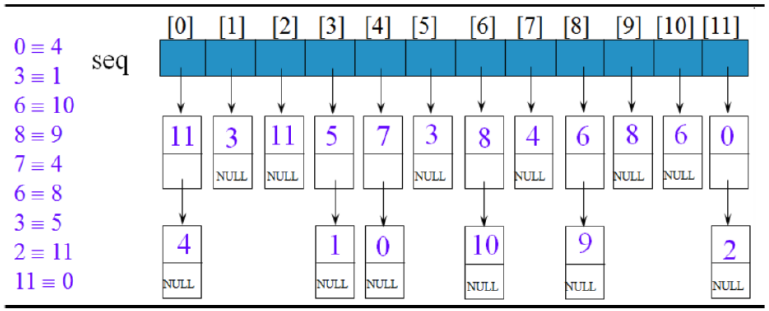
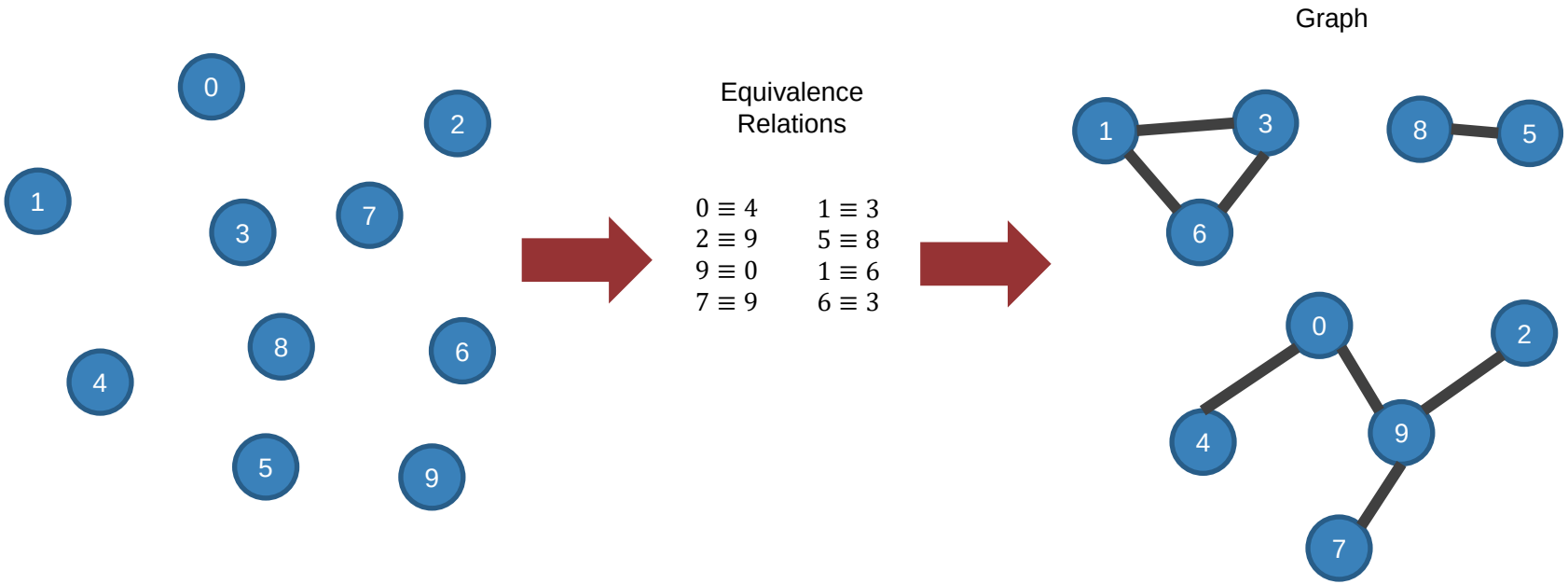


Figure 4.16 : Lists after pairs have been input

2. Equivalence Classes

- 동치관계는 그래프 상의 연결여부로 표현할 수 있다.



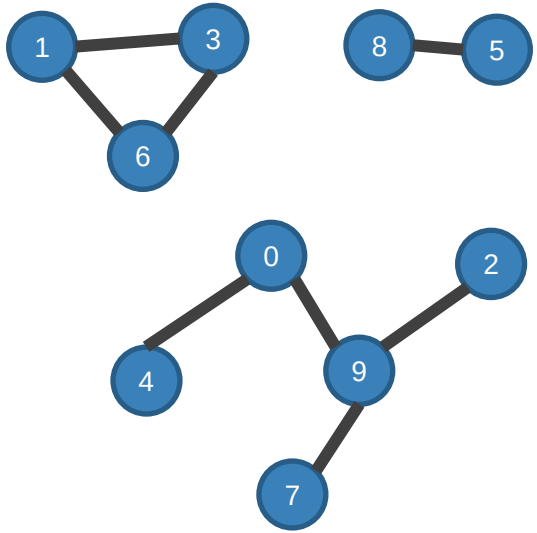
2. Equivalence Classes

- 동치관계는 그래프 상의 연결여부로 표현할 수 있다.

Equivalence Relations

$0 \equiv 4$ $1 \equiv 3$
$2 \equiv 9$ $5 \equiv 8$
$9 \equiv 0$ $1 \equiv 6$
$7 \equiv 9$ $6 \equiv 3$

Graph



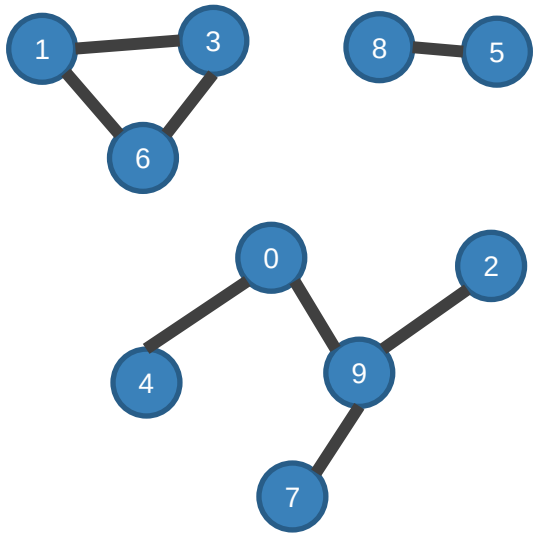
2. Equivalence Classes

- 동치류의 원소들은 동치류에 속하는 노드와 연결된 모든 노드를 방문하여 알 수 있다.

Equivalence Relations

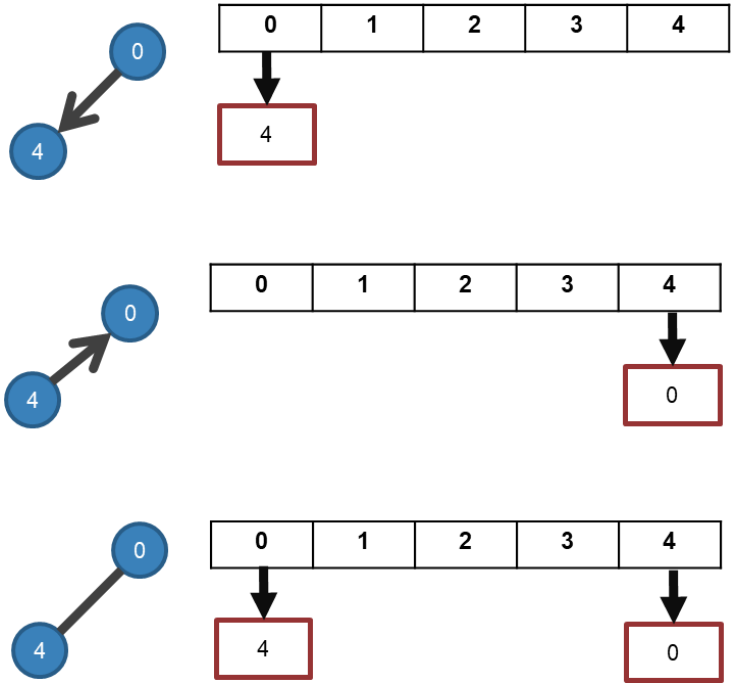
$0 \equiv 4$ $1 \equiv 3$
$2 \equiv 9$ $5 \equiv 8$
$9 \equiv 0$ $1 \equiv 6$
$7 \equiv 9$ $6 \equiv 3$

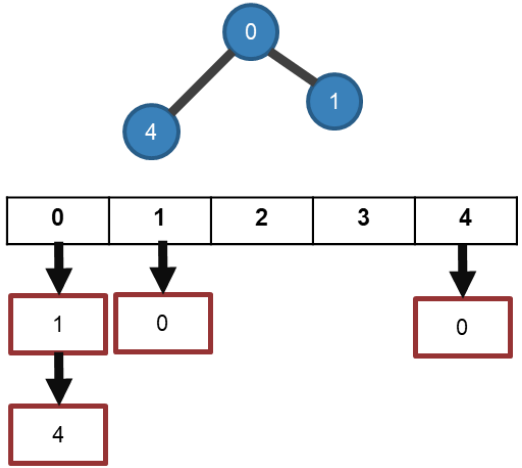
Graph



2. Equivalence Classes

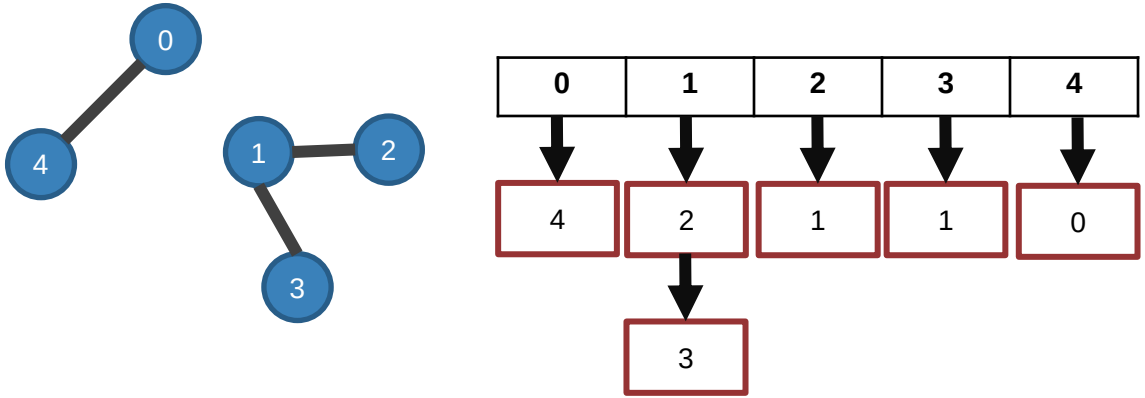
- 그래프는 여러 개의 리스트로 표현될 수 있다.





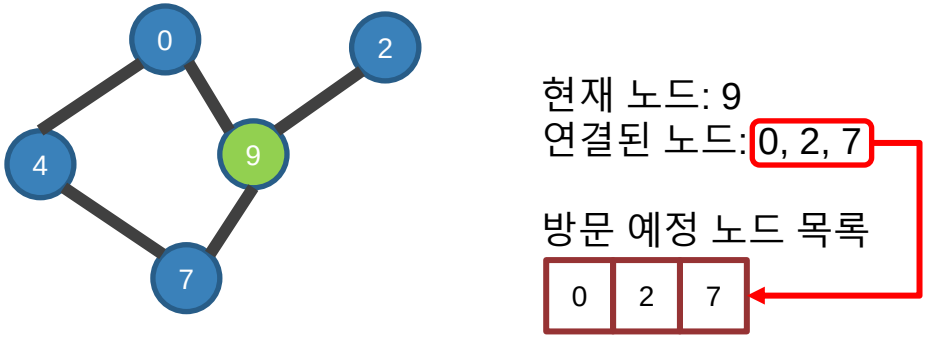
2. Equivalence Classes

- 그래프는 여러 개의 리스트로 표현될 수 있다.



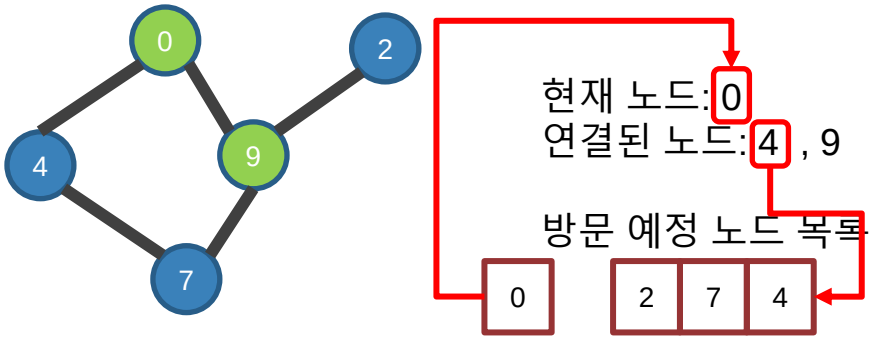
2. Equivalence Classes

- 그래프에서 연결된 노드들을 한번씩 방문하면 동치류의 모든 원소를 알 수 있다.
  - 하나의 노드에 방문하면 연결된 노드를 방문할 노드에 기록해 놓은 뒤 나중에 방문함.
  - 이미 방문예정목록에 있는 노드/방문했던 노드면 목록에 추가하지 않음.



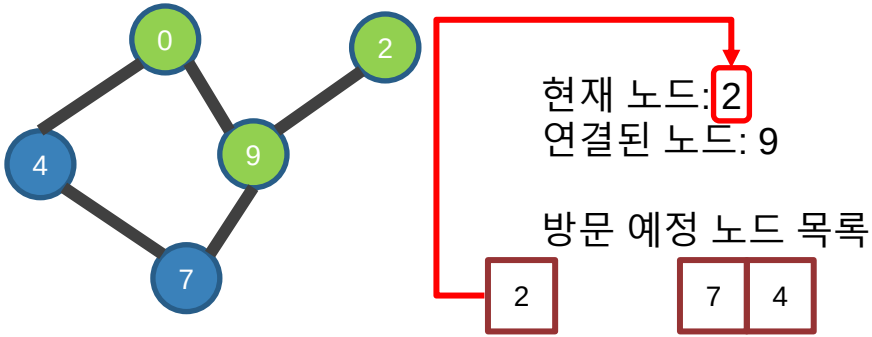
2. Equivalence Classes

- 그래프에서 연결된 노드들을 한번씩 방문하면 동치류의 모든 원소를 알 수 있다.
  - 하나의 노드에 방문하면 연결된 노드를 방문할 노드에 기록해 놓은 뒤 나중에 방문함.
  - 이미 방문예정목록에 있는 노드/방문했던 노드면 목록에 추가하지 않음.



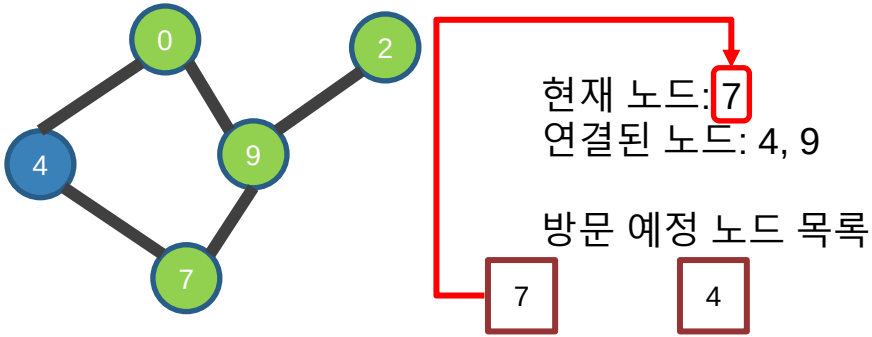
2. Equivalence Classes

- 그래프에서 연결된 노드들을 한번씩 방문하면 동치류의 모든 원소를 알 수 있다.
  - 하나의 노드에 방문하면 연결된 노드를 방문할 노드에 기록해 놓은 뒤 나중에 방문함.
  - 이미 방문예정목록에 있는 노드/방문했던 노드면 목록에 추가하지 않음.



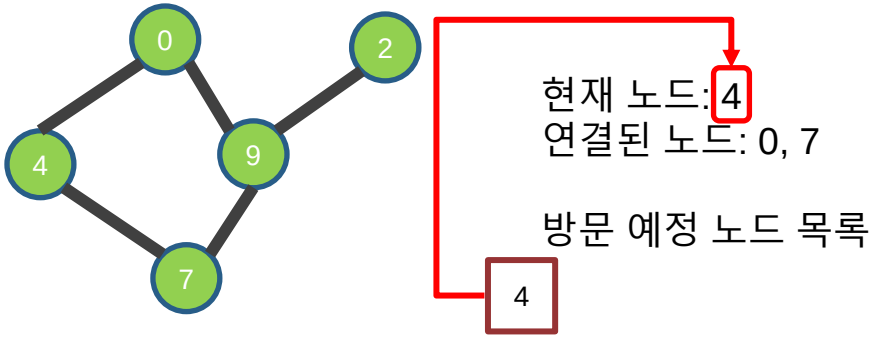
2. Equivalence Classes

- 그래프에서 연결된 노드들을 한번씩 방문하면 동치류의 모든 원소를 알 수 있다.
  - 하나의 노드에 방문하면 연결된 노드를 방문할 노드에 기록해 놓은 뒤 나중에 방문함.
  - 이미 방문예정목록에 있는 노드/방문했던 노드면 목록에 추가하지 않음.



2. Equivalence Classes

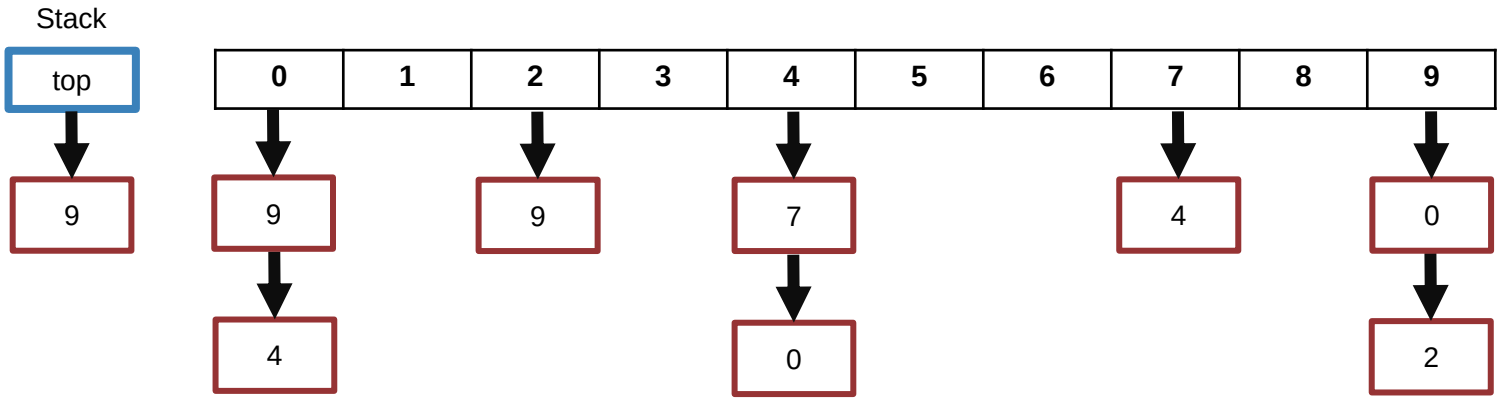
- 그래프에서 연결된 노드들을 한번씩 방문하면 동치류의 모든 원소를 알 수 있다.
  - 하나의 노드에 방문하면 연결된 노드를 방문할 노드에 기록해 놓은 뒤 나중에 방문함.
  - 이미 방문예정목록에 있는 노드/방문했던 노드면 목록에 추가하지 않음.



Enumerating equivalence class including 9.

Equivalence Relations

$0 \equiv 4$
$7 \equiv 4$
$2 \equiv 9$
$9 \equiv 0$



방문할 노드(9)를 스택에 push

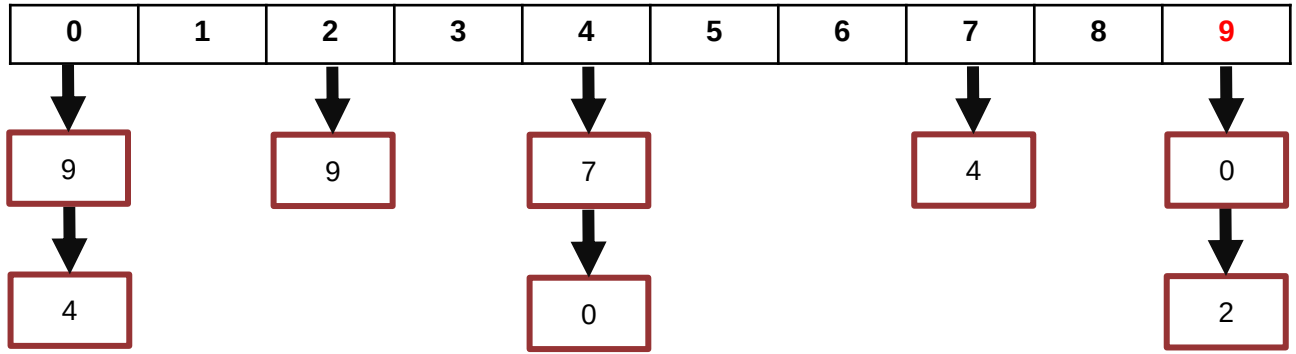
Enumerating equivalence class including 9.

Equivalence Relations

$0 \equiv 4$
$7 \equiv 4$
$2 \equiv 9$
$9 \equiv 0$

Stack

top



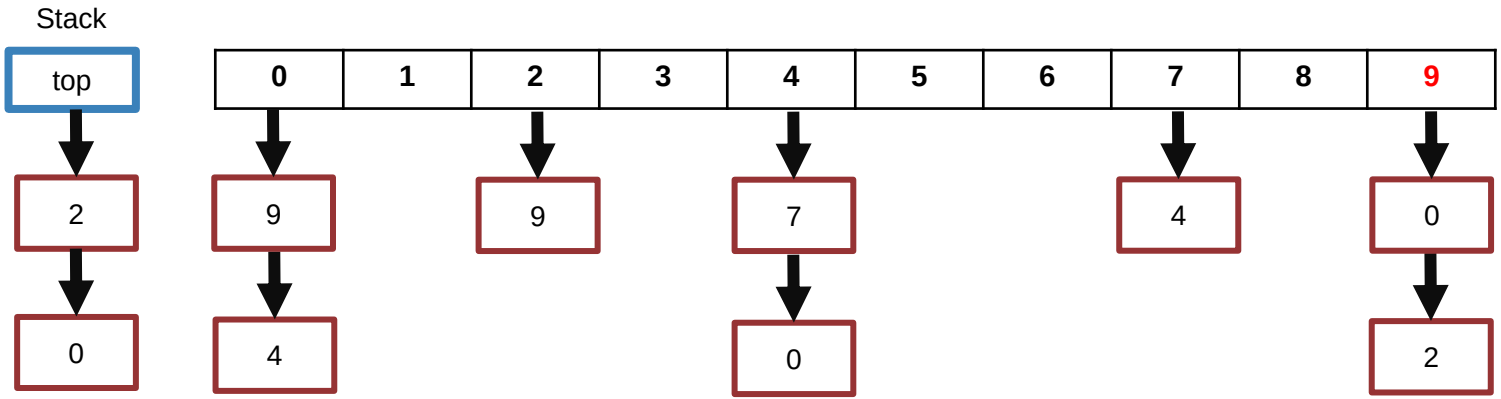
9

스택에서 노드를 pop하여 노드(9) 방문

Enumerating equivalence class including 9.

Equivalence Relations

$0 \equiv 4$
$7 \equiv 4$
$2 \equiv 9$
$9 \equiv 0$

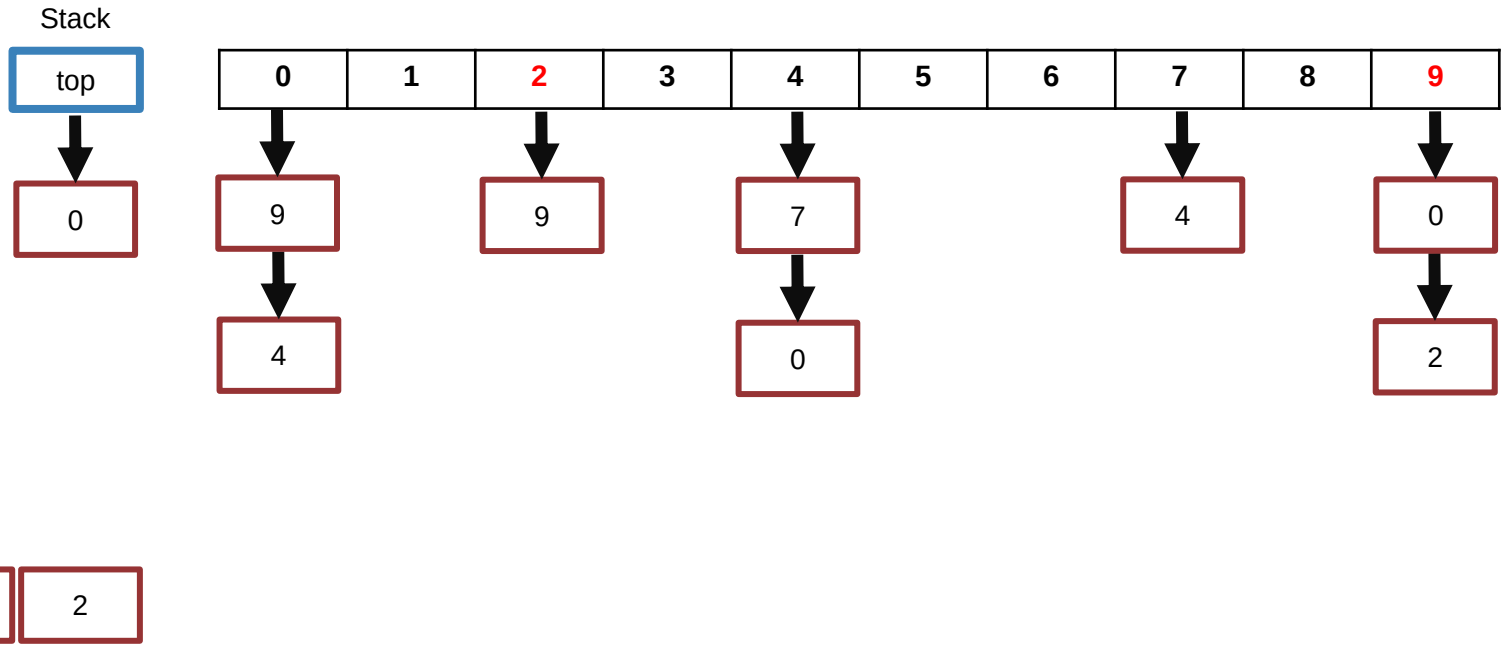


방문한 노드(9)에 연결된 노드(2, 0)를 스택에 push

Enumerating equivalence class including 9.

Equivalence Relations

$0 \equiv 4$
$7 \equiv 4$
$2 \equiv 9$
$9 \equiv 0$



스택에서 노드를 pop하여 노드(2) 방문 및 연결된 노드를 스택에 push

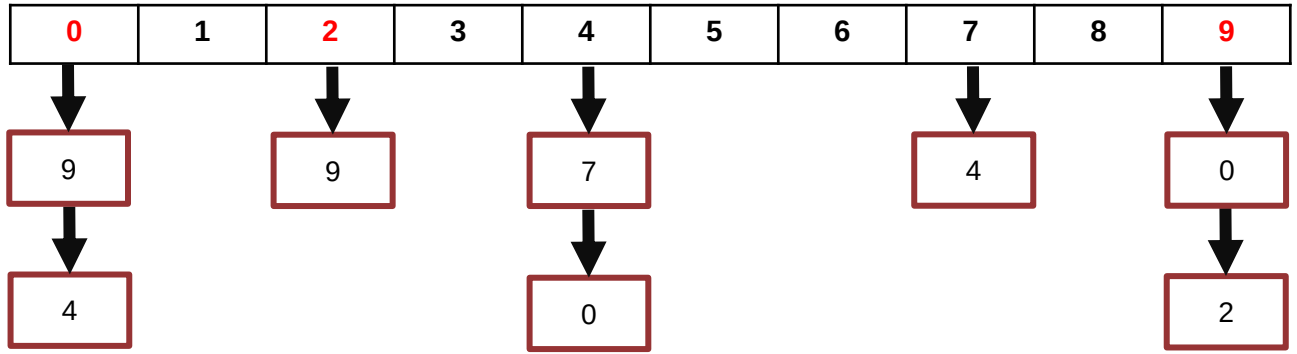
Enumerating equivalence class including 9.

Equivalence Relations

$0 \equiv 4$
$7 \equiv 4$
$2 \equiv 9$
$9 \equiv 0$

Stack

top

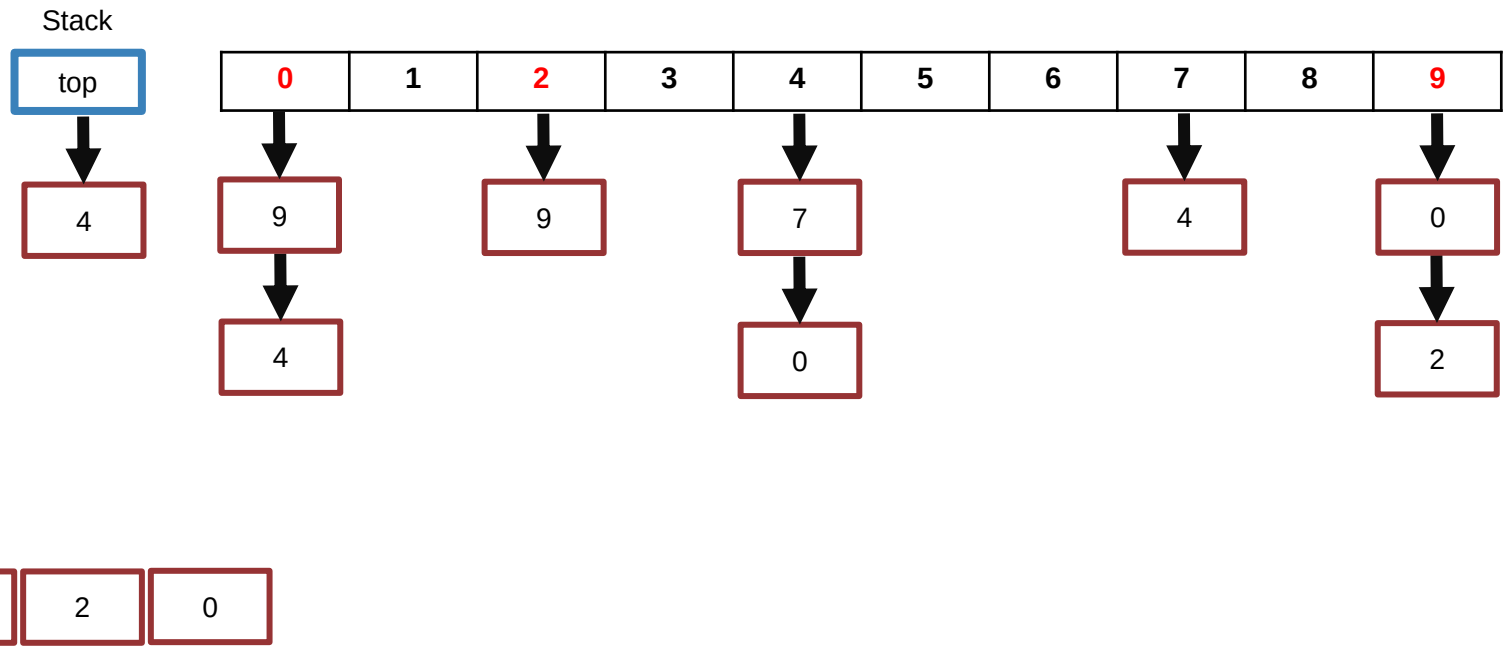


스택에서 노드를 pop하여 노드(0) 방문

Enumerating equivalence class including 9.

Equivalence Relations

$0 \equiv 4$
$7 \equiv 4$
$2 \equiv 9$
$9 \equiv 0$



방문한 노드(0)에 연결된 노드(4)를 스택에 push

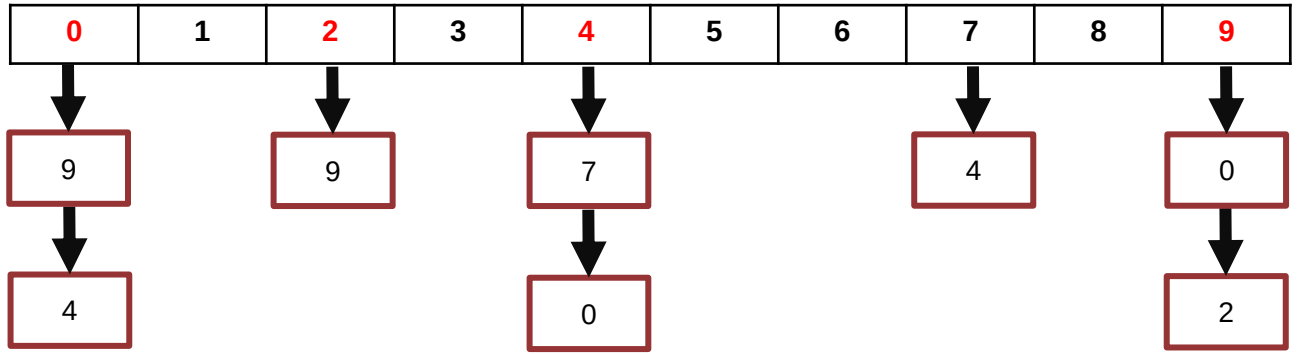
Enumerating equivalence class including 9.

Equivalence Relations

$0 \equiv 4$
$7 \equiv 4$
$2 \equiv 9$
$9 \equiv 0$

Stack

top

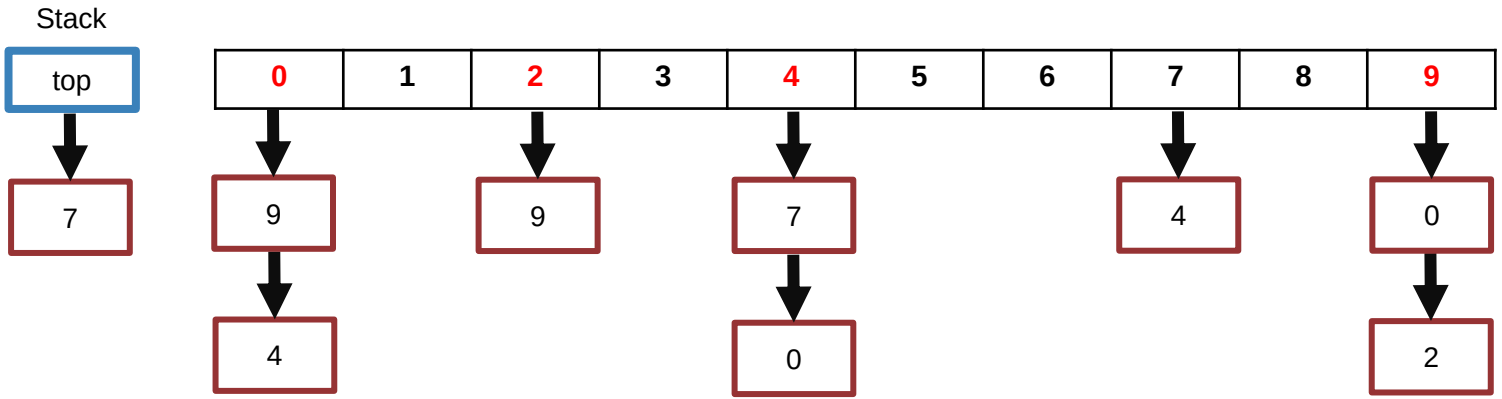


스택에서 노드를 pop하여 노드(4) 방문

Enumerating equivalence class including 9.

Equivalence Relations

$0 \equiv 4$
$7 \equiv 4$
$2 \equiv 9$
$9 \equiv 0$



방문한 노드(4)에 연결된 노드(7)를 스택에 push

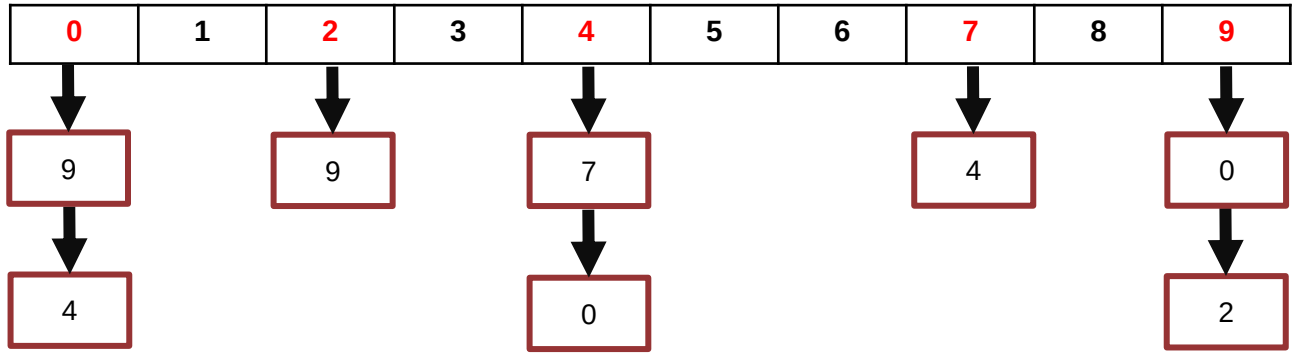
Enumerating equivalence class including 9.

Equivalence Relations

$0 \equiv 4$
$7 \equiv 4$
$2 \equiv 9$
$9 \equiv 0$

Stack

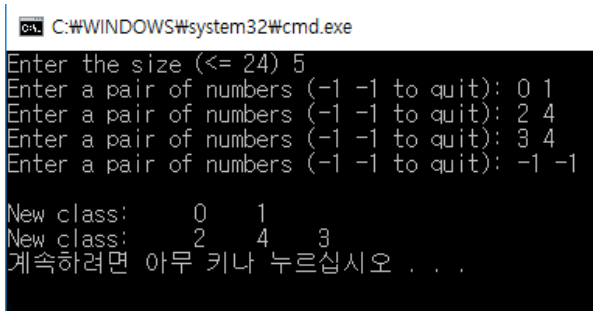
top



스택에서 노드를 pop하여 노드(7) 방문

2. Equivalence classes - 실습

- Node 구조체 선언
- 변수초기화




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

#define MAX_SIZE 24
#define FALSE 0
#define TRUE 1

//구조체 선언
typedef struct node *nodePointer;
typedef struct node {
    int data;
    nodePointer link;
};

int main() {
    int out[MAX_SIZE];
    nodePointer seq[MAX_SIZE];
    nodePointer x, y, top;
    int i, j, n;

    printf("Enter the size (<= %d) ", MAX_SIZE);
    scanf("%d", &n);
    for (i = 0; i < n; i++) {

        out[i] = TRUE;
        seq[i] = NULL;
    }

```

2. Equivalence classes - 실습

- 숫자 쌍 입력
- i j 형식으로 입력하고 -1 -1을 입력하면 종료
- 입력을 받을 때마다 노드를 생성하고 리스트에 추가

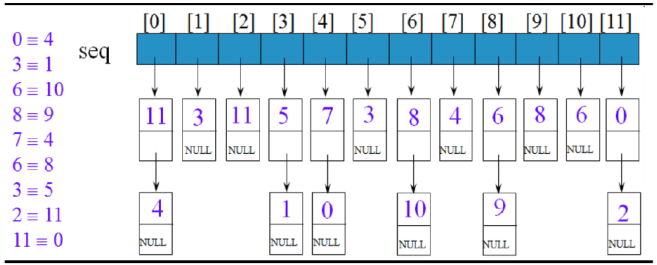


Figure 4.16 : Lists after pairs have been input

Equivalence classes (Cont.)

```
/* Phase 1: Input the equivalence pairs: */
printf("Enter a pair of numbers (-1 -1 to quit): ");
scanf("%d%d",&i,&j);
while (i >= 0) {
  MALLOC(x, sizeof(*x));
  x->data = j;  x->link = seq[i]; seq[i] = x;
  MALLOC(x, sizeof(*x));
  x->data = i;  x->link = seq[j]; seq[j] = x;
  printf("Enter a pair of numbers (-1 -1 to quit): ");
  scanf("%d%d",&i,&j);
}
```

Program 4.22_2 : Program to find equivalence classes

```
29  printf("Enter a pair of numbers (-1 -1 to quit): ");
30  scanf("%d%d", &i, &j);
31  while (i >= 0)
32  {
33      //새 노드를 만들어 j를 넣고 i번 노드에 추가
34
35
36
37
38
39      //새 노드를 만들어 i를 넣고 j번 노드에 추가
40
41
42
43
44
45      printf("Enter a pair of numbers (-1 -1 to quit): ");
46      scanf("%d%d", &i, &j);
47  }
```

2. Equivalence classes - 실습

- 숫자 쌍 입력
- i j 형식으로 입력하고 -1 -1을 입력하면 종료
- 입력을 받을 때마다 노드를 생성하고 리스트에 추가

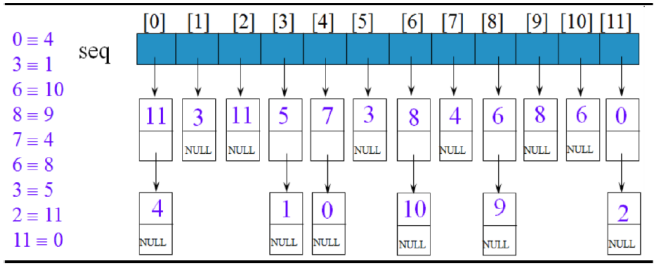


Figure 4.16 : Lists after pairs have been input

Equivalence classes (Cont.)

```
/* Phase 1: Input the equivalence pairs: */
printf("Enter a pair of numbers (-1 -1 to quit): ");
scanf("%d%d",&i,&j);
while (i >= 0) {
  MALLOC(x, sizeof(*x));
  x->data = j;  x->link = seq[i]; seq[i] = x;
  MALLOC(x, sizeof(*x));
  x->data = i;  x->link = seq[j]; seq[j] = x;
  printf("Enter a pair of numbers (-1 -1 to quit): ");
  scanf("%d%d",&i,&j);
}
```

Program 4.22_2 : Program to find equivalence classes

```
29    printf("Enter a pair of numbers (-1 -1 to quit): ");
30    scanf("%d%d", &i, &j);
31    while (i >= 0)
32    {
33        //새 노드를 만들어 j를 넣고 i번 노드에 추가
34        x = (nodePointer)malloc(sizeof(node));
35        x->data = j;
36        x->link = seq[i];
37        seq[i] = x;
38
39        //새 노드를 만들어 i를 넣고 j번 노드에 추가
40        x = (nodePointer)malloc(sizeof(node));
41        x->data = i;
42        x->link = seq[j];
43        seq[j] = x;
44
45        printf("Enter a pair of numbers (-1 -1 to quit): ");
46        scanf("%d%d", &i, &j);
47    }
```

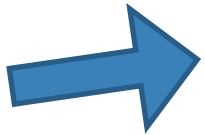
2. Equivalence classes - 실습

- 전부 탐색하며 클래스로 묶기
- 한 리스트를 탐색 중 아직 탐색되지 않은 리스트가 있으면 그 리스트로 이동해서 계속 탐색

Equivalence classes (Cont.)

```
/* Phase 2: output the equivalence classes */
for (i = 0; i < n; i++)
  if (out[i]) {
    printf("\nNew class: %5d",i);
    out[i] = FALSE;  /* set class to false */
    x = seq[i];  top = NULL; /* initialize stack */
    for(;;) {  /* find rest of class */
      while (x) { /* process list */
        j = x->data;
        if (out[j]) {
          printf("%5d",j);  out[j] = FALSE;
          y = x->link; x->link = top; top = x; x = y;
        }
        else x = x->link;
      }
      if (!top) break;
      x = seq[top->data];  top = top->link;
                           /* unstack */
    }
  }
}
```

Program 4.22_3 : Program to find equivalence classes



```
49      for (i = 0; i < n; i++)
50          if (out[i]) {
51              printf("₩nNew class: %5d", i);
52
53              //클래스 i를 탐색
54              out[i] = FALSE;
55              x = seq[i];
56              top = NULL;
57              for (;;) {
58                  while (x) {
59
60                      j = x->data;
61                      //아직 탐색되지 않았다면 위치를 저장하고 새로운 리스트로 이동
62                      if (out[j]) {
63                          printf("%5d", j);
64                          out[j] = FALSE;
65
66
67
68
69
70                      }
71                      //이미 탐색되었다면 통과
72                      else
73
74                  }
75                  //top이 FALSE이면 전부 탐색한것
76                  if (!top)
77                      break;
78                  //위치를 저장해뒀던 top으로 이동
79                  x = seq[top->data];
80                  top = top->link;
81              }
82          }
83      printf("%₩n");
84  }
```

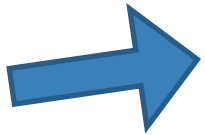
2. Equivalence classes - 실습

- 전부 탐색하며 클래스로 묶기
- 한 리스트를 탐색 중 아직 탐색되지 않은 리스트가 있으면 그 리스트로 이동해서 계속 탐색

Equivalence classes (Cont.)

```
/* Phase 2: output the equivalence classes */
for (i = 0; i < n; i++)
  if (out[i]) {
    printf("\nNew class: %5d",i);
    out[i] = FALSE;  /* set class to false */
    x = seq[i];  top = NULL; /* initialize stack */
    for(;;) {  /* find rest of class */
      while (x) { /* process list */
        j = x->data;
        if (out[j]) {
          printf("%5d",j);  out[j] = FALSE;
          y = x->link; x->link = top; top = x; x = y;
        }
        else x = x->link;
      }
      if (!top) break;
      x = seq[top->data];  top = top->link;
                           /* unstack */
    }
  }
}
```

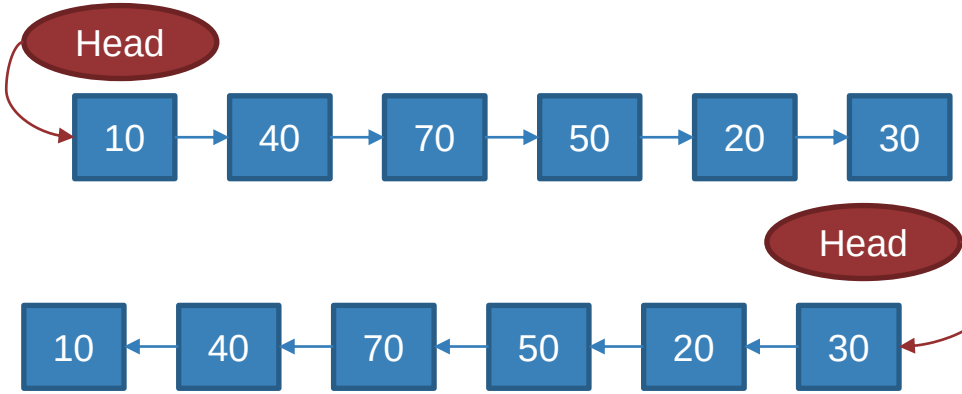
Program 4.22_3 : Program to find equivalence classes



```
49      for (i = 0; i < n; i++)
50          if (out[i]) {
51              printf("₩nNew class: %5d", i);
52
53              //클래스 i를 탐색
54              out[i] = FALSE;
55              x = seq[i];
56              top = NULL;
57              for (;;) {
58                  while (x) {
59
60                      j = x->data;
61                      //아직 탐색되지 않았다면 위치를 저장하고 새로운 리스트로 이동
62                      if (out[j]) {
63                          printf("%5d", j);
64                          out[j] = FALSE;
65
66                          y = x->link;
67                          x->link = top;
68                          top = x;
69                          x = y;
70                      }
71                      //이미 탐색되었다면 통과
72                      else
73                          x = x->link;
74                  }
75                  //top이 FALSE이면 전부 탐색한것
76                  if (!top)
77                      break;
78                  //위치를 저장해뒀던 top으로 이동
79                  x = seq[top->data];
80                  top = top->link;
81              }
82          }
83      printf("%₩n");
84  }
```

3. Reversing Linked List - 과제

- Linked List내의 Node들의 순서를 거꾸로 만드는 함수를 작성
- 실습 4에서 사용한 Linked List 실습과 실습 5에서 사용한 Stack 실습 코드를 이용하면 해결 가능
- main 함수는 실습 4에서 사용된 main 코드를 이용 (오른쪽 메인 함수 이용)




```
int main()
{
    int pos;
    LinkedList* linkedList = (LinkedList*)malloc(sizeof(LinkedList));
    linkedList->curCount = 0;
    linkedList->headNode.nextNode = NULL;

    StackNode* top = NULL;
    StackNode* pNode;

    //showNode(linkedList);
    addNode(linkedList, 0, 10);
    addNode(linkedList, 5, 100);
    addNode(linkedList, 1, 20);
    addNode(linkedList, 2, 30);
    addNode(linkedList, 1, 50);
    addNode(linkedList, 1, 70);
    addNode(linkedList, 1, 40);

    showNode(linkedList);

    reverseList(linkedList, &top);

    showNode(linkedList);

    //removeNode(linkedList, 1);
    //showNode(linkedList);

    //pos = findPos(linkedList, 30);
    //printf("the location of node with data '30': %d\n", pos);

    makeEmpty(linkedList);
    showNode(linkedList);
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
addNode() error2: 추가 범위 초과
현재 Node 개수 : 6
[10]
[40]
[70]
[50]
[20]
[30]
--------------------------------
Reverse Linked List!
현재 Node 개수 : 6
[30]
[20]
[50]
[70]
[40]
[10]
--------------------------------
현재 Node 개수 : 0
--------------------------------
계속하려면 아무 키나 누르십시오 . . .
```

3. Reversing Linked List - 과제

- 실습 4 Linked List 실습 코드에 실습 5에서 사용된 Stack 함수를 가져와 활용용
- 부가적으로 'reverseList' 라는 함수를 선언 및 정의해야 함

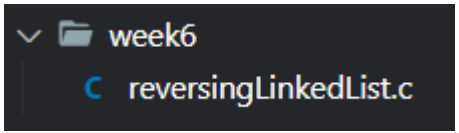
```
void reverseList(LinkedList* pList, StackNode** top) {
    Node *pNode = NULL;
    StackNode *sNode = NULL;

    printf("Reverse Linked List!₩n");
    //Stack에 List 저장

    //List에 Stack 저장

}
```

감사합니다.

과제 제출 기한: 2023년 4월 13일 23:59분 까지

*기한 내에 본인의 원격 저장소에 push된 건에 대해서만 인정

파일 이름: "week6/reversingLinkedList.c"



궁금한 것이 생기면 언제든지 질문하시면 됩니다 ☺

- 공업센터본관 304호로 방문하시거나
- hameli@hanyang.ac.kr 로 연락 바랍니다.
- 담당조교: 백승한