자료구조 (특별)

Chap 06. 집합과 맵 (최다 단어 찾기)

2019년 하계 특별 학기

컴퓨터과학과
민경하

Contents

6. 집합과 맵 (최다 단어 찾기)

6.1 집합

6.2 맵

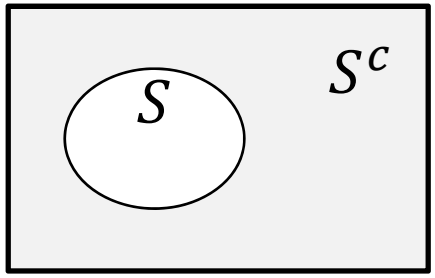
6.3 맵의 구현

6.4 최다 빈도수 단어 찾기

6.1 집합

- 집합 (set)의 정의
  - 구체적으로 정의할 수 있는 원소들의 모임
  - Cantor의 역설을 피할 수 있는 원소들의 모임

$$S = \{x | x \in S^c\}$$



  - 성질 1:동일한 원소의 반복을 허용하지 않음

$$\{0, 1, 3, 5\} = \{0, 1, 1, 1, 3, 3, 5\}$$

  - 성질 2: 원소의 순서를 고려하지 않음

$$\{0, 1, 3, 5\} = \{0, 5, 1, 3\}$$

6.1 집합

- 집합의 연산
  - Same
    $$\{0, 1, 3, 5\} = \{0, 5, 1, 3\}$$
  - Find
    $$5 \in \{0, 5, 1, 3\}$$
  - Union
    $$\{2, 4, 5, 7\} \cup \{0, 1, 3, 5\}$$
  - Intersection
    $$\{2, 4, 5, 7\} \cap \{0, 1, 3, 5\}$$
  - Insertion
    $$7 + \{0, 1, 3, 5\}$$
  - Deletion
    $$\{0, 1, 3, 5\} - 3$$

6.1 집합

- 집합의 구현 (1)
  - 배열 (또는 연결 리스트)
    - 정렬되지 않은 배열에 저장
    - 하나의 집합에 대해서 다양한 배열이 존재할 수 있음

$$\{0, 5, 1, 3, 9, 7, 2\}$$

0	5	1	3	9	7	2

||

0	1	5	2	9	7	3

6.1 집합

- 집합의 구현 (2)
  - 정렬된 배열 (또는 연렬 리스트)
    - 정렬된 배열에 저장
    - 하나의 집합에 대해서 하나의 배열만 존재함

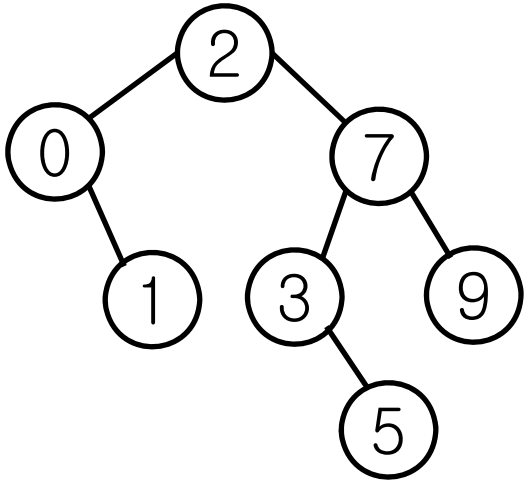
$$\{0, 5, 1, 3, 9, 7, 2\}$$

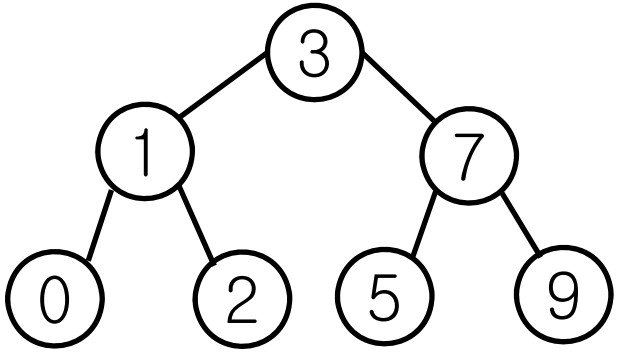
0	1	2	3	5	7	9

6.1 집합

- 집합의 구현 (3)
  - 이진 탐색 트리
    - 이진 탐색 트리에 저장
    - 하나의 집합에 대해서 다양한 이진 탐색 트리가 존재할 수 있음

$$\{0, 5, 1, 3, 9, 7, 2\}$$





6.1 집합

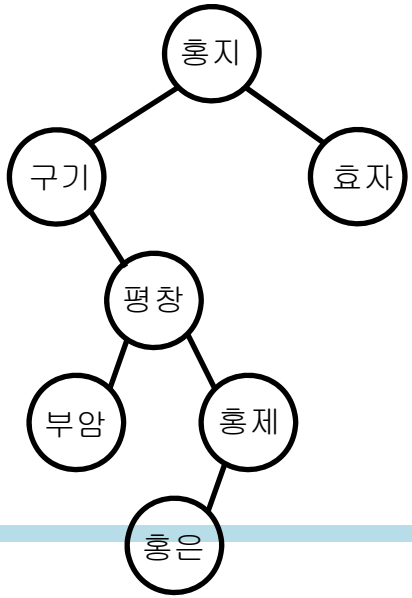
- 집합의 구현
  - 배열, 정렬된 배열, 이진 탐색 트리
  - 연산의 효율성 비교

	배열	정렬된 배열	이진 탐색 트리
Same	$O(n^2)$	O(n)	O(n)
Find	O(n)	O(log n)	O(log n)
Union	$O(n^2)$	O(n)	O(n)
Intersection	$O(n^2)$	O(n)	O(n)
Insertion	O(n)	O(n)	O(log n)
Deletion	O(n)	O(n)	O(log n)

6.1 집합

- 집합의 응용
  - 주거지 리스트를 작성하시오.

```
townset ← EMPTY;
for all lines in the list
    town ← line.get_town (“주거지”);
    townset.insert ( town );
townset.print ( );
```



	A	B	C	D
1		이름	주거지	전화번호
2	1	유연수	홍지동	010-2019-1711
3	2	김연지	홍지동	010-9765-4321
4	3	박영곤	홍지동	010-7123-9090
5	4	심기현	구기동	010-5491-7852
6	5	도일수	구기동	010-2210-9757
7	6	반규원	구기동	010-8852-7195
8	7	십진수	평창동	010-8572-2124
9	8	장거한	구기동	010-5278-1209
10	9	한수연	홍지동	010-2129-3744
11	10	천태백	홍지동	010-4296-4778
12	11	유정인	홍지동	010-3889-7451
13	12	풍상현	홍지동	010-6951-9114
14	13	전나래	홍지동	010-3329-2177
15	14	이기현	구기동	010-1818-1414
16	15	연아린	구기동	010-6541-0921
17	16	명지수	효자동	010-3789-5117
18	17	강용권	평창동	010-9006-2215
19	18	문효정	홍제동	010-5549-7532
20	19	김중선	효자동	010-5225-2521
21	20	단주성	부암동	010-8483-4239
22	21	사지정	효자동	010-7291-9265
23	22	김바름	구기동	010-2318-5821
24	23	주한미	홍은동	010-4299-8796
25	24	김무진	부암동	010-1212-1212

6.2 맵

- 맵 (map)의 정의
  - Key + Value
  - 집합의 원소 (key)에 속성 (value)를 결합한 개념
  - 연산
    - 집합의 연산 + count

6.2 맵

- 맵 (map)의 자료 구조
  - 배열 사용

```
class element {
    int key;
    int value;
};

int n;
int cnt;
element *map;
```

6.2 맵

- 맵 (map)의 자료 구조
  - 이진 탐색 트리 사용

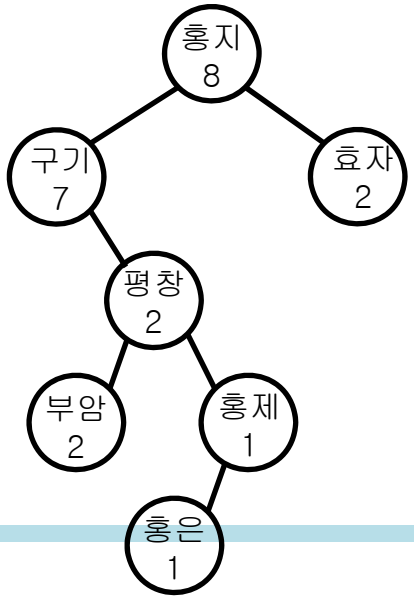
```
typedef class node *nptr;
class node {
    int key;
    int value;
    nptr lchild, rchild;
};

node map;
```

  - 모든 node의 value는 1보다 크거나 같음

6.2 맵

- 맵의 응용
  - 가장 많이 거주하는 주거지는 어디인가?

```
townmap ← EMPTY;
for all lines in the list
    town ← line.get_town (“주거지”);
    townmap.insert ( town );
townmap.print ( );
```



	A	B	C	D
1		이름	주거지	전화번호
2	1	유연수	홍지동	010-2019-1711
3	2	김연지	홍지동	010-9765-4321
4	3	박영곤	홍지동	010-7123-9090
5	4	심기현	구기동	010-5491-7852
6	5	도일수	구기동	010-2210-9757
7	6	반규원	구기동	010-8852-7195
8	7	십진수	평창동	010-8572-2124
9	8	장거한	구기동	010-5278-1209
10	9	한수연	홍지동	010-2129-3744
11	10	천태백	홍지동	010-4296-4778
12	11	유정인	홍지동	010-3889-7451
13	12	풍상현	홍지동	010-6951-9114
14	13	전나래	홍지동	010-3329-2177
15	14	이기현	구기동	010-1818-1414
16	15	연아린	구기동	010-6541-0921
17	16	명지수	효자동	010-3789-5117
18	17	강용권	평창동	010-9006-2215
19	18	문효정	홍제동	010-5549-7532
20	19	김중선	효자동	010-5225-2521
21	20	단주성	부암동	010-8483-4239
22	21	사지정	효자동	010-7291-9265
23	22	김바름	구기동	010-2318-5821
24	23	주한미	홍은동	010-4299-8796
25	24	김무진	부암동	010-1212-1212

6.3 맵의 구현

- 이진 탐색 트리의 연산
  - Init ( )
  - Search ( )
  - Insert ( )
  - Delete ( )

6.3 맵의 구현

- 필요한 맵의 연산
  - Init (Set)
  - Find
  - Insertion
  - Deletion

6.3 맵의 연산 (1): Init

- Init ( )
  - Empty map의 생성 → root node의 value = 0

```
void node::init ( )
{
    this->key = -1;
    this->value = 0;
    this->lchild = this->rchild = NULL;
}
```

6.3 맵의 연산 (2): Set

- Set ( )
  - node에 지정된 key를 저장하고 value를 1로 설정

```
void node::set ( int skey )
{
    this->key = skey;
    this->value = 1;
    this->lchild = this->rchild = NULL;
}
```

6.3 맵의 연산 (3): Find

- Find ( ) // Get value ( )
  - 내가 찾는 key의 원소가 있는지 탐색
  - 있으면 그 원소의 value를, 없으면 0을 리턴

```
int node::find ( int skey )
{
    if ( skey == this->key )
        return this->value;
    int temp;
    if ( skey > this->key && this->rchild != NULL ) {
        temp = this->rchild->find ( skey );
        if ( temp > 0 )
            return temp;

    }
    if ( skey < this->key && this->lchild != NULL ) {
        temp = this->lchild->find ( skey );
        if ( temp > 0 )
            return temp;
    }
    return 0;
}
```

6.3 맵의 연산 (4): Insert

- Insert ( )
  - 새로운 key를 맵에 추가
  - 해당 key가 존재하면 그 key의 value를 1 증가

```
void node::insert ( int ikey )
{
//  degenerate case?

    if ( ikey == this->key ) {
        this->value++;
        return;
    }
    else if ( ikey > this->key ) {
        if ( this->rchild != NULL )
            this->rchild->insert ( ikey );
        else {
            this->rchild = (node *) malloc ( sizeof(node) );
            this->rchild->set ( ikey );
        }
    }
```

6.3 맵의 연산 (4): Insert

- Insert ( )
  - 새로운 key를 맵에 추가
  - 해당 key가 존재하면 그 key의 value를 1 증가

```
    else {
        if ( this->lchild != NULL )
            this->lchild->insert ( ikey );
        else {
            this->lchild = (node *) malloc ( sizeof(node) );
            this->lchild->set ( ikey );
        }
    }
}
```

6.3 맵의 연산 (5): Delete

- Delete ( )
  - 해당 key를 맵에서 제거
  - 이진 탐색 트리와 같은 과정 (딱 하나 다름!!)

```
int node::delete ( int dkey )
{
//  dkey와 key와 같으면
    if (dkey == this->key) {
        printf("Deleting %d\n", dkey);
        // (1) 두 child가 다 NULL
        // (2-1) lchild만 NULL
        // (2-2) rchild만 NULL
        // (3) 둘 다 NULL이 아닌 경우
    }
//  dkey가 key보다 작으면
    else if (dkey < this->key) {
        this->lchild->delete(dkey);
    }
//  dkey가 key보다 크면
    else {
        this->rchild->delete(dkey);
    }
}
```

6.4 최다 빈도수 단어 찾기

- 텍스트 파일로부터 단어를 사용된 빈도순으로 정리하여 출력함.
  - 텍스트 파일에서 단어를 하나씩 읽어 들임
  - 의미 없는 단어 배제
    - 예: 44th, 2007, 20
  - 문장 부호를 제거하여 처리
    - (born → born
    - presidency. → presidency
    - that, → that

6.4 최다 빈도수 단어 찾기

- 예) 다음의 텍스트 파일에서 다음과 같은 결과를 출력 (최다 빈도수 상위 10개만 출력)

```
Barack Hussein Obama II (born August 4, 1961) is an American politician who
served as the 44th President of the United States from January 20, 2009 to
January 20, 2017. The first African American to assume the presidency, he was
previously the junior United States Senator from Illinois from 2005 to 2008.
Before that, he served in the Illinois State Senate from 1997 until 2004.
```

```
[5] the
[4] from
[3] to
[2] January
[2] Illinois
[2] United
[2] States
[2] American
[2] he
[2] served
```

6.4 최다 빈도수 단어 찾기

- 맵과 히입 (max heap)을 동시에 사용할 것
  - text file의 각 단어들을 맵에 삽입
  - 맵에서 단어를 추출해서 히입에 저장 (value 순으로)
  - 히입에서 n 개의 단어를 추출

```
map tmap;
heap theap;

tmap.init ( );
for each word in the text file
    tmap.insert ( word );

while ( !tmap.is_empty ( ) )
    theap.push ( tmap.pop ( ) );

while ( !theap.is_empty ( ) )
    theap.pop ( );
```

main ( ) 함수

```
int main()
{
    node tmap;
    heap theap;

    FILE *fp = fopen("obama.txt", "r+t");
    char input[64];

    tmap.init ( );
    while (fscanf(fp, "%s", input) != EOF) {
        if (!is_word(input))
            continue;
        tmap.insert(input);
    }
```

main ( ) 함수

```
    theap.init();

    int dcnt;
    hnode temp;
    while (1) {
        strcpy(input, tmap.get_data());
        dcnt = tmap.get_cnt();
        temp.set(dcnt, input);
        theap.push(temp);

        if (tmap.remove(input))
            break;
    }

    int i;
    for (i = 0; i < 10; i++) {
        temp = theap.pop();
        printf("%s (%d)\n", temp.get_data(), temp.get_cnt());
    }

    system("pause");
    return 0;
}
```