자료구조 및 설계
실습

제 출 일	2018. 11. 03
과 제 번 호	06
분 반	01
학 과	컴퓨터공학과
학 번	201702087
이 름	함지희

1. 구현 내용 설명 (method's Algorithm & result)

put(), nextProbe()을 제외하고는 리스팅 9.5, 리스팅 9.6과 동일 (이중 해싱은 hash2()가 추가로 들어감)

(1) 개방주소법

1) **public int** getcollision()

- collision의 값을 리턴하는 함수, collision은 충돌횟수를 세기 위한 변수이다.

2) **private int** nextProbe()

- 선형 조사 :i를 증가시키며 조사

```
private int nextProbe(int h, int i) { //Linear Probing
    return (h + i)%table.length;
}
```

- 제곱 조사 :i*i를 증가시키며 조사

```
private int nextProbe(int h, int i) { //Quadratic Probing
    return (h + i*i)%table.length;
}
```

- 이중 해싱 : 해시 함수를 추가로 작성(hash2)하여 그것과 I를 곱한 것을 증가시키며 조사

```
private int hash2(Object key) {
    if (key == null) throw new IllegalArgumentException();
    return 1 + (key.hashCode() & 0x7FFFFFFF) % (table.length-1);
}

private int nextProbe(int h,int d, int i) { // Double Hashing
    return (h + i * d)%table.length;
}
```

3) put()

(이중 해싱은 해시함수가 2개이므로 추가로 int d = hash2()가 있고 , nextProbe에서 3개의 인수를 받는다. 그 외는 같다.)

```
public Object put(Object key, Object value) {
    if(used > loadFactor*table.length) rehash();
    int h = hash(key);
    for(int i = 0; i < table.length;i++) {
        int j = nextProbe(h,i);
        Entry entry = table[j];
        if(entry == null) {
            table[j] = new Entry(key,value);
            ++size;
            ++used;
            return null; // success
        }
        if(entry == NIL) {collision++; continue;}
        if(entry.key.equals(key)) {
            Object oldValue = entry.value;
            int v = (int)table[j].value; // value가 Object형이라서 int형으로 형변환
            table[j].value = v+1; // value에 +1을 해준다
            return oldValue;
        }
        collision++;
    }
    return null; //table overflow
}
```

1. 사이즈가 일정 이상이면 재해싱을 해준다. 해시함수로 해시를 받아 h에 저장
2. 테이블 크기만큼 반복하는 반복문에 넣고 받은 해시를 nextProbe(선형조사, 제곱조사 , 이중해싱)에 넣어 나온 값을 j에 넣고 그걸 인덱스로 받음

3. 해당 인덱스의 값이 비어 있으면 받은 인수들로 새로운 Entry를 만들고 그걸 그 인덱스의 값으로 넣어준다.
4. 해당 인덱스의 값이 NIL이면 건너뛰고 반복문을 계속 한다.
5. 해당 인덱스의 키 값과 입력받은 키가 같으면 업데이트를 해주어야 한다. j인덱스에 있는 값의 value을 int로 형변환 해서 v에 저장 후, 다시 j인덱스에 있는 value값을 v+1로 바꾸어준다.

(2) 폐쇄 주소법

1) put()

```
public Object put(Object key, Object value) {
    int h = hash(key);
    for(Entry e = table[h]; e!= null; e = e.next) {
        if(e.key.equals(key)) {
            Object oldValue = e.value;
            int v = (int)e.value;
            e.value = v+1;
            return oldValue;
        }
    }
    table[h] = new Entry(key,value,table[h]);
    ++size;
    if(size > loadFactor*table.length) rehash();
    return null;
}
```

1. 해시함수에 key를 넣어 해시를 받고 h에 넣는다.
2. h를 인덱스로 넣고 , next로 이동하면서 null이 될 때까지 반복하는 반복문으로 key값이 같은지 검사한다. key값이 같으면 value값을 업데이트 해줘야 한다. e의 value값을 int로 형변환해서 v에 따로 저장한 후 다시 e의 value값에 v+1을 넣어주어 업데이트 해준다.
3. 그 외 경우엔 해당 인덱스에 새로운 entry를 넣어주는데 next값에 현재 인덱스의 값을 넣어준다. 사이즈가 일정이상 이면 재해싱을 해준다.

2. 실행 결과 화면 (captured image)