[제 12 주 실습]

동적 해싱

강 지 훈
jhkang@cnu.ac.kr

충남대학교 컴퓨터공학과

[문제 12]
동적 해싱

Motivation

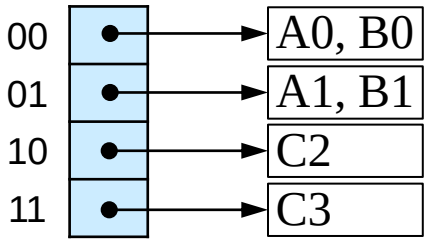
- Hash table에 삽입된 원소가 너무 많아지면?
  - Collision 증가
- 해결책은?
  - Hash Table의 크기를 동적으로 증가시키자!
- 크기를 동적으로 늘릴 때의 문제점은?
  - 모든 원소를 다시 hashing하여 삽입해야 하는가?

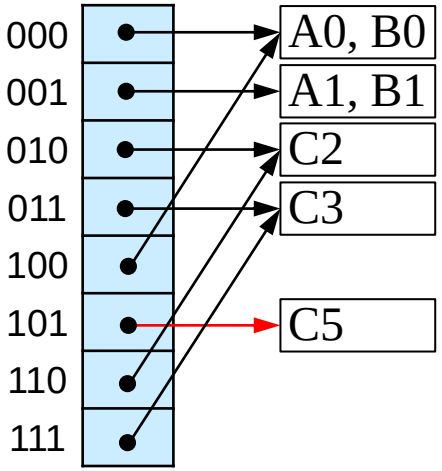
문제 개요

- 기존의 Hash Table이 큰 데이터를 이용 할 경우 생기는 문제점을 파악하고 이를 해결 하기 위한 방안으로 동적 해싱 구현한다.

출력화면

```
<<프로그램을 시작합니다>>
삽입할 값을 입력하세요 : A0
[0]  - A0 |
삽입할 값을 입력하세요 : B0
[0]  - A0 | B0 |
삽입할 값을 입력하세요 : A1
[0]  - A0 | B0 |
[1]  - A1 |
삽입할 값을 입력하세요 : B1
[0]  - A0 | B0 |
[1]  - A1 | B1 |
삽입할 값을 입력하세요 : C2
[0]  - A0 | B0 |
[1]  - A1 | B1 |
[2]  - C2 |
삽입할 값을 입력하세요 : C3
[0]  - A0 | B0 |
[1]  - A1 | B1 |
[2]  - C2 |
[3]  - C3 |
삽입할 값을 입력하세요 : C5
[0]  - A0 | B0 |
[1]  - A1 | B1 |
[2]  - C2 |
[3]  - C3 |
[4]  - A0 | B0 |
[6]  - C5 |
[7]  - C2 |
[8]  - C3 |
삽입할 값을 입력하세요 : -1
<<프로그램을 종료합니다>>
```





PAGE_SIZE = 2

테스트 진행

이 과제에서 필요한 객체는?

- AppView
- AppController
  - DynamicHash
- DynamicHash
  - Page
- Page
  - BucketRecord
- BucketRecord

AppControllor의 공개 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public void run()

AppView의 공개 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public AppView()
  - public void showInputSelectMenu()
  - public void showErrorMsg()
  - public void showResult()
  - public int inputInt()

DynamicHash의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public DynamicHash()
  - public int size(Page aPtr)
  - public boolean insert(BucketRecord aRecord, String aKey)
  - public boolean delete(BucketRecord aRecord, String aKey)
  - public int find(String aKey)

Page의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public Page(int givenNumOfIdents)
  - public int localDepth()
  - public int numOfIdents()
  - public boolean insert(BucketRecord aRecord)
  - public BucketRecord delete(BucketRecord aRecord)
  - public PageIterator pageIterator()
  - public class PageIterator

BucketRecord의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public BucketRecord(String givenKey)
  - public void setKey(String aKey)
  - public String key()

Class “AppControllor“

AppControllor– 비공개 인스턴스 변수

```
public class AppControllor {
    private AppView _appView;
    private DynamicHash _dynamicHash;
```

AppControllor 의 공개 함수 run()의 구현

```
public void run() {
    _dynamicHash = new DynamicHash();
    _appView = new AppView();

    _appView.outputMsg(_appView.MSG_StartProgram);

    _appView.outputMsg(_appView.MSG_RequestInput);
    String input = _appView.inputString();
    while(!input.endsWith("-1"))
    {
        BucketRecord aRecord = new BucketRecord(input);
        _dynamicHash.insert(aRecord, input);

        _appView.outputMsg(_appView.MSG_RequestInput);
        input = _appView.inputString();
    }
    _appView.outputMsg(_appView.MSG_EndProgram);
}
```

Class “AppView“

AppView – 비공개 인스턴스 변수

```
import java.util.Scanner;

public class AppView {
    private Scanner _scanner;
```

AppView의 Public Method

- AppView 의 Public Member의 선언
  - public String MSG_StartProgram = "<<프로그램을 시작합니다>>₩n";
  - public String MSG_EndProgram = "<<프로그램을 종료합니다>>₩n";
  - public String MSG_RequestInput = "삽입할 값을 입력하세요 : ";
  - public String MSG_Error = "Error: 오류가 발생하였습니다. ₩n";

AppView의 Public Method

- AppView 의 Public Member function의 사용법과 구현
  - public AppView()
    - 생성자
  - public void outputMsg(String aString)
    - aString을 출력
  - public String inputString()
    - String을 하나 입력 받아 반환
  - public int inputInt()
    - Int를 하나 입력 받아 반환

Class “DynamicHash”

h(k,p)

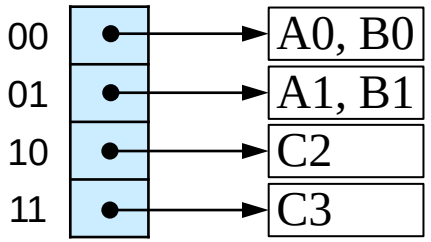
- h(k): hash function
  - The range of h is sufficiently large
- h(k,p): the least significant p bits of h(k)

- Example:
  - h(A0,2) = 00 = 0
  - h(A0,3) = 000 = 0
  - h(C5,2) = 01 = 1
  - h(C5,3) = 101 = 5

k	h(k)
A0	100 000
A1	100 001
B0	101 000
B1	101 001
C1	110 001
C2	110 010
C3	110 011
C5	110 101

Directory

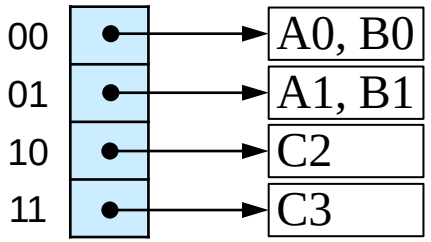
- Pointers to buckets
- Directory Size for h(k,p) : 2^p
  - 2^2 = 4 if h(k,2) is used
  - 2^5 = 32 if h(k,5) is used
- directory depth
  - the number of bits of h(k) used to index the directory
- Example:
  - Directory Depth: 2
    - So, the directory size: 2^4 = 4
  - 2 slots per bucket



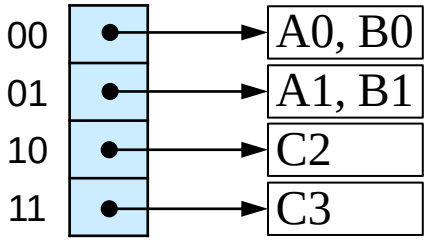
How to search for a key k?

- Merely examine the bucket pointed to by the directory entry d[h(k,p)] when the current directory depth is p.

- Example: Search for a key B1:
  - h(B1,2) = 01
  - Examine the bucket pointed to by d[01] .
  - Found.

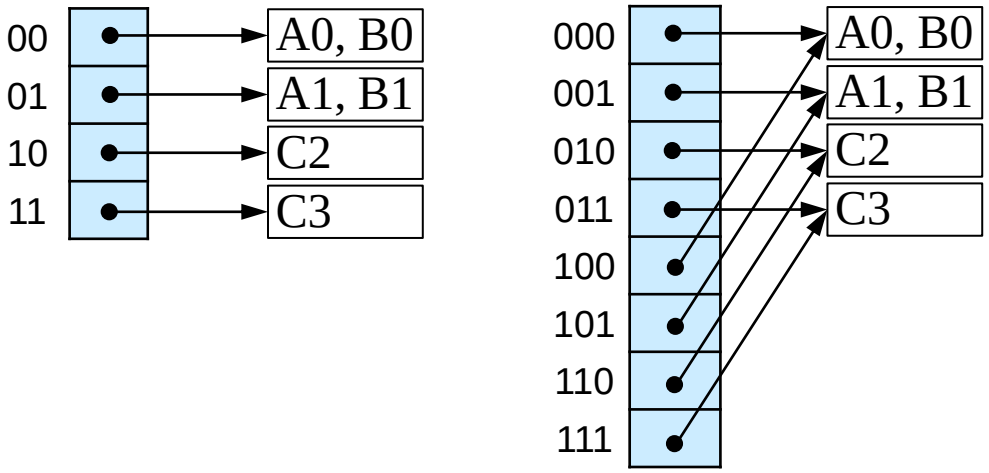


Example 1: Insert C5 [1]



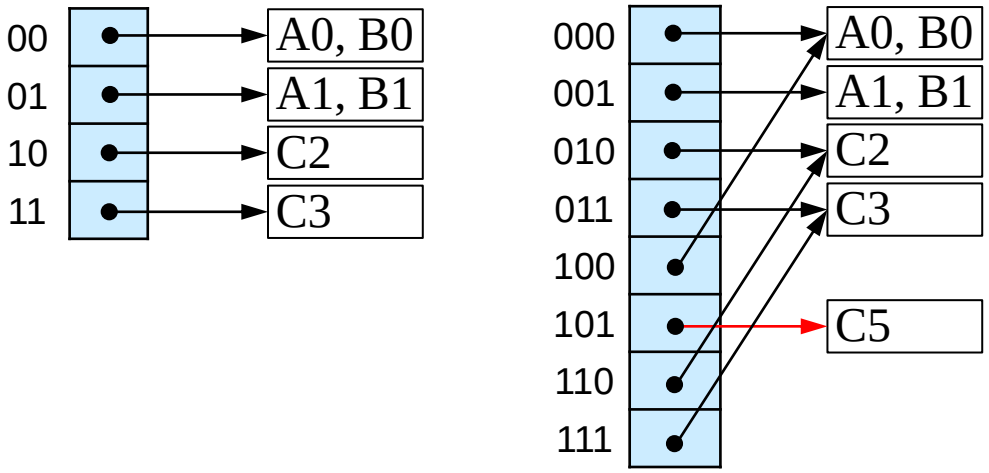
- directory depth: 2
- h(C5,2) = 01
- Examine directory entry d[01] (= d[h(C5,2)])
  - Overflow
- Determine the least u such that h(k,u) is not the same for any keys in the overflow bucket
  - h(A1) = 100001, h(B1) = 101001, & h(C5) = 110101
    So, u = 3

Example 1: Insert C5 [2]



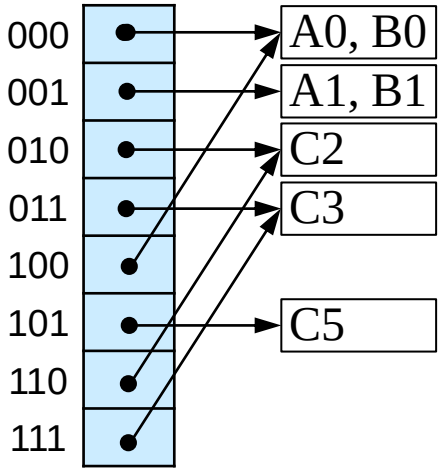
- Double the directory since u is greater than the current directory depth 2.
- Copy the pointers to the buckets so that the pointer in each half of the directory are the same.

Example1: Insert C5 [3]



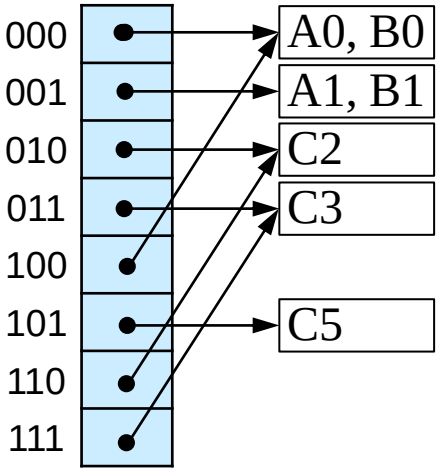
- Split the overflow bucket using h(k,3):
  - h(A1,3) = 001, h(B1,3) = 001, h(C5,3) = 101
  - Create a new bucket with C5 and place a pointer to this bucket in d[101].
- Notice: d[100] points the bucket for A0 and B0 although h(A0,3) = h(B0,3) != 000 ! Why?

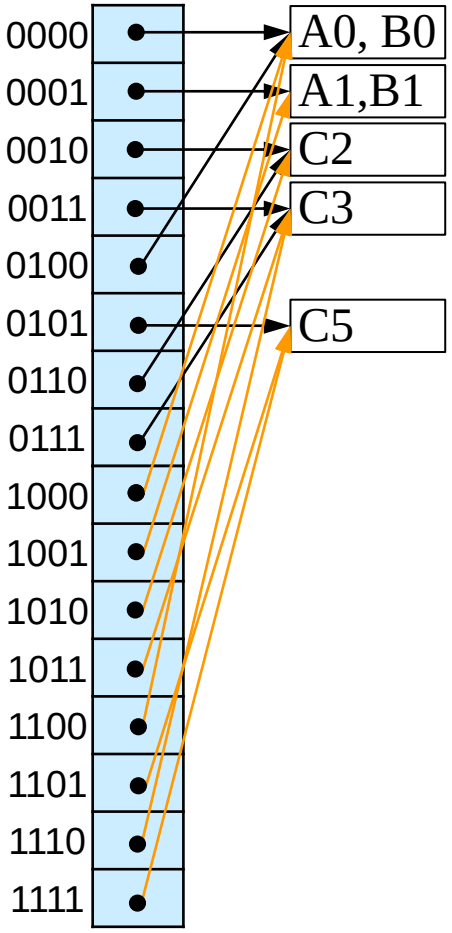
Example 2: Insert C1 [1]



- h(C1,3) = 001
- Examine the bucket pointed to by d[001].
  - Overflow
- Determine the least u:
  - h(A1) = 100001, h(B1) = 101001, h(C1) = 110001
    So, u = 4

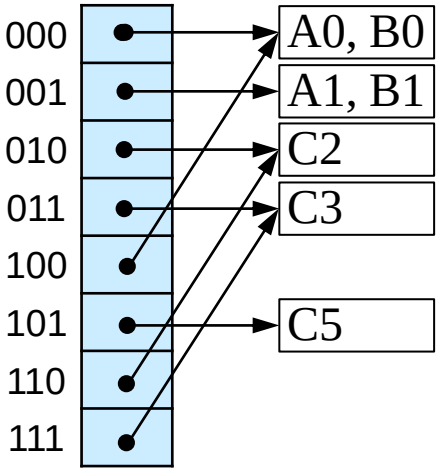
Example 2: Insert C1 [2]

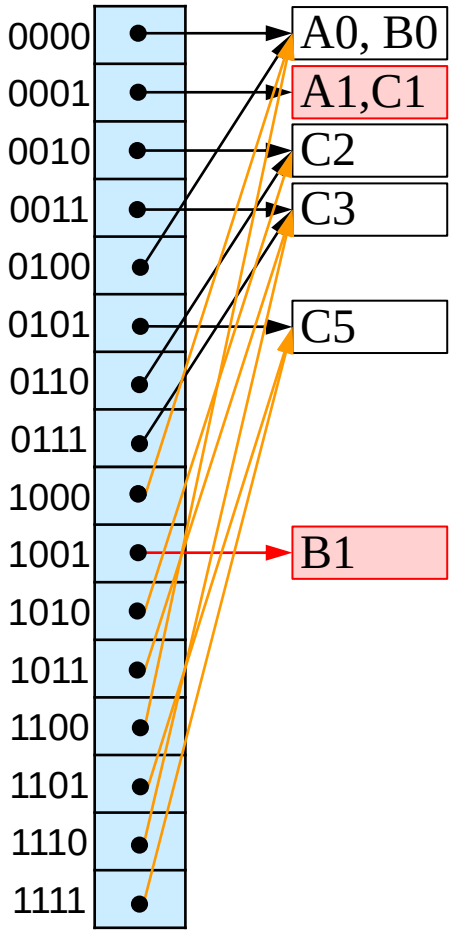




- Double the directory whit the size 16.
- Copy the pointers.

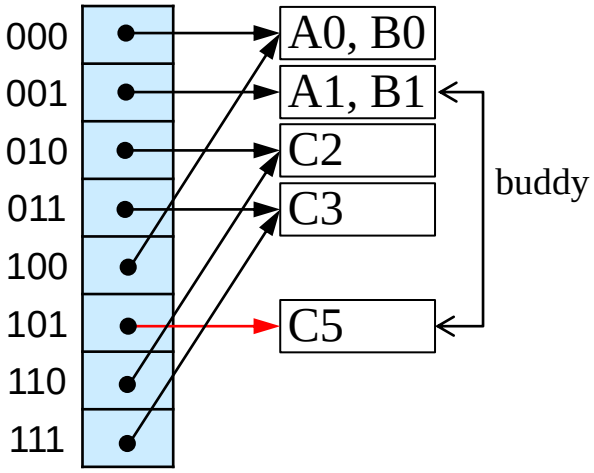
Example 2: Insert C1 [3]

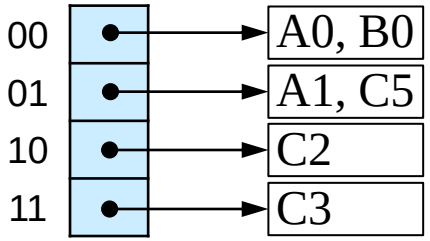




- Split the overflow bucket using h(k,4):
  - h(A1,4) = 0001, h(B1,4) = 1001, & h(C1,3) = 0001
  - Remove B1 from the bucket.
  - Create a new bucket with B1 and place a pointer to this bucket in d[1001].
  - Insert C1 into the bucket pointed to by d[0001].

Example 3: Delete B1 [1]





- B1를 삭제 할 경우
  - 001의 개수가 1개, Buddy인 101의 개수가 1개이므로 Page는 하나로 합친다(coalesce)
  - 이때 다른 directory들을 비교하여 같은 depth의 다른 page들에 더 이상 Page가 없는 경우, depth를 줄인다.

비공개 인스턴스 변수

```
public class DynamicHash {
    private static final int PAGE_SIZE = 10;
    private static final int DIRECTORY_SIZE = 32;

    private int _global_depth; // 디렉토리의 높이
    private Page [] _directory; // 페이지를 가리키는 포인터
```

Public Method

- DynamicHash의 Public Member function의 사용법
  - public DynamicHash()
    - 생성자
  - public int size(int aPageIndex)
    - 해당 페이지 내에 있는 식별자의 개수를 반환
  - public boolean insert(BucketRecord aRecord, String aKey)
    - 입력 받은 record를 aKey에 맞게 삽입
  - public boolean delete(BucketRecord aRecord, String aKey)
    - 해당 파일에서 aRecord를 찾아서 삭제
  - public int find(String aKey)
    - 키가 있는지 확인

Private Method

- DynamicHash의 Private Member function의 사용법
  - private String hash(String aKey, int precision)
    - uniform hashing function을 사용하여 key를 hashing하면 precision개의 하위 비트가 페이지로 반환
  - private boolean pageSearch(String aKey, Page aPage)
    - 어떤 key를 가지고 있는 페이지를 탐색해서 발견 시 true를 반환
  - private int convert(String anIndex)
    - 페이지를 가리키는 index를 동일한 값의 정수로 변환

Private Method

- DynamicHash의 Private Member function의 사용법
  - private int buddy(Page anPage)
    - 한 페이지를 입력 받아 이 페이지의 buddy를 반환 선행 비트가 보수된 값을 구함
  - private void coalesce(int aPageIndex, int aBuddyIndex)
    - 페이지와 그 페이지의 버디를 하나의 페이지로 통합

Public Method

- DynamicHash의 Public Member function의 구현
  - public DynamicHash()
    - _directory를 DIRECTORY_SIZE만큼 초기화
    - _global_depth를 초기화
  - public int size(int aPageIndex)
    - aPageIndex의 numOfIdents를 반환

Public Method

DynamicHash의 Public Member function의 구현

```
public boolean insert(BucketRecord aRecord, String aKey)
{
    int pageIndex = find(aKey);
    Page page = _directory[pageIndex];
    if (pageSearch(aKey, page) == true) {
        return false;
    } else {
        if(page == null)
            page = new Page(PAGE_SIZE);

        if (page.numOfIdents() != PAGE_SIZE) {
            page.insert(aRecord);
        } else {
            // 해당 페이지를 두 페이지로 나누고 새로운 키를 삽입한 다음
            // 필요하면 global_depth의 값을 갱신
            // 만약 global_depth의 값이 WORD_SIZE보다 크면 false를 반환
            // 기존의 _directory에 연결 되어 있는 page들의 링크를 정리
        }
        showAllDirectory();
        return true;
    }
}
```

Public Method

- DynamicHash의 Public Member function의 구현

```
public boolean delete(BucketRecord aRecord, String aKey)
{
    int pageIndex = find(aKey);
    Page page = _directory[pageIndex];
    if (pageSearch(aKey, page) == false) {
        return false;
    }

    page.delete(aRecord);

    if ( size(pageIndex) + size(buddy(page)) <= PAGE_SIZE ) {
        coalesce(pageIndex, buddy(page));
    }

    return true;
}
```

Public Method

- DynamicHash의 Public Member function의 구현

```
public int find(String aKey)
{
    String index = hash(aKey, _global_depth);
    int intindex = convert(index);
    return intindex;
}
```

Public Method

- DynamicHash의 Public Member function의 구현

```
public void showAllDirectory()
{
    for(int i = 0; i < DIRECTORY_SIZE; i++)
    {
        Page currentPage = _directory[i];
        if( currentPage != null)
        {
            Page.PageIterator pageIterator = currentPage.pageIterator();
            System.out.print("[" + i + "]  - ");
            while(pageIterator.hasNext())
            {
                System.out.print(pageIterator.next().key()  + " | ");
            }
            System.out.println();
        }
    }
}
```

디버깅용 함수

Private Method

- DynamicHash의 Private Member function의 구현
  - private String hash(String aKey, int precision)
    - 1. h(k,p)의 방식을 따라 구현 – hash Table을 만들어서 구현
    - 2. 기존의 hash 함수를 이진수로 변환하여 구현하여도 됨
  - private boolean pageSearch(String aKey, Page aPage)
    - 어떤 key를 가지고 있는 aPage를 PageIterator를 이용하여 탐색해서 발견을 할 경우 page를 반환
  - private int convert(String anIndex)
    - h(k,p)를 참고하여 convert > How? 이진수 변형은 어떻게?

Private Method

- DynamicHash의 Private Member function의 구현
  - private int buddy(Page anPage)
    - 하위 비트가 같은 anPage의 buddy를 확인
  - private void coalesce(int aPageIndex, int aBuddyIndex)
    - aPageIndex에 있는 page에 aBuddyIndex에 있는 page의 값을 PageIterator를 이용하여 하나씩 얻어와서 삽입
    - aBuddyIndex의 page를 초기화
    - 현재 _global_depth에 있는 모든 page가 비어 있을 경우 _global_depth를 하나 줄인다.

Class “Page”

비공개 인스턴스 변수

```
public class Page {
    private int _localDepth; // page level
    private BucketRecord [] _record;
    private int _numOfIdents;
    private int _maxPagesize;
```

Public Method

- Page 의 Public Member function의 구현
  - public Page(int givenNumOfIdents)
    - _record를 givenNumOfIdents만큼 생성
    - _numOfIdents를 초기화
    - _maxPagesize에 givenNumOfIdents를 저장
  - public int localDepth()
    - _localDepth을 반환
  - public int numOfIdents()
    - _numOfIdents을 반환

Public Method

- Page 의 Public Member function의 구현
  - public boolean insert(BucketRecord aRecord)
    - 현재 삽입 Size를 _maxPagesize와 비교하여 가득 차 있으면 false를 반환
    - aRecord를 _record에 삽입
  - public BucketRecord delete(BucketRecord aRecord)
    - _record에서 aRecord를 찾음
    - 찾은 위치에서부터 하나씩 앞으로 이동하여 aRecord를 삭제
    - _numOfIdents를 하나 감소

Public Method

- Page 의 Public Member function의 구현
  - public PageIterator pageIterator()
  - public class PageIterator
    - private int _nextIndex;
    - private PageIterator()
    - public boolean hasNext()
    - public BucketRecord next()

Class “BucketRecord”

비공개 인스턴스 변수

```
public class BucketRecord {
    private String _key;
```

Public Method

- Page 의 Public Member function의 구현
  - public BucketRecord(String givenKey)
    - _key에 givenKey를 저장
  - public void setKey(String aKey)
  - public String key()

[문제 12] 요약

- 동적 해싱과 정적해싱의 차이는 무엇인가?
  - 구현한 동적 해싱의 과정을 확인하시오.

- 각 요약에 대한 내용을 보고서에 작성하여 제출하세요.

과제 제출

과제 제출

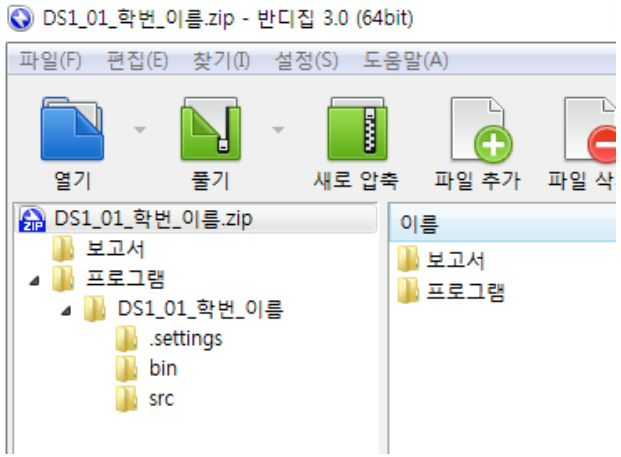
- pineai@cnu.ac.kr
  - 메일 제목 : [0X]DS2_12_학번_이름
    - 양식에 맞지 않는 메일 제목은 미제출로 간주됨
    - 앞의 0X는 분반명 (오전10시 : 00반 / 오후4시 : 01반)

- 제출 기한
  - 11월 26일(화) 23시59분까지
  - 시간 내 제출 엄수

  - 제출을 하지 않을 경우 0점 처리하고, 숙제를 50% 이상 제출하지 않으면 F 학점 처리하며, 2번 이상 제출하지 않으면 A 학점을 받을 수 없다.

과제 제출

- 파일 이름 작명 방법
  - DS2_12_학번_이름.zip
  - 폴더의 구성
    - DS2_12_학번_이름
      - 프로그램
        - 프로젝트 폴더 / 소스
        - 메인 클래스 이름 : DS2_12_학번_이름.java
      - 보고서
        - 이곳에 보고서 문서 파일을 저장한다.
        - 입력과 실행 결과는 화면 image로 문서에 포함시킨다.
        - 문서는 pdf 파일로 만들어 제출한다.



보고서 작성 방법

- 겉장
  - 제목: 자료구조 실습 보고서
  - [제xx주] 숙제명
  - 제출일
  - 학번/이름
- 내용
  1. 프로그램 설명서
     1. 주요 알고리즘 /자료구조 /기타
     2. 함수 설명서
     3. 종합 설명서 : 프로그램 사용방법 등을 기술
  2. 구현 후 느낀 점 : 요약의 내용을 포함하여 작성한다.
  3. 실행 결과 분석
     1. 입력과 출력 (화면 capture : 실습예시와 다른 예제로 할 것)
     2. 결과 분석

     -------- 표지 제외 3장 이내 작성 --------
  4. 소스코드 : 화면 capture가 아닌 소스를 붙여넣을 것

     소스는 장수 제한이 없음.

[제 12 주 실습] 끝