Project #2

B+tree Practice

B+tree Practice

- B+tree (originated http://www.amittai.com/prose/bpt.c)
- 제공되는 **bpt.zip** 파일은 HYU LMS에서 다운받을 수 있습니다

1. bpt.zip 파일을 압축해제하면 아래와 같습니다
   - bpt/include/
   - bpt/include/bpt.h
   - bpt/lib/
   - bpt/src/
   - bpt/src/bpt.c
   - bpt/src/main.c
   - bpt/Makefile

B+tree Practice

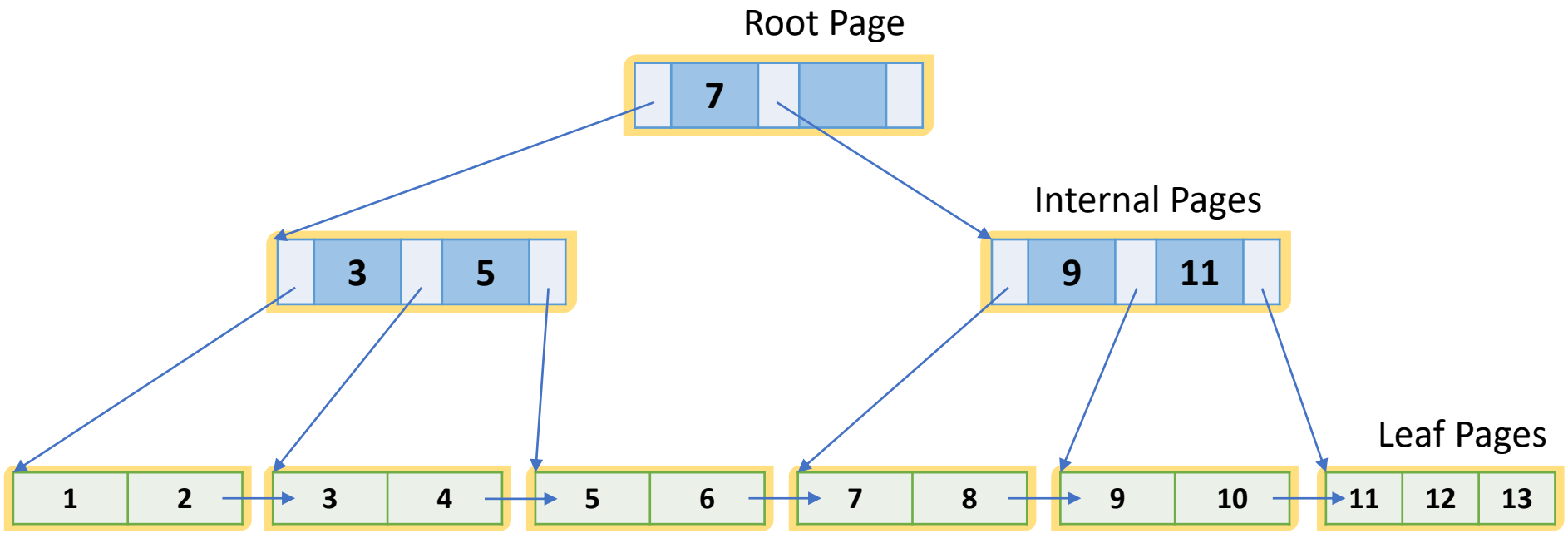
2. Makefile을 이용하여 compile하세요
   - 만약 make가 안된다면 'sudo apt-get install make' 를 했는지 확인하세요
   - Makefile에 다른 source file을 임의로 추가하거나 수정하지 마세요
   - Your_git_repo/assignment2/lib/libbpt.a 파일을 꼭 확인 후 제출하세요
     - 과제 제출시 lib 디렉토리에 libbpt.a 파일이 없을 경우 0점 처리됩니다

3. 제출할 때 프로젝트는 아래와 같은 계층구조로 이루어져 있어야 합니다
   - Your_git_repo
     - assignment2
       - include/
       - lib/
       - src/
       - Makefile

Disk-based B+tree

- 현재 제공 된 B+tree코드는 On-disk 형식입니다
  - B+tree의 동작 과정을 가시적으로 볼 수 있는 사이트를 소개합니다
  - https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html
- 이번 프로젝트의 목표는 아래와 같습니다

  보조 연산(ex. Merge, Split)에 대한 overhead를 줄이기 위한 전략 구현



B+tree Overhead

- 실습 수업 시간에 B+tree의 보조 연산에 대한 비용 부담(overhead)에 대한 내용을 다룬 바 있습니다
  - (Split과 Merge와 같은 보조 연산은 Disk I/O를 발생시켜 성능 하락의 원인이 된다는 내용)

- 따라서 이를 줄이기 위한 전략을 함께 구현하여 최종 disk-based B+tree code를 제출하시면 됩니다
  - 'Delay-merge', 'Key-rotation' 등 자유
  - 어떤 방법을 채택하였고, 어떻게 구현했으며, 왜 자신의 성능개선 전략이 타당한지 wiki에 상세히 기술되어 있어야 합니다

왠만하면binary search같은거 말고 노드에 관한것 하기 흐름도 설명

위키 사용법:

notion 사용
node 구조체
(코드)

함수별로 설명

개발환경
draw IO 등 그림 그려서 설명
다른 사람이 이해할수있게 하기

Judging System

- 이번 프로젝트의 채점 환경(**Default Version**)은 아래와 같습니다

```
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
GNU Make 4.2.1
```

❖ Version차이로 인하여 생기는 문제는 조교가 책임지지 않으며 **빌드와 실행이 안 될 경우 점수는 0점 입니다**

Submit

❖ Code

- **Completeness** : 명세의 요구 조건을 모두 올바르게 구현해야 합니다
- **Defensiveness** : 발생할 수 있는 예외 상황에 대처할 수 있어야 합니다
- **Comment** : 코드에는 반드시 주석이 있어야 합니다

➢ Submission via HYU GitLab "git push"

❖ Wiki

- **Design** : 명세에서 요구하는 조건에 대해서 어떻게 구현할 계획인지, 어떤 자료구조와 알고리즘이 필요한지, 수업시간에 배운 이론이 어떻게 적용되어야 하는지 등 자신만의 디자인을 서술합니다
- **Implement** : 코드를 그대로 복사하여 문서에 붙여놓지 마시고 본인이 새롭게 구현하거나 수정한 부분에 대해서 무엇이 기존과 다른지, 해당 코드가 무엇을 목적으로 하는지에 대한 설명을 구체적으로 서술합니다
- **Result** : 해당 명세에서 요구한 부분이 정상적으로 동작하는 실행 결과를 첨부하고, 이에 대한 동작 과정에 대해 설명합니다
- **Trouble shooting** : 과제를 수행하면서 마주하였던 문제와 이에 대한 해결 과정을 서술합니다. 혹여 문제를 해결하지 못하였다면 어떤 문제였고 어떻게 해결하려 하였는지에 대해서 서술합니다

➢ Submission via HYU LMS

Submit

❖ Milestone1

- 제공 드린 On-disk B+tree code를 분석하여 report(wiki) 제출
- report(wiki)는 아래의 내용을 포함해야 합니다
  1. Detail flow of the structure modification (insert, delete, split, merge)
  2. (Naïve) designs or required changes for reducing overhead in B+tree
- Deadline : ~11/20 11:59 pm

❖ Milestone2

- 보조 연산 overhead에 대한 성능개선이 담긴 최종 On-disk B+tree code 제출
- 자신의 디자인을 포함한 report(wiki) 제출
- Deadline : ~11/29 11:59 pm