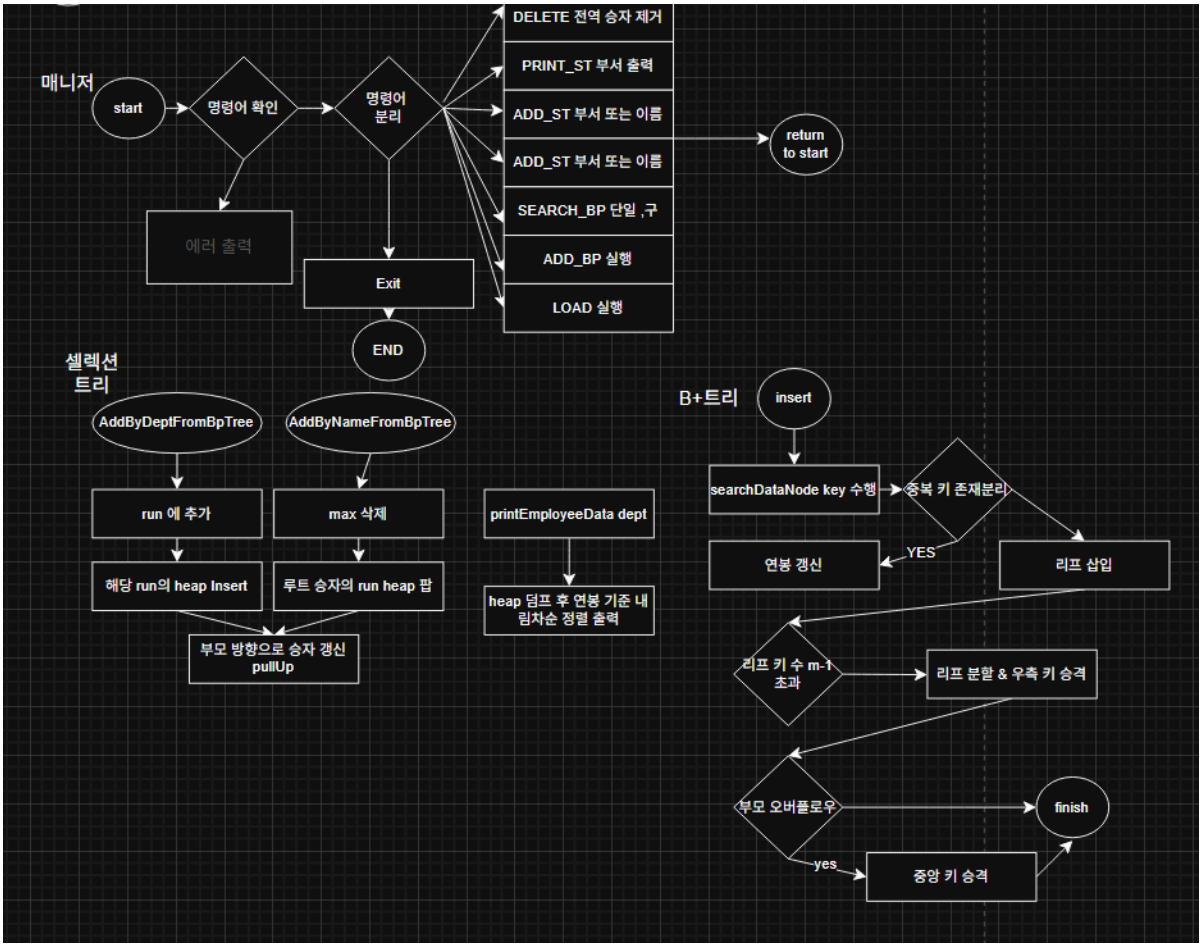
데이터 구조 설계/ 실습

2025_Kwangwoon_DS_Project_2
2021202003 강준우

Introduction

해당 프로젝트에서 직원 관리를 수행하는 프로그램을 제작한다. 자료구조는 b+ tree와 selection tree, heap 구조를 기반으로 사용하며 순서대로 직원 관리와 연봉 순위 연봉 기반 max heap 구조를 수행한다. 각각의 구조를 통해서 우리는 직원들의 이름과 부서 사번 연봉에 대한 내용을 저장 및 관리한다. 또한 연봉을 기반으로 나열하여 추가 관리를 가능하게 한다.

Flowchart



Algorithm

BpTree
차수 m의 B+트리 사용, 실제 레코드는 리프 노드의 정렬 맵에만 저장함
인덱스 노드는 분기 키와 mostLeftChild만 가짐
리프 노드는 prev·next로 연결되어 범위와 전체 순회가 선형으로 동작함
탐색은 루트에서 시작해 인덱스 맵의 upper_bound로 마지막으로 name 이하인 간선을 선택한 뒤 리프에서 맵 조회로 확정함
삽입은 리프를 찾아 중복이면 연봉만 갱신하고 신규면 삽입 후 키 개수가 m-1을 넘으면 리프를 좌우로 분할하고 오른쪽 리프의 첫 키를 부모로 올리며 필요 시 새 루트를 생성함
인덱스 분할은 가운데 키를 부모로 올리고 그 키의 오른쪽 첫 자식을 새 오른쪽 인덱스의 mostLeftChild로 두며 승격 키 이후 항목을 모두 오른쪽으로 이동함
범위 검색은 start가 들어갈 리프에서 lower_bound로 시작해 end를 넘기 전까지 출력하고 리프 체인을 따라 계속 진행함
전체 출력은 가장 왼쪽 리프부터 리프 체인을 따라 모든 항목을 오름차순으로 출력함 왼쪽 서브트리는 키 이하 오른쪽은 키 초과 실제 데이터는 리프에만 존재 리프 체인은 항상 일관 유지 각 노드는 키 개수가 m-1을 초과하지 않음

EmployeeHeap
연봉 내림차순 동률이면 이름 오름차순을 따르는 최대 힙을 1부터 시작하는 배열로 구현함
삽입은 배열 끝에 원소를 넣고 UpHeap으로 부모와 비교해 우선이면 교환하며 루트까지 거슬러 올라감
Top은 비어 있으면 널 포인터 비어 있지 않으면 인덱스 1의 원소를 반환함
삭제는 마지막 원소를 루트로 올린 뒤 DownHeap으로 두 자식 중 우선이 높은 쪽과 교환하며 내려가 구조를 복구함
IsEmpty는 원소 수가 0인지로 판단함
ResizeArray는 용량이 찼을 때 두 배 크기의 새 배열을 만들고 1 기반 인덱싱을 유지한 채 복사해 확장함
비교 규칙은 연봉이 크면 우선 연봉이 같으면 이름이 사전상 앞서면 우선임
시간 복잡도는 삽입과 삭제가 log N Top과 IsEmpty는 O(1)이며 1 기반 인덱싱에서 부모는 i/2 자식은 2i와 2i+1로 계산함

Manager
명령 파일을 열어 전체 크기와 한 줄 길이와 비ASCII 여부를 점검하고 앞뒤 공백을 정리한 뒤 첫 토큰을 대문자 명령으로 엄격 확인하고 나머지를 꼬리 인자로 저장하며 꼬리의 토큰 개수와 길이에 상한을 적용함
꼬리는 레거시 char 버퍼에도 복사해 내부 인터페이스와 호환함
LOAD는 직원 파일을 바이너리로 열어 파일 크기와 라인 길이를 제한하고 각 라인을 토크나이즈해 이름은 소문자 최대 10자 정수는 부호 없는 십진수로 엄격 파싱하며 부서 코드는 100부터 800까지만 허용해 유효할 때만 레코드를 만들어 B+트리에 삽입하고 성공 로그를 남김
ADD_BP는 꼬리 인자 네 개를 검증한 뒤 존재 이름이면 연봉 갱신 없으면 삽입하고 단일 레코드를 성공 포맷으로 기록함
SEARCH_BP는 인자 한 개면 단일 검색 두 개면 구간 검색을 수행하며 결과가 없거나 형식 오류면 300 에러를 기록함

PRINT_BP는 트리 전체를 이름 오름차순으로 출력하고 데이터가 없으면 400 에러를 기록함
ADD_ST는 부서 번호 또는 이름 방식으로 추가하며 부서 번호는 화이트리스트와 정수 검증을 통과해야 하고 이름도 소문자 규칙을 통과해야 하며 성공 로그를 남김
PRINT_ST는 부서 하나를 받아 해당 런을 이름 오름차순으로 출력하고 없으면 600 에러를 기록함
DELETE는 선택 트리의 전역 승자를 제거하고 재정렬하며 실패 시 700 에러를 기록함
EXIT는 꼬리 인자가 비어 있으면 성공 로그를 남기고 종료하며 알 수 없는 명령 대문자 규칙 위반 꼬리 검증 실패는 명령별 지정 에러 코드로 응답함

SelectionTree
부서 코드를 런으로 매핑한 최대 우승자 트리 구조이며 각 런은 연봉 우선 동률 시 연봉 기준 내림차순 힙을 가짐
setTree는 완전 이진 구조의 내부 노드와 8개의 리프를 만들고 리프마다 힙을 초기화함
Insert는 새 직원의 부서 런을 찾아 힙에 넣은 뒤 해당 리프에서 부모 방향으로 올라가며 좌우 자식 대표를 비교해 더 좋은 쪽으로 노드 대표를 갱신함
Delete는 루트 대표를 확인해 해당 부서 런의 힙에서 삭제하고 새 Top을 리프 대표로 설정한 뒤 동일하게 상향 갱신함
printEmployeeData는 요청된 부서 런의 힙을 벡터로 덤프해 연봉 기준 내림차순 정렬 출력하고 런이 비었거나 부서가 유효하지 않으면 600 에러를 기록함
AddByDeptFromBpTree는 B+트리의 가장 왼쪽 리프부터 리프 체인을 순회하며 해당 부서 레코드만 선택 트리에 삽입하고 AddByNameFromBpTree는 이름으로 리프를 찾아 정확히 일치하는 항목만 삽입함
승자 비교는 연봉이 크면 우선 연봉이 같으면 이름이 사전상 앞서면 우선임

Result Screen

```
1   LOAD
2   ADD_BP  luis    100 230079  5000
3   ADD_BP  alex    200 210038  5900
4   ADD_BP  ryan    300 220094  8200
5   ADD_BP  steven  400 170027  9700
6   SEARCH_BP   alex
7   SEARCH_BP   c   g
8   PRINT_BP
9   ADD_ST  dept_no 100
10  ADD_ST  name    steven
11  ADD_ST  name    ryan
12  ADD_ST  name    alex
13  PRINT_ST    100
14  DELETE
15  PRINT_ST    100
16  EXIT
```

```
1   lionel  100 250001  8000
2   bob 100 240011  5900
3   alice   300 220005  1000
4   mohammed    400 190311  7600
5   florian 200 200719  1200
6   eric    100 250011  4000
7   cristiano   100 220058  9900
```

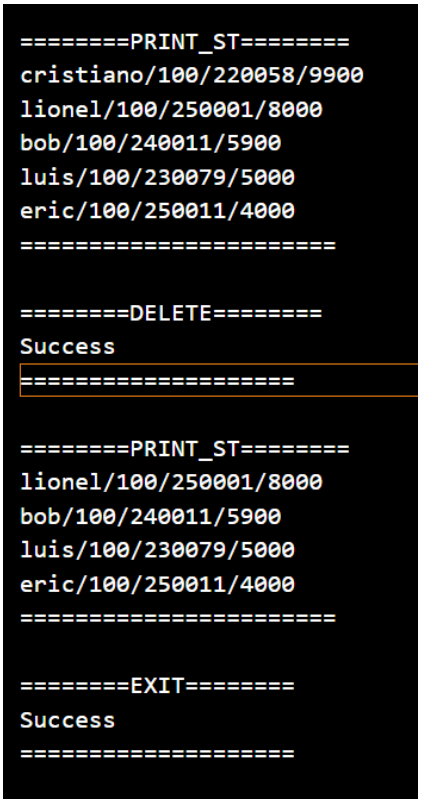
해당 명령어와 직원 데이터를 기반으로 수행했습니다.

```
========LOAD========
Success
====================

========ADD_BP========
luis/100/230079/5000
====================

========ADD_BP========
alex/200/210038/5900
====================

========ADD_BP========
ryan/300/220094/8200
====================

========ADD_BP========
steven/400/170027/9700
====================

========SEARCH_BP========
alex/200/210038/5900
========================

========SEARCH_BP========
cristiano/100/220058/9900
eric/100/250011/4000
florian/200/200719/1200
========================
```

텍스트 파일을 읽어 트리에 이름 오름차순으로 입력합니다. 올바른 직원 데이터가 존재해 성공을 출력한다.
4개 정보를 모두 입력하는 모습이다. 데이터에서는 4개 데이터가 모두 존재함을 확인할 수 있다.
이후 이름 기반으로 검색을 하며 이후 추가로 오름차순기반으로 출력하는 모습이다.

```
========PRINT_BP========
alex/200/210038/5900
alice/300/220005/1000
bob/100/240011/5900
cristiano/100/220058/9900
eric/100/250011/4000
florian/200/200719/1200
lionel/100/250001/8000
luis/100/230079/5000
mohammed/400/190311/7600
ryan/300/220094/8200
steven/400/170027/9700
========================

========ADD_ST========
Success
====================

========ADD_ST========
Success
====================

========ADD_ST========
Success
====================

========ADD_ST========
Success
====================
```

오름차순 기반으로 출력하는 모습이다.
셀렉션 트리에 추가하는 명령어 수행 모습이다.
-트리를 스캔하여 부서에 저장 및 재정렬하는 모습
-3명을 이름으로 검색해 추가한 후 재정렬 하는모습 부서 100에서 luis, lionel, eric, bob, cristiano이 들어가는 모습, 각부서에 1명씩 추가하는 모습



부서에 기반하여 셀렉션 트리에서의 내용을 출력하는모습
딜리트를 통해서 최대 연봉 직원을 제거하고 재정렬하는 모습
출력시 연봉기준으로 하여 출력되는 모습
이후 재출력을 통해서 삭제가 잘 이루어지는지 확인하는 모습

Consideration

이번 과제를 진행하면서 자료구조에서 b+트리를 통해서 효율적으로 자료를 탐색 및 처리하고 실시간으로 반영할 수 있음을 확인했다. 특히 오름차순 기반으로 앞으로도 유사한 작업을 할 때 매번 정렬을 하지 않아도 됨에 있어서 효율도를 더 높일 수 있음을 확인했다.