□ 프로그램 구현 환경

windows 10 운영체제에서 Eclipse Jee Oxygen를 이용해 java언어로 작성.

□ 함수 설명 및 시간 복잡도

|  |  |
| --- | --- |
| **자료구조**  **함 수** | **Hash Table** |
| hash\_M1 | key의 하위 8bit만 골라내 key가 들어갈 table의 index을 나타냄.  key와 11111111의 &연산만 하면 되므로 시간복잡도 O(1) |
| hash\_M2 | key의 상위 8bit를 골라내 collision시에 key가 jump할 크기.  jump크기가 짝수면 홀수로 바꿈.  hash\_M1와 마찬가지로 비트연산만 하면 되므로 시간복잡도 O(1) |
| Search | key를 찾는 메서드.  key의 bin의 상태가 EMPTY라면 있지 않으므로 NOT FOUND 출력  OCCUPIED상태라면 key값을 확인해서 찾으려는 key와 일치하는지 확인 후  key값 출력. 아니라면 다음 방으로 bin set후 탐색  best case(초기 hash value에 key 있다면) : O(1)(array access)  worst case(테이블의 대부분의 상태가 ERASED인 경우) : O(M)(  M:size of table, key를 찾을 때까지 계속 탐색해야 하므로)  average case : O(1) |
| Insert | key를 table에 넣어주는 메서드.  Search메서드를 호출해 이미 테이블에 key가 들어가있는지 확인 한 후에 없으면  넣고 있으면 아무 행동도 하지 않는다. Search메서드 실행 시에  NOTFOUND와 key값 출력하는 실행문이 있어서 printFlag설정.  table의 상태가 EMPTY나 DELETED일때만 삽입이 가능하므로  OCCUPIED상태가 아니라면 해당 key를 삽입. OCCUPIED상태라면  다음방을 탐색할 위치 bin을 설정.  bin은 더블해쉬 기법으로 key의 init hash value + n\*jump(n:1~)  Search메서드 실행시간 + 데이터 삽입할 공간 찾는 시간 Search와 비슷하므로  best case(초기 hash value에 삽입가능하다면) : O(1)(array access)  worst case(테이블의 대부분의 상태가 OCCUPIED) : O(M)(  M:size of table, 삽입가능한 방을 찾을 때까지 계속 탐색해야 하므로)  average case : O(1) |
| Erase | key를 삭제하는 메서드.  key의 bin의 상태가 EMPTY라면 있지 않으므로 종료  OCCUPIED상태라면 key값을 확인해서 찾으려는 key와 일치하는지 확인 후  맞다면 해당 방 상태 ERASED로 설정. 아니라면 다음 방으로 bin set후 탐색  best case(초기 hash value에 key 있다면) : O(1)(array access)  worst case(테이블의 대부분의 상태가 ERASED인 경우) : O(M)(  M:size of table, key를 찾을 때까지 계속 탐색해야 하므로)  average case : O(1) |

|  |  |
| --- | --- |
| **자료구조**  **함 수** | **Hash Table** |
| Print | 모든 방 탐색해가며 출력해야하므로 O(M) |