Lecture 310 : 해싱의 주요 방법 (Static, Dynamic Hashing)

31.1 정적 해싱 = 사용할 메모리 공간을 미리 확정하는 방법

정적 해싱의 문제점
- 속도는 빠르지만
- 미리 확보해야하는 Table 공간이 부담스러울 수 있다.
- 상황을 예측할 수 없다.
- 어떻게 해야 할까 ?

정적 데이트비용) 항상 5만원만 준비해 간다.
동적 데이터비용) 지난 번 데이트에 사용한 비용을 참고하여 결정한다.

31.2 외출할 때 한번에 100만원을 가져가는 것은 남는 경우 부담스러울 수가 있고, 그렇다고 겨우 2만원만 가져가는 것은 좀 불안하다. <지갑에 돈을 얼마나 넣어가야 가장 합리적인가> 라는 문제를 동적해싱 방법(doubling)을 이용해서 비유적으로 설명하시오.

31.3 해싱의 성능 비교 및 분석: 자료가 N개 있다. 이진탐색과 그 성능을 비교해보자.

자료구조	find K Best	find K worst	find K average	전체 출력	delete	insert x
unsorted array						
sorted array						
sorted list						
unsorted list						
static hash						
dynamic hash						

31.4 성능분석의 위한 기초 : N=10000개의 박스(bin)가 있다. 이 상황에서 우리가 5000개 공을 100000개의 박스에 무작위로 뿌릴 때 다음의 값을 계산해보자.

a. 각 bin에 들어있는 평균 공의 개수

b. 2개의 공이 들어있는 bin의 개수(the expected number of bins with 2 balls)

c. 5공이 들어있는 bin의 예상 개수

d. 하나의 bin에 들어있는 공의 최대 개수 M

31.5 해싱에서 오버플로우(Overflow)를 처리하는 다양한 기법의 이론적 평가

b개의 bucket이 있다. $k_1, k_2, \ldots k_n$ KEY가 있다. 해시함수 h( )가 있고 이 함수는 uniform하게 KEY를 뿌린다. 따라서 적재밀도 $\alpha = n/b$ 이다. 이 경우 이미 존재하여 입력이 된 KEY를 다시 찾는데 걸리는 시간 S_n(Successful)과 지금까지 들어가지 못한 KEY를 넣는 경우 즉 Unsuccessful U_n을 구해보면 다음과 같다.

방법	U_n	S_n
Open Probing	$\frac{1}{2} + \frac{1}{2(1-\alpha)^2}$	$\frac{1}{2} + \frac{1}{2(1-\alpha)}$
Rehashing	$\frac{1}{1-\alpha}$	$-\frac{\log_e(1-\alpha)}{\alpha}$
Chaining	α	$1+\alpha/2$

3.16 위의 표에서 동적해싱을 Chaining으로 처리할 경우의 예를 들어 설명해 보시오.

n의 크기는 100만이고, b는 10만이라고 하자. 즉 하나의 bucket에 평균 10개의 자료가 들어간다고 할 때 실제 위의 값을 구해보자.

방법	U_n (n=100m, b=10m)	S_n (n=100m, b=10m)
Open Probing		
Rehashing		
Chaining		

방법	U_n (n=100m, b=50m)	S_n (n=100m, b=50m)
Open Probing		
Rehashing		
Chaining		

31.7 STL에서 제공해주는 최신의 hashing code를 example을 사용해서 익혀보자.

특히 다차원 Hashing 방법은 꼭 알아두어야 한다. 선언부터 사용까지. 예를 들면

hitting_average[“이대호”][“4월”] 과 같은 형식을 사용할 수 있어야 한다.