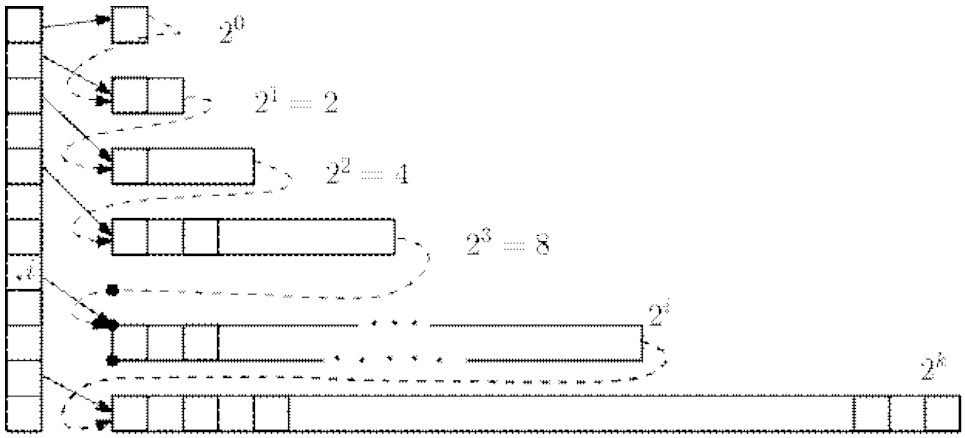
Dynamic Binary Search

[문제] 크기가 N인 sorted linear array에서 특정 원소를 찾는 작업에는 $O(\log N)$ 시간이 걸린다. 이 log time은 매우 빠른 시간이다. 그러나 일차원 배열에서 새로운 원소가 추가(insert)되면 최악의 경우 $O(\log_2 N + N) = \Theta(N)$의 시간이 걸린다. 이러한 추가 삽입 삭제를 위하여 다양한 높이 균형 트리가 존재한다. 우리는 이 문제는 단순한 복수개의 sorted array를 이용하는 새로운 자료구조를 만들어 해결해보고자 한다. 일단 이 자료구조는 search와 insert( )만 제공한다고 가정한다.

원소의 수 N을 이진수 $< n_{k-1}, n_{k-2}, \ldots n_1, n_0 >$ 라고 하자. 특징 이진수 digit $n_i = 0$인지 $n_i = 1$에 따라서 크기 2^i인 정렬된 linear배열 A_i가 대응된다. 즉 원소의 개수는 다음의 식으로 표시된다.

$$\sum_{i=0}^{k-1} n_i 2^i$$

아래는 이 자료구조의 구조를 보여주는 것이며 이러한 Linear array가 linked list로 연결되어 있다고 생각해도 좋다. 또는 하나의 배열에 순차적으로 link되어 있다고 생각할 수 도 있다. 어떤 경우든 자신이 구현한 방법에 따라 time complexity를 구현해야 한다.



위 자료구조에서 두 가지 상황을 가정할 수 있다.

가정 a) 각 배열 A_i는 정렬되어 있지만 다른 A_i와 A_j에는 아무런 관련이 없다.

가정 b) $|A_i|, |A_j| \geq 1$, $i < j$에 대하여 $A_i < A_j$ 가 성립한다. 즉 개별 배열도 전체적으로 정렬되어 있다.

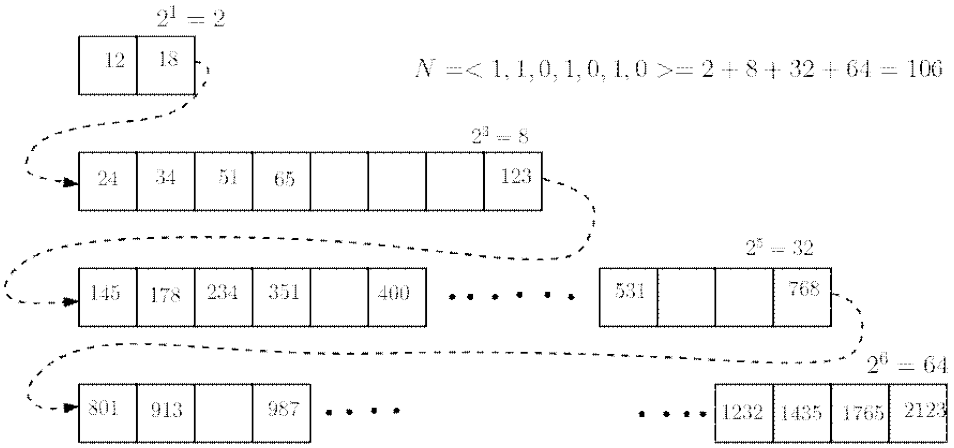
[구현] 시나리오 a)

문제 a-1) search 방법을 제시하고 (worst case) time complexity를 구하시오.

문제 a-2) 그리고 새 원소를 삽입하게 되면 전체 배열의 구조는 바뀌게 된다. 왜냐하면 각 linear array은 가득차(full)있기 때문이다. 이 경우 insert( ) 하는 방법을 제시하고 그 때의 amortized time complexity를 구하시오. 즉 처음 0개의 원소가 있는 상황에서 n개의 원소가 random하게 insert될 때 자신이 구성한 알고리즘에서 basic operation을 얼마나 하는지 그 전체 개수를 세어 보시오.

문제 a-3) 이 자료 구조에서 특정 존재하는 원소를 삭제하는 과정을 보이시오.

이제 시나리오 b)를 가정해본다. 즉 아래 그림과 같이 각 A_i의 값이 순서대로 정렬되어 있다. 이 시나리오 b의 경우에 search 방법을 제시하고 (worst case) time complexity를 구하시오. 그리고 새 원소를 삽입하게 되면 전체 배열의 구조는 바뀌게 된다. 왜냐하면 각 linear array은 가득차(full)있기 때문이다. 이 경우 insert( ) 하는 방법을 제시하고 그 때의 time complexity를 구하시오. 동시에 delete 하는 동작의 복잡도를 계산해 보시오. 아래 그림에 106개의 자료로 구성된 시나리오 b)의 한 예가 있다.



[조건] 시나리오 a나 b)를 선택해서 총 200,000 번의 insert( ), search( ) 작업을 이어서 한다. insert( )와 search( )의 비율은 각각 1/2 이다. 즉 100,000번의 insert와 100,000번의 search가 있다. 즉 원소의 총 개수는 십만개, N=100,000인 셈이다 단 탐색의 경우 원 자료구조에 있는 원소가 70%, 아닌 경우 즉 search fail이 될 경우가 30%이다. 이 경우 자신이 설정한 동작의 시행 횟수를 계산해서 그 횟수를 표로 만들어 제시하고 이 방법으로 실험적인 방법으로 amortized complexity를 계산해보는 것이 이 과제물의 목적이다. 이번 과제의 제출물은 다음과 같이 2개의 파일이다. `NAME_darray.{py,c,c++}`, `NAME_darry_report.pdf` 이다. amortized time complexity analysis의 취지에 맞도록 보고서가 작성되어야 한다. 즉 특정한 순간에는 worst case complexity $f(n)$이 걸리지만 이것이 연속된 동작에는 적용되지 않는다는 것을 이해하는 것이다.

제출은 **22일 금요일 저녁 10시**까지이다. 제출 장소는 이전과 같이 ESPA 게시판이다.