

강의 5: 일정 예약과 이진 탐색 트리

강의 개요

- 활주로 예약 시스템
 - 정의
 - 리스트로 해결하는 방법
- 이진 탐색 트리
 - 계산 과정

읽을거리

CLRS 10장, 12.1-3

활주로 예약 시스템

- 활주로는 매우 혼잡하고 단 하나뿐인 공항 (보스턴 6 -> 1)
- 향후 착륙을 위한 “예약”
- 비행기가 착륙하면 대기 중인 사건 집합에서 제거하기
- 각각의 착륙 요청에는 “요청 예약 시간” t 가 명시되어 있음
- t 로부터 k 분 전이나 k 분 후에 다른 착륙 예약이 없다면, t 를 집합에 추가함. k 값은 변할 수 있음
 - 그 외의 경우는 에러로 보고, 예약을 잡지 않음

예시

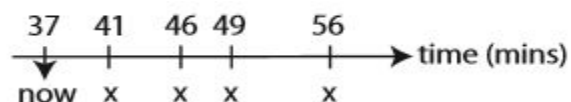


그림 1: 활주로 예약 시스템 예시

R 은 예약된 착륙 시간을 나타냄: $R = (41, 46, 49, 56)$ 과 $k = 3$

착륙 시간 요청: 44는 허용되지 않음 ($46 \in R$)

53 허용

20 허용되지 않음 (이미 지난 시간)

$|R| = n$

목표: 본 시스템을 $O(\lg n)$ 시간 안에 효율적으로 운영하기.

알고리즘

R을 정렬된 리스트로 유지하기.

```

init: R = [ ]
req(t): if t < now: return "error"

for i in range (len(R)):
    if abs(t-R[i]) < k: return "error"
R.append(t)
R = sorted(R)
land: t = R[0]
if (t != now) return error
R = R[1: ]          (drop R[0] from R)

```

더 나은 방법은?

- **정렬된 리스트:** 추가 및 정렬에는 $\Theta(n \lg n)$ 시간이 소요된다. 추가와 정렬 과정을 하지 않고 새로운 시간/비행기를 삽입할 수 있다. 그러나 삽입에는 $\Theta(n)$ 만큼 시간이 소요된다. 삽입 위치가 파악되기만 하면 k분 떨어져 있는지 확인하는 작업은 $O(1)$ 안에 가능하다.

- **정렬된 배열:** 이진 탐색을 이용하여 $O(\lg n)$ 시간 안에 삽입하려는 위치를 찾을 수 있다. 이진 탐색을 이용하여 $R[i] \geq t$ 를 만족하는 최소의 i 값을 찾을 수 있다. (예를 들면, 다음으로 큰 요소) 그 다음 t에 대해 $R[i]$ 와 $R[i - 1]$ 비교한다. 실제 삽입 과정에는 $\Theta(n)$ 시간이 소요되는 자리 이동 필요하다.

- **정렬되지 않은 리스트/배열:** k분 떨어져 있는지 확인하는 데에 $O(n)$ 만큼 시간이 소요된다.

- **최소-힙:** $O(\lg n)$ 시간 안에 삽입 가능하다. 그러나 k분 떨어져 있는지 확인하는 데에는 $O(n)$ 시간만큼 소요된다.

- **딕셔너리 또는 파이썬 Set:** 삽입은 $O(1)$ 만큼 시간이 소요된다. k분 떨어져

있는지 확인하는 데에는 $\Omega(n)$ 만큼 시간이 소요된다.

전체 분들을 다 저장하면 어떨까?

이 트릭은 시간에 대해 인덱스된 큰 배열을 이용해 구현할 수 있다. 하지만 이 트릭은 임의의 정밀도 시간이나 착륙을 위해 너비 슬롯 확인에 대해서는 작동하지 않는다.

핵심: 정렬된 리스트에 빠르게 삽입하는 것이 필요하다.

이진 탐색 트리 (BST)

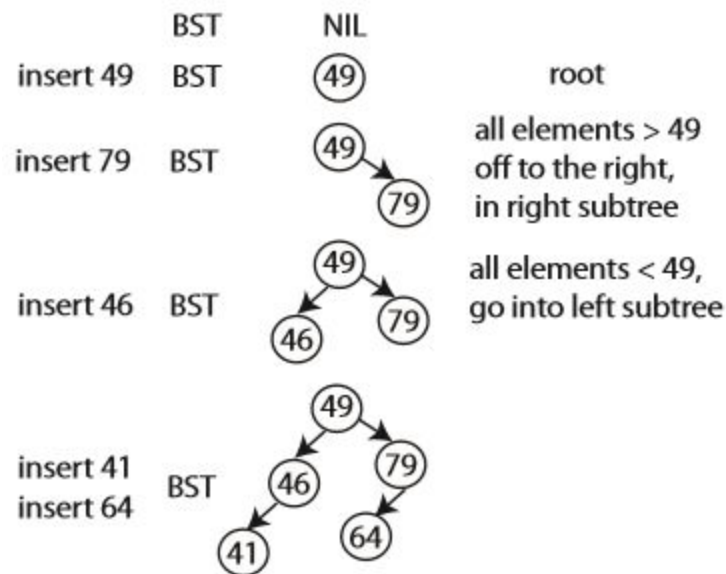


그림 2: 이진 탐색 트리

특성

이진 트리의 각 노드 x 에는 키인 $key(x)$ 가 있다. 루트가 아닌 다른 노드는 부모인 $p(x)$ 가 있다. 노드에는 왼쪽 자식인 $left(x)$ 와 오른쪽 자식인 $right(x)$ 이 있다. 이것들은 힙과는 달리 포인터이다.

불변성: 모든 노드 x 와 x 의 왼쪽 하위 트리에 있는 모든 노드 y 에 대해서 $key(y) \leq key(x)$ 이다.

또한 x 의 오른쪽 하위 트리에 있는 모든 노드 y 에 대해 $key(y) \geq key(x)$ 이다.

삽입: $insert(val)$

그림 2와 같이, 알맞은 위치를 찾을 때까지 왼쪽 및 오른쪽 포인터를 따라 내려간다. 삽입을 하는 도중에 활주로 예약 시스템에서 “ $k=3$ 내”에 다른 예약이 있는지 확인이 가능하다. 루트에서부터 보았을 때, 삽입을 원하는 값의 $k=3$ 이내에 있는 다른 요소를 발견하게 되면, 과정을 중단하고 삽입하지 않는다.

이진 탐색 트리에 값이 존재하는 경우, 그 값 찾기: $find(val)$

값을 찾거나 NIL에 도달할 때까지 왼쪽과 오른쪽 포인터들을 따라간다.

이진 탐색 트리에서 최소 요소 찾기: findmin()

더 이상 왼쪽으로 갈 수 없을 때까지 왼쪽으로 가는 것이 중요하다.

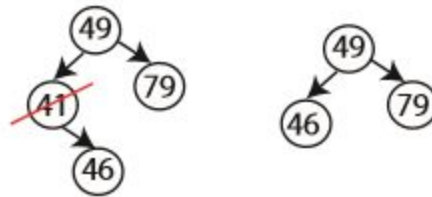


그림 3: 최소값 제거: 최소값을 찾고 제거함

복잡성

이진 탐색 트리의 높이가 h 일 때, 모든 연산 과정에는 $O(h)$ 가 소요된다.

다음으로 큰 요소 찾기: next-larger(x)

x 는 값이 아니라 이진 탐색 트리에서의 노드라는 점 주의.

next-larger(x)

오른쪽 자식이 NIL이 아닐 경우, minimum(right)를 리턴하기

NIL인 경우, $y = \text{parent}(x)$

y 가 NIL이 아니고 $x = \text{right}(y)$ 인 경우,

$x = y; y = \text{parent}(y)$

return(y);

예시인 그림 4를 보라. next-larger(find(46))는 무엇을 리턴할까?

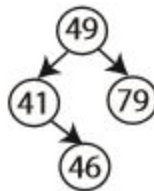


그림 4: 다음으로 큰(x)

새로운 요구 조건

Rank(t): 시간 $\leq t$ 에 착륙 예정인 비행기는 몇 대일까? 새로운 요구 조건은 설계도의 수정을 필요로 한다.

현재 우리가 가진 것으로는 효율적으로 해결할 수 없다. 하지만 전체적인 이진 탐색 트리 구조를 보완하면 가능하다.

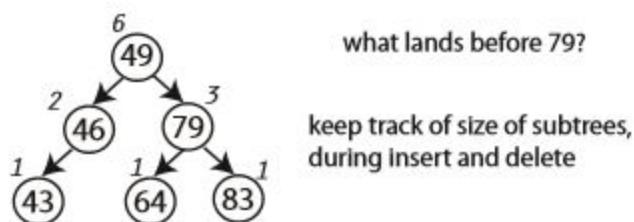


그림 5: 이진 탐색 트리 구조 보완

그림 5를 요약하면, 보완을 위한 알고리즘은 다음과 같다:

1. 원하는 시간을 찾기 위해 트리를 따라 내려오기
2. 노드의 값이 찾는 값보다 작다면 1 더하기
3. 왼쪽 서브 트리의 크기만큼 더하기

전체적으로, 이것은 $O(h)$ 만큼의 시간이 소요된다.

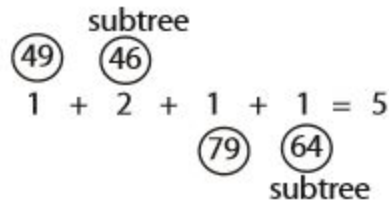


그림 6: 보완 알고리즘 예제

여기에서 논의되는 이진 탐색 트리에 대한 파이썬 코드는 이 링크에서 볼 수 있다. [at this link](#).

우리는 무엇을 성취 했나?

트리의 높이 h 는 $O(\lg n)$ 이어야 한다.

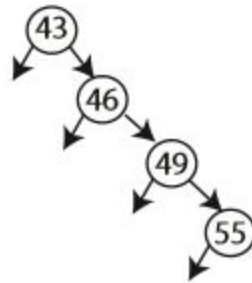


그림 7: 이진 탐색 트리에 정렬된 순서로 삽입

그림 7의 트리는 연결된 리스트 같아 보인다. 우리는 $O(\lg n)$ 이 아닌 $O(n)$ 을 달성했다. 다음 강의에서 배울 균형 이진 탐색 트리가 이를 해결해줄 예정이다!

MIT OpenCourseWare

<http://ocw.mit.edu>

6.006 알고리즘의 기초

가을 2011

본 자료 이용 또는 이용 약관에 대한 정보를 확인하려면 다음의 사이트를 방문하십시오:

<http://ocw.mit.edu/terms>.