Skip Lists

Jie Zhu, Jingwen Pu, Chuanyang Cheng

May 20, 2020

Chollege of Computer and Technology, Zhejiang University

Outline

Introduction

Definition

A skip list is a data structure that allows $O(n)$ search complexity as well as $O(\log n)$ insertion complexity within an ordered sequence of n elements. [1]

Linked List

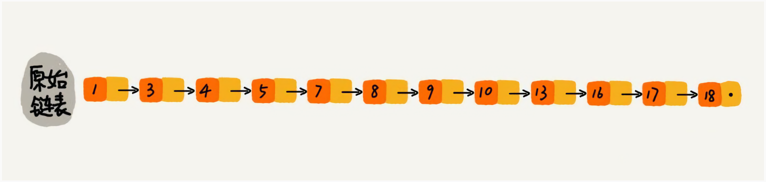

Figure 1: A Linked List

Skip List

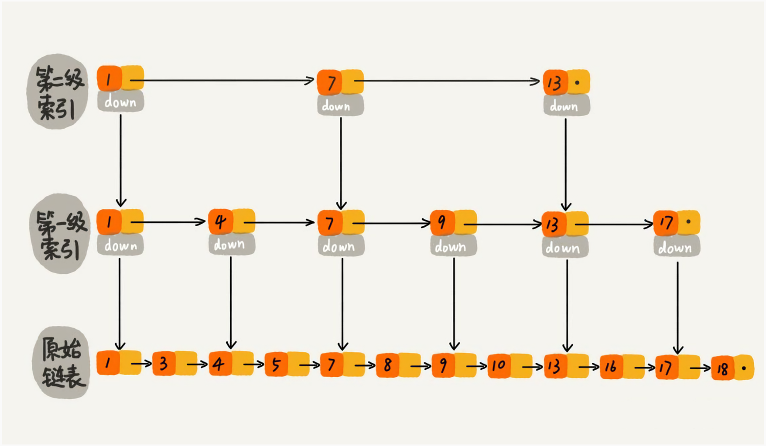

Figure 2: A Skip List

Complexity in Big-O Notation

Algorithm	Average	Worst Case
Space	$O(n)$	$O(n \log n)$ [2]
Search	$O(\log n)$	$O(n)$ [2]
Insert	$O(\log n)$	$O(n)$
Delete	$O(\log n)$	$O(n)$

Table 1: Complexity

Implement

Data Structure Definition

1. dsa
2. jll;
3. dkal

Implement

Search

Search

```
Search(list, searchKey)
  x:=list→header
  --loop invariant:  x→key
  for i:=list→level downto 1 do
    while  x→forward[i]→key < searchKey do
      x:=x→forward[i]
  --x→key < searchKey ≤ x→forward[1]→key
  x:=x→forward[1]
  if x→key = searchKey  then return x→value
    else return  failure
```

Implement

Insert

Random Level

```
RandomLevel()
  newLevel:=1
  --random()returns a random value in [0, 1)
  while random() $<$ p do
    newLevel:=newLevel + 1
  return min(newLevel, MaxLevel)
```

Insert i

```
Insert(list, searchKey, newValue)
  local update[1...MaxLevel]
  x:=list→header
  for i:=list→level downto 1 do
    while  x→forward[i]→key < searchKey do
      x:=x→forward[i]
    --x→key < searchKey ≤ x→forward[1]→key
    update[i]:=x
  x:=x→forward[1]
```

Insert ii

```
if x→key = searchKey then x→value:=newValue
else
  newLevel:=RandomLevel()
  if newLevel > list→level then
    for i:=list→level+1 to newLevel do
      update[i]:=list→header
    list→level:=newLevel
  x:=makeNode(newLevel, searchKey, value)
```

Insert iii

```
for i:=1 to newLevel do
    x→forward[i]:=update[i]→forward[i]
    update[i]→forward[i]:=x
```

Implement

Delete

Delete i

```
Delete(list, searchKey, newValue)
  local update[1...MaxLevel]
  x:=list→header
  for i:=list→level downto 1 do
    while x→forward[i]→key < searchKey do
      x:=x→forward[i]
    update[i]:=x
  x:=x→forward[1]
```

Delete ii

```
if x→key = searchKey then
  for i:=1 to list→level do
    if update[i]→forward[i] $\neq$ x then break
    update[i]→forward[i]:=x→forward[i]
  free(x)
  while list→level $>$ 1 and
    list→header→forward[list→level] = NIL do
    list→level:=list→level-1
```

Complexity Analysis

Complexity Analysis

Space Complexity Analysis

References

William Pugh.
Skip lists: a probabilistic alternative to balanced trees.
Communications of the ACM, 33(6):668–676, 1990.

Thomas Papadakis.
Skip lists and probabilistic analysis of algorithms.
University of Waterloo Ph. D. Dissertation, 1993.

Thank you!