

資料結構

Data Structure

TZU-CHUN HSU¹

¹vm3y3rmp40719@gmail.com

¹Department of Computer Science, Zhejiang University



2020 年 12 月 18 日
Version 3.0

Disclaimer

本文「資料結構」為台灣研究所考試入學的「資料結構」考科使用，內容主要參考 Introduction to Algorithms[1]，以及 wjungle 網友在 PTT 論壇上提供的資料結構筆記 [2]。本文作者為 TZU-CHUN HSU，本文及其 L^AT_EX 相關程式碼採用 MIT 協議，更多內容請訪問作者之 GITHUB 分頁 [Oscarshu0719](#)。

MIT License

Copyright (c) 2020 TZU-CHUN HSU

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1 Overview

1. 本文頁碼標記依照 TKB 筆記 [2] 的頁碼。

2. TKB 筆記 [2] 章節頁碼：

Chapter	Page No.	Importance
1	3	***
2	259	*
3	52	***
4	259	*
5	82	*****
6	228	****
7	180	****
8	221	***
9	129	****

3.

$$\log 2 = 0.3010$$

$$\log 3 = 0.4771$$

$$\log 5 = 0.6990$$

$$\log 7 = 0.8451$$

(1)

4. OBST 在「演算法」中，不再贅述。

Trees				
Tree	Insert x	Delete x	Search x	Remark
BST	$O(\log n) \sim O(n)$			Create: $O(n \log n) \sim O(n^2)$
AVL tree	$O(\log_m n)$			$F_{h+2} - 1 \leq n \leq 2^h - 1$
B tree				$1 + 2^{\frac{\lceil \frac{m}{2} \rceil^{h-1} - 1}{\lceil \frac{m}{2} \rceil - 1}} \leq n \leq 2^{\lceil \frac{m}{2} \rceil^{h-1} - 1}$
RBT				$h \leq 2 \log(n + 1)$
Splay tree				Worst: $O(n)$, Amortized: $O(\log n)$

Priority queues					
Operations	Max (Min)	Min-max & Deap & SMMH	Leftist	Binomial	Fibonacci
Insert x	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n), O(1)^*$	$O(1)^*$
Delete max	$O(\log n)$	$O(\log n)$			
Delete min	$O(n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)^*$
Delete x				$O(\log n)$	$O(\log n)^*$
Merge	$O(n)$		$O(\log n)$	$O(\log n)$	$O(1)^*$
Decrease key				$O(\log n)$	$O(1)^*$
Search x	$O(n)$				
Find max	$O(1)$	$O(1)$			
Find min		$O(1)$		$O(\log n)$	$O(1)$
Remark			$\text{shortest}(\text{root}) \leq \log(n+1) - 1$		

Sorting algorithms					
Method	Time complexity			Space complexity	Stable
	Best	Worst	Average		
Insertion	$O(n)$	$O(n^2)$		$O(1)$	✓
Selection	$O(n^2)$			$O(1)$	×
Bubble	$O(n)$	$O(n^2)$		$O(1)$	✓
Shell	$O(n^{1.5})$	$O(n^2)$		$O(1)$	×
Quick	$O(n \log n)$	$O(n^2)$	$O(n \log n)$	$O(n \log n) \sim O(n)$	×
Merge	$O(n \log n)$			$O(n)$	✓
Heap	$O(n \log n)$			$O(1)$	×
LSD Radix	$O(n \times k)$			$O(n + k)$	✓
Bucket/MSD Radix	$O(n)$	$O(n^2)$	$O(n + k)$	$O(n \times k)$	✓
Counting	$O(n + k)$				✓

2 Summary

1. Theorem (17) Permutation:

```
1: function PERM(list, i, n)
2:   if i == n then
3:     PRINT(list)
4:   else
5:     for j := i to n do
6:       SWAP(list, i, j)
7:       PERM(list, i + 1, n)
8:       SWAP(list, i, j)
9:     end for
10:  end if
11: end function
```

2. Theorem (87) 節點數:

$$n = \left(\sum_{i=1}^{\deg} i \times n_i \right) + 1 \quad (2)$$
$$n_0 = n_2 + 1 \text{ (二叉樹)}$$

3. Theorem (95, 97, 98)

- 可以確定二叉樹，其他則否：
 - Preorder 和 Inorder。
 - Postorder 和 Inorder。
 - Level-order 和 Inorder。
 - Complete 和任意排序。
- Preoder = Inoder: Empty、Root、Right-skewed tree。
- Postoder = Inoder: Empty、Root、Left-skewed tree。
- Preoder = Postoder: Empty、Root。

4. Theorem (116)

```

1: function CREATEMINHEAP(Tree  $s$ , size  $n$ )
2:   for  $i := n/2$  to 1 do                                ▷ Start from parent of the last node.
3:      $tmp := s[i]$ 
4:      $j := 2 \times i$                                          ▷ Left child of  $i$ .
5:     while  $j \leq n$  do                                   ▷ There is a child.
6:       if  $j < n$  then                                     ▷ Right child exists.
7:         if  $s[j] > s[j+1]$  then                             ▷ Choose the smaller child.
8:            $j := j + 1$ 
9:         end if
10:      end if
11:      if  $tmp \leq s[j]$  then
12:        Break.
13:      else                                                ▷ Percolate one level.
14:         $s[j/2] := s[j]$ 
15:         $j := j \times 2$ 
16:      end if
17:    end while
18:     $s[j/2] := tmp$ 
19:  end for
20: end function

```

5. Theorem (162) Red-black tree:

```

1: function INSERTREDBLACKTREE(RedBlackTree  $s$ , Element  $x$ )
2:    $x.color := red$ .
3:   INSERTBST( $x$ ).
4:   while  $x.parent.color = red$  do
5:     if  $x.parent$  is left child then
6:       if  $x.parent.sibling.color = red$  then
7:          $x.parent.color := black$ 
8:          $x.parent.sibling.color := black$ 
9:          $x.parent.parent.color := red$ 
10:         $x := x.parent.parent$ 
11:      else if  $x$  is right child then
12:         $x := x.parent$ 
13:        LEFTROTATE( $s, x$ )
14:      else
15:         $x.parent.color := black$ 
16:         $x.parent.parent.color := red$ 
17:        RIGHTROTATE( $s, x.parent.parent$ )
18:      end if
19:    else
20:      Similar to the process above, just change LEFTROTATE and RIGHTROTATE.
21:    end if
22:  end while
23:   $s.root := black$ 
24: end function

```

```

1: function DELETEREDBLACKTREE(RedBlackTree  $s$ , Element  $x$ )
2:    $org - color := x.color$ 
3:   if  $x$  is leaf then
4:     Set link from  $x.parent$  to  $x$  as NIL.
5:   else if  $x$  has 1 child then
6:     Replace  $x$  with its child.
7:   else ▷  $x$  has 2 children
8:     Replace  $x$  with largest in left subtree or smallest in right subtree.
9:   end if
10:  if  $org - color = black$  then
11:    DELETEDFIXREDBLACKTREE( $s, x$ )
12:  end if
13: end function

```

```

1: function DELETEFIXREDBLACKTREE(RedBlackTree  $s$ , Element  $x$ )
2:   while  $x \neq s.root \wedge x.color = black$  do
3:     if  $x$  is left child then
4:        $w := x.sibling$ 
5:       if  $w.color = red$  then
6:          $w.color := black$ 
7:          $x.parent.color := red$ 
8:         LEFTROTATE( $s, x.parent$ )
9:          $w := x.sibling$ 
10:      else if  $w.lchild.color = black \wedge w.rchild.color = black$  then
11:         $w.color := red$ 
12:         $x := x.parent$ 
13:      else if  $w.rchild.color = black$  then
14:         $w.lchild.color := black$ 
15:         $w.color := red$ 
16:        RIGHTROTATE( $s, w$ )
17:         $w := x.sibling$ 
18:      else
19:         $w.color := x.parent.color$ 
20:         $x.parent.color := black$ 
21:         $w.rchild.color := black$ 
22:        LEFTROTATE( $s, x.parent$ )
23:         $x = s.root$ 
24:      end if
25:    else
26:      Similar to the process above, just change LEFTROTATE and RIGHTROTATE.
27:    end if
28:  end while
29:   $x.color = black$ 
30: end function

```

6. Theorem (195) Quick sorting:

```

1: function QUICKSORT(Array  $A$ , index  $p, r$ )                                ▷ Sorting from  $A[p]$  to  $A[r]$ 
2:   if  $p < r$  then
3:      $q := \text{PARTITION}(A, p, r)$ 
4:     QUICKSORT( $A, p, q - 1$ )
5:     QUICKSORT( $A, q + 1, r$ )
6:   end if
7: end function

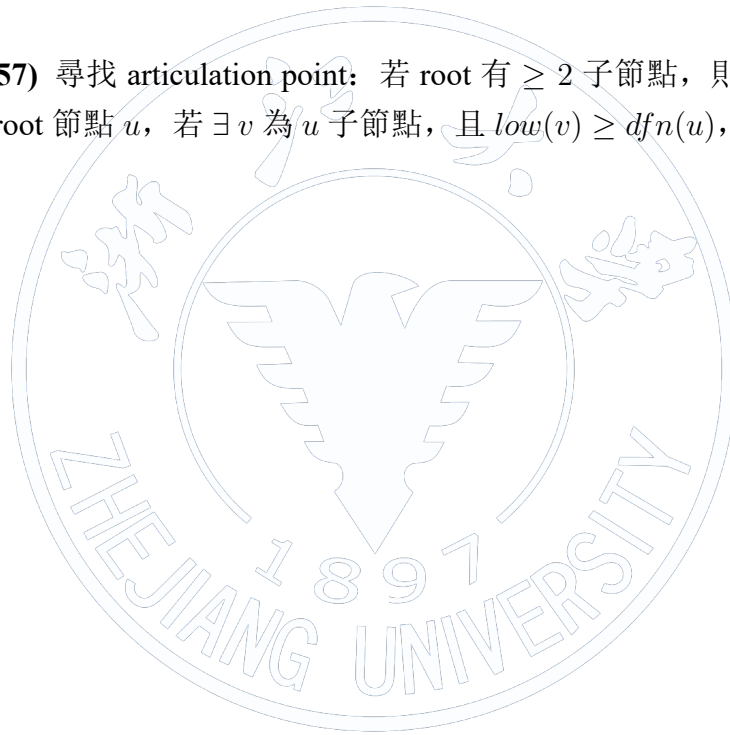
```

```

1: function PARTITION(Array  $A$ , index  $p$ ,  $r$ )
2:    $x := A[r]$  ▷ Pivot.
3:    $i := p - 1$ 
4:   for  $j := p$  to  $r - 1$  do
5:     if  $A[j] \leq x$  then
6:        $i := i + 1$ 
7:       SWAP( $A[i]$ ,  $A[j]$ )
8:     end if
9:   end for
10:  SWAP( $A[r]$ ,  $A[i + 1]$ )
11:  return  $i + 1$ 
12: end function

```

7. **Theorem (257)** 尋找 articulation point: 若 root 有 ≥ 2 子節點, 則 root 為 articulation point; \exists 非 root 節點 u , 若 $\exists v$ 為 u 子節點, 且 $low(v) \geq dfn(u)$, 則 u 為 articulation point。



References

- [1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3 edition, 2009.
- [2] wjungle@ptt. 資料結構 @tkb 筆記. <https://drive.google.com/file/d/0B8-2o6L73Q2VeFpGejlYRk1WeFk/view?usp=sharing>, 2017.

