資料結構

Data Structure

TZU-CHUN HSU[1]

[1]vm3y3rmp40719@gmail.com

[1]Department 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 LaTeX 相關程式碼採用 **MIT 協議**，更多內容請訪問作者之 GitHub 分頁Oscarshu0719。

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	$\star\star\star$
2	259	$\star$
3	52	$\star\star\star$
4	259	$\star$
5	82	$\star\star\star\star\star$
6	228	$\star\star\star\star$
7	180	$\star\star\star\star$
8	221	$\star\star\star$
9	129	$\star\star\star\star$

3.
$$\begin{aligned}\log 2 &= 0.3010\\ \log 3 &= 0.4771\\ \log 5 &= 0.6990\\ \log 7 &= 0.8451\end{aligned} \tag{1}$$

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	$O(\log_m n)$			$F_{h+2} - 1 \le n \le 2^h - 1$
B tree				$1 + 2\frac{\lceil \frac{m}{2}\rceil^{h-1}-1}{\lceil \frac{m}{2}\rceil-1} \le n \le 2\lceil \frac{m}{2}\rceil^{h-1} - 1$
RBT				$h \le 2\log(n+1)$
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			$\textit{shortest}(\text{root}) \leq \log(n+1) - 1$		

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

2 Summary

1. **Theorem (17)** Permutation：

```
function PERM(list, i, n)
    if i == n then
        PRINT(list)
    else
        for j := i to n do
            SWAP(list, i, j)
            PERM(list, i + 1, n)
            SWAP(list, i, j)
        end for
    end if
end function
```

2. **Theorem (87)** 節點數：

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

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)**

function CreateMinHeap(Tree s, size n)
  for $i := n/2$ to 1 **do** ▷ Start from parent of the last node.
    $tmp := s[i]$
    $j := 2 \times i$ ▷ Left child of i.
    while $j \leq n$ **do** ▷ There is a child.
      if $j < n$ **then** ▷ Right child exists.
        if $s[j] > s[j+1]$ **then** ▷ Choose the smaller child.
          $j := j + 1$
        end if
      end if
      if $tmp \leq s[j]$ **then**
        Break.
      else ▷ Percolate one level.
        $s[j/2] := s[j]$
        $j := j \times 2$
      end if
    end while
    $s[j/2] := tmp$
  end for
end function

5. **Theorem (162)** Red-black tree:

```
function InsertRedBlackTree(RedBlackTree s, Element x)
    x.color := red.
    InsertBST(x).
    while x.parent.color = red do
        if x.parent is left child then
            if x.parent.sibling.color = red then
                x.parent.color := black
                x.parent.sibling.color := black
                x.parent.parent.color := red
                x := x.parent.parent
            else if x is right child then
                x := x.parent
                LeftRotate(s, x)
            else
                x.parent.color := black
                x.parent.parent.color := red
                RightRotate(s, x.parent.parent)
            end if
        else
            Similar to the process above, just change LeftRotate and RightRotate.
        end if
    end while
    s.root := black
end function
```

```
function DeleteRedBlackTree(RedBlackTree s, Element x)
    org − color := x.color
    if x is leaf then
        Set link from x.parent to x as NIL.
    else if x has 1 child then
        Replace x with its child.
    else                                                  ▷ x has 2 children
        Replace x with largest in left subtree or smallest in right subtree.
    end if
    if org − color = black then
        DeleteFixRedBlackTree(s, x)
    end if
end function
```

```
function DeleteFixRedBlackTree(RedBlackTree s, Element x)
    while x ≠ s.root ∧ x.color = black do
        if x is left child then
            w := x.sibling
            if w.color = red then
                w.color := black
                x.parent.color := red
                LeftRotate(s, x.parent)
                w := x.sibling
            else if w.lchild.color = black ∧ w.rchild.color = black then
                w.color := red
                x := x.parent
            else if w.rchild.color = black then
                w.lchild.color := black
                w.color := red
                RightRotate(s, w)
                w := x.sibling
            else
                w.color := x.parent.color
                x.parent.color := black
                w.rchild.color := black
                LeftRotate(s, x.parent)
                x = s.root
            end if
        else
            Similar to the process above, just change LeftRotate and RightRotate.
        end if
    end while
    x.color = black
end function
```

6. **Theorem (195)** Quick sorting:

```
function QuickSort(Array A, index p, r)                ▷ Sorting from A[p] to A[r]
    if p < r then
        q := Partition(A, p, r)
        QuickSort(A, p, q − 1)
        QuickSort(A, q + 1, r)
    end if
end function
```

```
function Partition(Array A, index p, r)
    x := A[r]                                  ▷ Pivot.
    i := p − 1
    for j := p to r − 1 do
        if A[j] ≤ x then
            i := i + 1
            Swap(A[i], A[j])
        end if
    end for
    Swap(A[r], A[i + 1])
    return i + 1
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.