

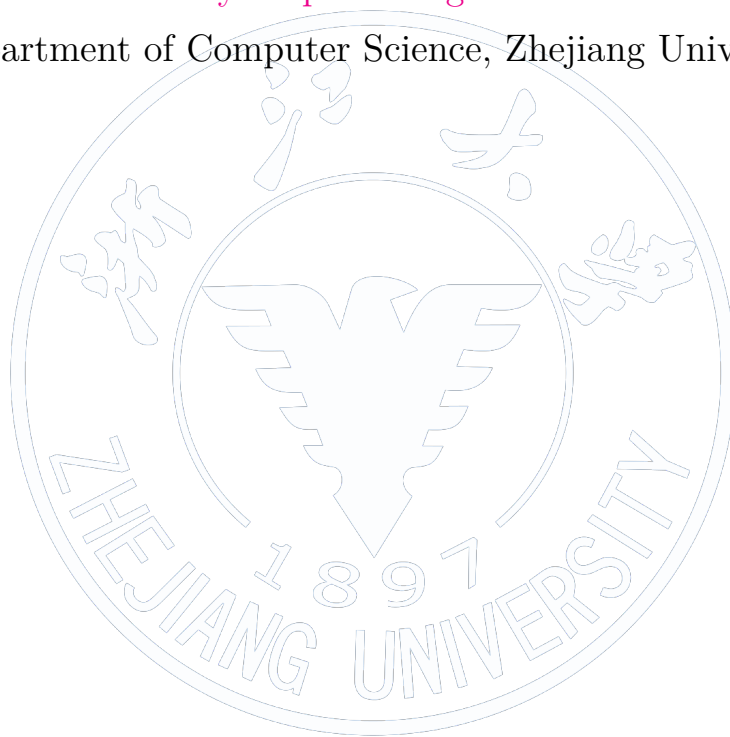
# 資料結構和演算法

## Data Structure and Algorithm

TZU-CHUN HSU<sup>1</sup>

<sup>1</sup>[vm3y3rmp40719@gmail.com](mailto:vm3y3rmp40719@gmail.com)

<sup>1</sup>Department of Computer Science, Zhejiang University



2020 年 10 月 21 日  
Version 1.0

# Disclaimer

本文「資料結構與演算法」為「資料結構」和「演算法」筆記的總整理，內容主要參考 Introduction to Algorithms[2] 和洪捷先生的演算法參考書 [1]，以及 wjungle 網友在 PTT 論壇上提供的資料結構筆記 [3][4]。

本文作者為 TZU-CHUN HSU，本文及其 L<sup>A</sup>T<sub>E</sub>X 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 Summary

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	$O(\log_m n)$			$F_{n+2} - 1 \leq n \leq 2^h - 1$
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: $O(n)$

Disjoint set		
Combination	Union	Find
Arbitrary Union & Simple find	$O(1)$	$O(h)$ Worst: $O(n)$
Union-by-height & Simple find	$O(1)$	$O(\log n)$
Union-by-height & Find with path compression	$O(1)$	$O(\alpha(m, n)) = O(\log^* n)$ close to $O(1)$

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			Merge faster than heap.	Find min can be down to $O(1)$ .	Decrease key is faster than binomial heap

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	$O(n + k)$				✓

Graph algorithms		
Problem	Time complexity	Remark
Depth-First Search (DFS)	$O( V  +  E )$	
Kosaraju's	$O( V  +  E )$	
Kruskal's	$O( E  \log  V )$	
Prim's (Adjacency matrix)	$O( V ^2)$	
Prim's (Heap, Adjacency lists)	$O( E  \log  V )$	
Prim's (Fibonacci heap, Adjacency lists)	$O( E  +  V  \log  V )$	
Sollin's (Borůvka's)	$O( E  \log  V )$	
Dijkstra's (Min-heap)	$\Theta(( V  +  E ) \log  V )$	Greedy, no negative edges or cycles
Dijkstra's (Fibonacci-heap)	$\Theta( E  +  V  \log  V )$	
Bellman-Ford	$O( V  E )$	DP, no negative cycles
Floyd-Warshall	$\Theta( V ^3)$	DP, no negative cycles
Johnson's	$\Theta( V  E  +  V ^2 \log  V )$	No negative cycles
Ford-Fulkerson	$O( E  f^* )$	Greedy, $f^*$ 為最大流
Edmond-Karp	$O( V  E ^2)$	

Dynamic Programming algorithms			
Problem	Time complexity	Space complexity	Remark
Making change	$O(kn)$	$O(n)$	
Fractional Knapsack problem	$\Theta(n \log n)$	$O(n)$	Greedy
0/1 Knapsack problem (DP)	$O(n2^{\log W})$	$O(n2^{\log W})$	
0/1 Knapsack problem (Branch-and-Bound)	$O(2^n)$		
Longest Common Subsequence (LCS)	$O(mn)$	$O(mn)$	不必連續
Longest Increasing Subsequence (LIS)	$O(n^2)$	$O(n^2)$	
Longest Common Substring	$O(mn)$	$O(mn)$	必須連續
Minimum Edit Distance	$O(mn)$	$O(mn)$	
Matrix-chain Multiplication	$O(n^3)$	$O(n^2)$	
Traveling Salesperson problem	$\Theta(n^2 2^n)$	$O(n 2^n)$	
Optimal Binary Search Tree (OBST)	$\Theta(n^3)$	$\Theta(n^2)$	

Computational Geometry algorithms	
Problem	Time complexity
平面上點的rank	$\Theta(n \log n)$
Maximal points	$\Theta(n \log n)$
Closest pair	$O(n \log n)$
Farthest pair	$O(n \log n)$
Graham scan	$\Theta(n \log n)$

## References

- [1] 洪捷. 演算法—名校攻略秘笈. 鼎茂圖書出版股份有限公司, 9 edition, 2017.
- [2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3 edition, 2009.
- [3] wjungle@ptt.           Tkb   筆記.           <https://drive.google.com/file/d/0B8-2o6L73Q2VVmNWQk9DY3hsUm8/view?usp=sharing>, 2017.
- [4] wjungle@ptt.           Tkb   筆記.           <https://drive.google.com/file/d/0B8-2o6L73Q2VeFpGejlYRk1WeFk/view?usp=sharing>, 2017.

