

# 演算法

# 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 11 月 22 日  
Version 2.0

# Disclaimer

本文「演算法」為台灣研究所考試入學的「演算法」考科使用，內容主要參考洪捷先生的演算法參考書 [1]，以及 wjungle 網友在 PTT 論壇上提供的演算法筆記 [2]。  
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 Overview

1. 本文頁碼標記依照實體書 [1] 的頁碼。
2. TKB 筆記 [2] 章節頁碼：

Chapter	Page No.	Importance
1	1	★★★★
2	13	★★★★
3	18	★★★★★
4	34	★★★★★
5	43	★★★
6	48	★★★
7	×	★
8	×	★★★

3. 必考：（參考 TKB 筆記 [2] 中頁碼）

(a) 4

4. 省略第 7 章。

## 2 Summary

### 1. Theorem (335) 0/1 Knapsack problem (Branch-and-Bound):

- 依物品的單位價值從大到小排序。
- Bounding function 為目前價值加上通過 Fractional Knapsack problem 在總重不超過限制的情況下，拿剩下物品得到的值。
- 使用 Priority queue。

### 2. Theorem (89) Longest Common Subsequence (LCS):

•

$$c[i, j] = \begin{cases} 0 & , i = 0 \vee j = 0 \\ c[i - 1, j - 1] + 1 & , i, j > 0 \wedge x_i = y_j \\ \max(c[i, j - 1], c[i - 1, j]) & , i, j > 0 \wedge x_i \neq y_j \end{cases} \quad (1)$$

- $c[0 \cdots \text{Length}(X)][0 \cdots \text{Length}(Y)]$ ,  $c[0, 0]$  表示空字串，並初始化第一行及第一列為 0。
- 字符不同時，標示左邊或上面較大值方向，數值相同時預設  $\uparrow$ ；字符相同時標示  $\nwarrow$ 。

### 3. Theorem (94) Longest Common Substring:

•

$$c[i, j] = \begin{cases} 0 & , i = 0 \vee j = 0 \vee x_i \neq y_j \\ c[i - 1, j - 1] + 1 & , x_i = y_j \end{cases} \quad (2)$$

- $c[0 \cdots \text{Length}(X)][0 \cdots \text{Length}(Y)]$ ,  $c[0, 0]$  表示空字串，並初始化第一行及第一列為 0。

### 4. Theorem (94) Minimum Edit Distance:

•

$$c[i, j] = \min \begin{cases} c[i - 1, j] + 1 & , a_i \neq b_j \\ c[i, j - 1] + 1 & , a_i \neq b_j \\ c[i - 1, j - 1] + 1 & , a_i \neq b_j \\ c[i - 1, j - 1] & , a_i = b_j \end{cases} \quad (3)$$

- 各情況依序表示刪除  $\uparrow$ 、插入  $\leftarrow$ 、替換  $\searrow$  以及匹配  $\swarrow$  <sup>2</sup>。
- $c[0 \cdots \text{Length}(X)][0 \cdots \text{Length}(Y)]$ ,  $c[0, 0]$  表示空字串，並初始化第  $i$  行為  $i$  並標示  $\uparrow$ ，第  $j$  列為  $j$  並標示  $\leftarrow$ 。
- 字符不同時，標示左邊（刪除）、上面（插入）與左上（替換）較小值方向；字符相同時標示  $\swarrow$  <sup>2</sup>。

#### 5. Theorem (100) Matrix-chain Multiplication:

- $$m[i, j] = \begin{cases} 0 & , i = j \\ \min_{i \leq k \leq j-1} \{m[i, k] + m[k+1, j] + p_{i-1}p_kp_j\} & , i < j \end{cases} \quad (4)$$
- $p[0 \cdots \text{Number}(\text{Matrices})]$ ，存入矩陣大小。
- $m[1 \cdots \text{Number}(\text{Matrices})][1 \cdots \text{Number}(\text{Matrices})]$ ，初始化對角線上元素為 0。
- $s[1 \cdots \text{Number}(\text{Matrices}) - 1][2 \cdots \text{Number}(\text{Matrices})]$ ， $s[i, j]$  存入  $m[i, j]$  中最小值對應的  $k$ 。
- 理解： $m[i, k]$  為拆分的前部分， $m[k+1, j]$  為拆分的後部分， $p_{i-1}p_kp_j$  為前後部分相乘。

#### 6. Theorem (111) Optimal Binary Search Tree (OBST):

- $$e[i, j] = \begin{cases} q_{i-1} & , j = i - 1 \\ \min_{i \leq r \leq j} \{e[i, r-1] + e[r+1, j] + w[i, j]\} & , i \leq j \end{cases} \quad (5)$$
- $w[i, j] = w[i, j-1] + p_j + q_j$

其中， $p_j$  為 key（內部節點）機率， $q_j$  為 dummy key（外部節點）機率。

- $w[1 \cdots \text{Number}(\text{Key}) + 1][0 \cdots \text{Number}(\text{Key})]$ ，初始化對角線上元素  $w[j+1, j]$  為  $q_j$ 。
- $e[1 \cdots \text{Number}(\text{Key}) + 1][0 \cdots \text{Number}(\text{Key})]$ ，初始化對角線上元素  $e[j+1, j]$  為  $q_j$ 。
- $r[1 \cdots \text{Number}(\text{Key})][1 \cdots \text{Number}(\text{Key})]$ ， $r[i, j]$  存入  $e[i, j]$  中最小值對應的  $r$ 。
- 理解： $e[i, r-1]$  為左子樹， $e[r+1, j]$  為右子樹， $w[i, j]$  為節點權重和，因為計算 cost 時是節點階層加一。

**7. Theorem (76, 78, 84, 91, 93, 98, 109, 115, 335)**

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	$\Theta(n^2 2^n)$	$O(n2^n)$	
Optimal Binary Search Tree (OBST)	$\Theta(n^3)$	$\Theta(n^2)$	

**8. Theorem (149)**

- 團 (clique): 任兩點皆有邊相連, 即完全子圖。
- 獨立集 (independent set): 獨立集中任兩點無邊相連, 補圖的團。
- 支配集 (dominating set): 圖中支配集外的點皆與支配集相連。
- 點覆蓋 (vertex cover): 點覆蓋中的點為圖中所有邊的端點。

**9. Theorem (159) Kosaraju's algorithm:**

- 找 strongly connected component。
- 步驟:
  - 對原圖做 DFS。
  - 從結束時間最晚者開始, 對反向圖做 DFS。

**10. Theorem (171, 178, 183, 193, 195) Shortest path:**

- Floyd-Warshall: sparse 時, 也不能提升性能。
- Johnson's 在 sparse 時, 性能較 Floyd-Warshall 好; Reweight 後圖上所有邊權重皆  $> 0$ , 且最短路徑與原圖相同。
- Bellman-Ford:

$$D[v, k] = \min\{D[v, k-1], \min_{(u,v) \in E} \{D[u, k-1] + wt(u, v)\}\} \quad (6)$$

- Floyd-Warshall:

$$D^k[i, j] = \min\{D^{k-1}[i, j], D^{k-1}[i, k] + D^{k-1}[k, j]\} \quad (7)$$

# 11. Theorem (159, 172, 179, 183, 193, 204, 206)

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	$O( V  E ^2)$	

# 12. Theorem ()

Type of common problems	
Problem	Type
Longest path problem (graph)	NPC
Longest path problem (tree)	Linear
Minimum vertex cover (graph)	NPC
Minimum vertex cover (tree)	Linear
Max-cut	NPC
Euler circuit	P
Hamiltonian path	NPC

# 13. Theorem (363) Maximal points:

---

```

1: function MAXIMALPOINTS(Point [] points)
2:   s :=  $\emptyset$ 
3:   Sort points by x-coordinate in ascend order.
4:   max_y :=  $-\infty$ 
5:   for i := n to 1 do
6:     if points[i].y > max_y then
7:       Add points[i] to s.
8:       max_y := points[i].y
9:     end if
10:  end for
11:  return s
12: end function

```

---

14. Theorem (236, 240, 241, 245)

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	$\Theta(n \log n)$

15. Theorem (262, 265, 285)

- 所有 NP 問題都能多項式時間 reduce 到 NP-Hard。
- 證明 NPC: 問題屬於 NP; 已知 NPC 可以多項式時間 reduce 到該問題, 即證明該問題是 NP-Hard。
- 如果可以證明 **lower bound of worst case** of NPC problems is polynomial, 則  $P = NP$ 。

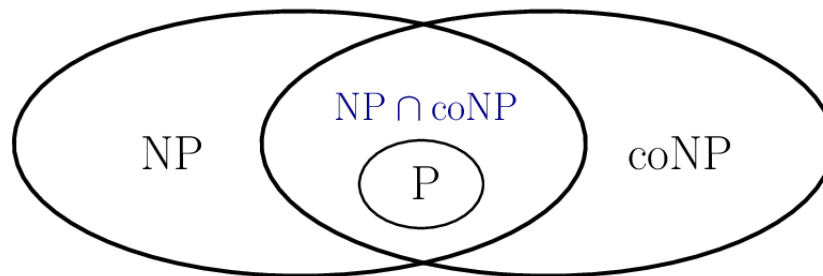


图 1: Relationship between NP and CO-NP.



## 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] wjungle@ptt. 演算法 @tkb 筆記. <https://drive.google.com/file/d/0B8-2o6L73Q2VVmNWQk9DY3hsUm8/view?usp=sharing>, 2017.

