演算法

Algorithm

TZU-CHUN HSU[1]

[1]vm3y3rmp40719@gmail.com

[1]Department of Computer Science, Zhejiang University

2021 年 1 月 28 日

Version 4.0

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 LaTeX 相關程式碼採用 **MIT 協議**，更多內容請詰訪問作者之 GitHub 分頁Oscarshu0719。

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	$\star\star\star\star$
2	13	$\star\star\star\star$
3	18	$\star\star\star\star\star$
4	34	$\star\star\star\star\star$
5	43	$\star\star\star$
6	48	$\star\star\star$
7	$\times$	$\star$
8	$\times$	$\star\star\star$

3. 省略第 7 章。

Dynamic Programming algorithms		
Problem	Time complexity	Space complexity
Making change	$O(kn)$	$O(n)$
Fractional Knapsack problem	$\Theta(n \log n)$	$O(n)$
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)	$O(mn)$	$O(mn)$
Longest Increasing Subsequence (LIS)	$O(n^2)$	$O(n^2)$
Longest Common Substring	$O(mn)$	$O(mn)$
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)	$\Theta(n^3)$	$\Theta(n^2)$

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 (Adjacency list)	$O(\|V\|\|E\|)$	
Prim's (Min-Heap, Adjacency list)	$O(\|E\| \log \|V\|)$	
Prim's (Fibonacci heap, Adjacency list)	$O(\|E\| + \|V\| \log \|V\|)$	
Sollin's (Borůvka's)	$O(\|E\| \log \|V\|)$	
Dijkstra's (Min-heap)	$\Theta((\|E\| + \|V\|) \log \|V\|)$	Greedy, no negative edges or cycles
Dijkstra's (Fibonacci-heap)	$\Theta(\|E\| + \|V\| \log \|V\|)$	
Bellman-Ford	$O(\|V\|\|E\|)$	DP
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	$O(\|V\|\|E\|^2)$	
Push-relabel	$O(\|V\|^2\|E\|)$	

2 Summary

1. **Theorem (100)** Matrix-chain Multiplication：

   - $$m[i,j] = \begin{cases} 0 & ,i=j \\ \min_{i\le k\le j-1}\{m[i,k]+m[k+1,j]+p_{i-1}p_kp_j\} & ,i<j \end{cases} \tag{1}$$
   - $p[0\cdots \text{Number(Matrices)}]$，存入矩陣大小。
   - $m[1\cdots \text{Number(Matrices)}][1\cdots \text{Number(Matrices)}]$，初始化對角線上元素為 0。
   - $s[1\cdots \text{Number(Matrices)}-1][2\cdots \text{Number(Matrices)}]$，$s[i,j]$ 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。

2. **Theorem (111)** Optimal Binary Search Tree (OBST)：

   - $$e[i,j] = \begin{cases} q_{i-1} & ,j=i-1 \\ \min_{i\le r\le j}\{e[i,r-1]+e[r+1,j]+w[i,j]\} & ,i\le j \end{cases} \tag{2}$$
     $$w[i,j]=w[i,j-1]+p_j+q_j$$
     其中，p_j 為 key（內部節點）機率，q_j 為 dummy key（外部節點）機率。
   - $w[1\cdots \text{Number(Key)}+1][0\cdots \text{Number(Key)}]$，初始化對角線上元素 $w[j+1,j]$ 為 q_j。
   - $e[1\cdots \text{Number(Key)}+1][0\cdots \text{Number(Key)}]$，初始化對角線上元素 $e[j+1,j]$ 為 q_j。
   - $r[1\cdots \text{Number(Key)}][1\cdots \text{Number(Key)}]$，$r[i,j]$ 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。

3. **Theorem ()** Minimum vertex cover (tree)：

   $$\begin{aligned} V(v) = \min\{&1+\text{sum}\{V(c),\ \forall c\in v.child\}, \\ &\text{Length}\{v.child\}+\text{sum}\{V(g),\ \forall c\in v.child\,\forall g\in c.child\}\} \end{aligned} \tag{3}$$

   First part: root is in the cover; second part: root is NOT in the cover.

4. **Theorem ()** Max-cut：
   - NPC。
   - 若所有邊權重皆負，則可乘上 -1，變為 Min-cut。
   - 若為平面圖，可轉換為 Chinese Postman Problem（若為無向圖，即 Euler circuit，若為有向圖，則為 NPC）。

5. **Theorem (285)**
   - 如果可以證明 **lower bound** of **worst case** of NPC problems is polynomial，則 $P = NP$。

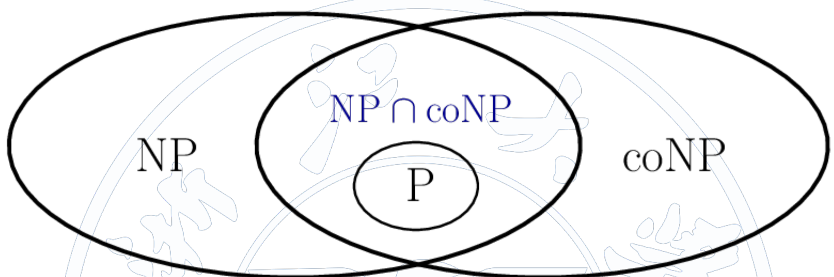

图 1: Relationship between NP and CO-NP.

6. **Theorem ()**
   - (**FALSE**) For two functions $f(n)$ and $g(n)$, either $f(n) = O(g(n))$ or $f(n) = \Omega((f(n))$. Counterexample:

$$
\begin{aligned}
f(n) &= \begin{cases} 1, & \text{if } n = 2k \\ 0, & \text{if } n = 2k+1 \end{cases} \\
g(n) &= \begin{cases} 0, & \text{if } n = 2k \\ 1, & \text{if } n = 2k+1 \end{cases}
\end{aligned} \tag{4}
$$

   - For any uniform cost RAM program $T(n) = \Omega(S(n))$, where $S(n)$ is the space an algorithm uses for an input of size n.
   - The capacity of each edge of a flow network can be floating-point, and it can be solved by linear programming.
   - A flow network of multiple sources can be reduced to a single source.
   - (**FALSE**) The value of any flow of a flow network is bounded by the capacity of only at most $O(n)$ cuts.

- 2-coloring: $O(n^2)$, 3-coloring, 4-coloring: superpolynomial.
- Weighted-union heuristic: Append the **smaller** list onto the **longer** list, with ties broken arbitrarily.
- $n! \neq \Theta(n^n)$.
- A DAG with n vertices can **NOT** have more than $\binom{n}{2}$ edges.
- Longest palindrome subsequence:

$$L(i,j) = \begin{cases} 0 & ,i = j+1 \\ 1 & ,i = j \\ L(i+1, j-1) + 2 & ,i < j \wedge s[i] = s[j] \\ \max(L(i+1,j), L(j,j-1)) & ,\text{otherwise} \end{cases} \tag{5}$$

where $L[1 \cdots n][1 \cdots n], s[1 \cdots n]$

- Minimum triangulation:

$$c(i,j) = \begin{cases} 0 & ,j < i+2 \\ \min_{i<k<j}\{c(i,k) + c(k,j) + dist(i,j) + dis(j,k) + dist(k,j)\} & ,\text{otherwise} \end{cases} \tag{6}$$

```
double triangulation(Point P[], int n) {
    if (n < 3)
        return 0;

    double c[n][n];
    for (int gap = 0, gap < n; gap++) {
        for(int i = 0, j = gap; j < n; i++, j++) {
            if (j < i + 2)
                c[i][j] = 0.0;
            else {
                c[i][j] = MAX;
                for (int k = i + 1; k < j; k++) {
                    double val = c[i][k] + c[k][j] + wt(
                        P, i, j, k);
                    if (c[i][j] > val)
                        c[i][j] = val;
                }
            }
```

```
            }
        }

        return c[0][n - 1];
    }
```

Listing 1: Minimum triangulation.

- Sort n integers ranged from 0 to $n^2 - 1$: 將 n 個整數表示成 **n 進位**數，每個數由 2-digit 表示，範圍 0 到 $n-1$，再用 radix sort 對 2-digit 排序，共兩次。
- If max frequency is ≤ 2 times of min frequency, Huffman code is **NOT** always better than an ordinary fixed-length code.
- Amortized analysis 與 average-case analysis 無關。
- (**FALSE**) **If** a graph has a unique MST then, for every cut of the graph, there is a **unique light edge** crossing the cut.
- (**TRUE**) A graph has a unique MST **if**, for every cut of the graph, there is a **unique light edge** crossing the cut.
- The worst-case running time and expected running time are equal to within **constant** factors for any randomized algorithm.
- Selection problem: $T(n) = T(\frac{n}{5}) + T(\frac{3n}{4}) + O(n)$
- Given an **undirected** graph and a positive integer k, is there a path of length $\leq k$, which each edge has weight 1 and each vertex is visited **exactly** once: P, solved by Floyd-Warshall algorithm.
- Given an **undirected** graph and a positive integer k, is there a path of length $\geq k$, which each edge has weight 1 and each vertex is visited $\leq$ once: NPC.
- A flow network of multiple sources can be reduced to a single source.
- Subset sum: $s(i, j)$: sum j can be found in $\{a_1, \cdots, a_i\}$

$$s(i,j) = \begin{cases} 0 & , i = 0 \\ 1 & , j = 0 \\ s(i-1, j) \vee s(i-1, j-v_i) & , j \geq v_i \end{cases} \tag{7}$$

result is $s(m, n)$.

- Hanoi tower iterative version: Check if the **input number** n is even or odd.

  If n is even,

$$\begin{cases} A \leftrightarrow C \\ A \leftrightarrow B \\ C \leftrightarrow B \end{cases} \tag{8}$$

  If n is odd,

$$\begin{cases} A \leftrightarrow B \\ A \leftrightarrow C \\ B \leftrightarrow C \end{cases} \tag{9}$$

- Fibonacci search:

```python
def fibSearch(arr, data):
    max = len(arr) - 1
    y = getY(fib, max + 1) # Find the largest index,
        which its value is smaller than data.
    m = max - fib[y]
    x = y - 1
    i = x
    if arr[i] < data: # Check at first.
        i += m
    while fib[x] > 0:
        if arr[i] < data:
            x -= 1
            i += fib[x]
        elif arr[i] > data:
            x -= 1
            i -= fib[x]
        else:
            return i
    return -1
```

Listing 2: Fibonacci search.

- Box stacking:
  (a) Generate all 3 rotations of all boxes. We consider width as always smaller than or equal to depth.
  (b) Sort the above generated $3n$ boxes in **decreasing** order of **base area**.

(c) $msh(i)$: Max possible stack height with box i at top of stack.

$$\begin{aligned} msh(i) = \{ & \max\{msh(j)\} + height(i)\}, \\ & \forall j < i \wedge width(j) > width(i) \wedge depth(j) > depth(i) \end{aligned} \tag{10}$$

result is

$$\max_{0<i<n}\{msh(i)\} \tag{11}$$

- Building bridge:
  (a) Sort the north-south pairs on the basis of **increasing** order of **south** x-coordinates.
  (b) Find **LIS** of north x-coordinates.
- Optimal strategy: $f(i, j)$: max value the user can collect from i-th coin to j-th coin.

$$f(i,j) = \begin{cases} v_i & , j = i \\ \max\{v_i, v_j\} & , j = i+1 \\ \max\{v_i + \min\{f(i+2,j), f(i+1,j-1)\}, \\ \quad v_j + \min\{f(i+1,j-1), f(i,j-2)\}\} & \text{, otherwise} \end{cases} \tag{12}$$

- (TIOJ-1097) Find the largest square submatrix with all 0s in a 0/1 matrix: $dp(i, j)$: max square submatrix in $i \times j$ left upper submatrix.

$$dp(i,j) = \min\{dp(i-1,j-1), dp(i,j-1), dp(i-1,j)\} + 1 \tag{13}$$

- (UVA-10934) Dropping water balloons (k balloons and height n): $dp(i, j)$: max height i balloons can be dropped j times.

$$dp(i,j) = \begin{cases} dp(i,j-1) + dp(i-1,j-1) + 1 & , arr(i,j) = 1 \\ 0 & , arr(i,j) = 0 \end{cases} \tag{14}$$

result is

$$\min_j\{dp(k,j) \geq n\} \tag{15}$$

- (TIOJ-1471) Skyline: $dp(i, j)$: walk i distance and height is j.

$$\begin{cases} dp(i,j) & = dp(i-1,j-1) + sum(j) \\ sum(j) & = sum(j) - dp(i-j,j) + dp(i,j) \end{cases} \tag{16}$$

result is

$$\sum_j dp(n,j) \tag{17}$$

- Largest rectangle in histogram:
  - If the new element is higher than stack top element, push it; otherwise, pop and calculate the area until the new element is higher than stack top element.
  - Maximal rectangle: Similarly, for each column, the count of 1 of each row, can be seen as the element.

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.