主題: Branch-and-bound (II)

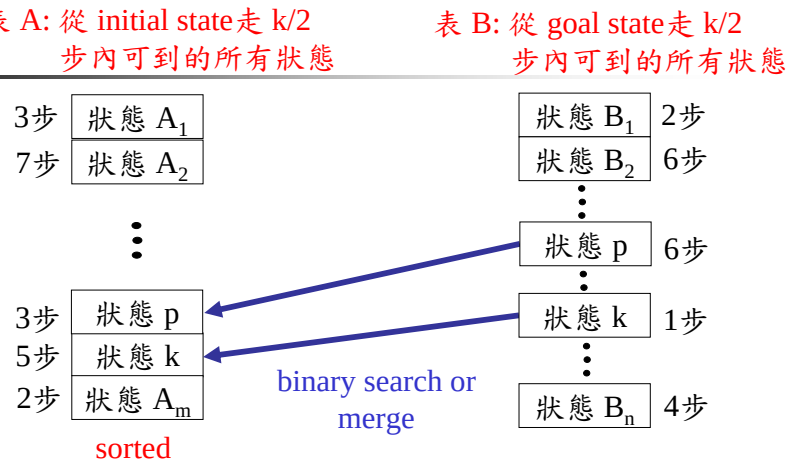
- 基礎
- 應用
- 作業與自我挑戰

基礎

- 2-way BFS
- memoized BFS
- BFS + DFS

2-way BFS

- 適用於已知 initial state、goal state、以及步數限制 k ，且可以由雙方各走 k/2 步的情況 (移動為可逆)
  - 先從 initial state BFS 走 k/2 步，建表 A 記住可以到達的狀態 (重覆的狀態只留下步數最小者)
  - 同樣的，從 goal state 走 k/2 步，建表 B
  - 找出 A, B 中的相同狀態 (有交集代表能在 k 步內走到)，找出總和最小的步數



- Another implementation: hashing

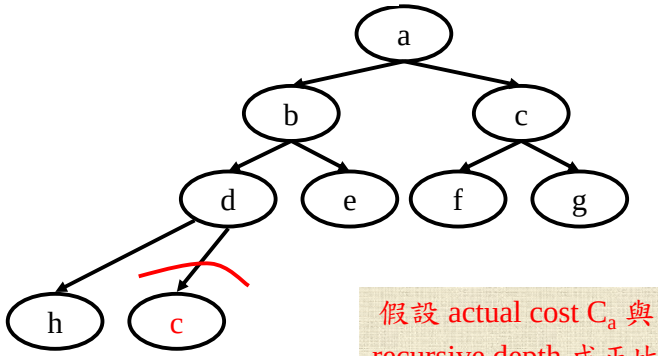
Memoized BFS

- 在執行 BFS 的過程中，用一個 data structure S 將所有見到的 states 存起來 (建議用 hash table)
- 假設 actual cost C_a 與 recursive depth 成正比
- 當一個 state s 產生時
  - 先 check s 是否已經在 S 中 (重複出現)
  - 如果是則扔掉 s 避免重複展開 s 下面的 sub-tree

註: 先出現的 depth 小，所以 actual cost 比較好

Example

S = {a, b, c, d, e, f, g} ∪ { h}



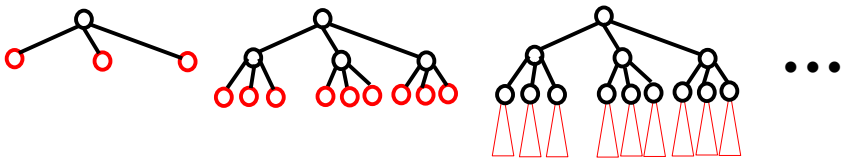
cut this subtree

假設 actual cost C_a 與 recursive depth 成正比

BFS + DFS

- For (i = 1, 2, ...,), do i-level DFS until a solution is found

1-level DFS　2-level DFS　3-level DFS



- do not need to maintain a queue
- smaller storage

BFS + DFS 也適用於預期 solution 不會太遠，但是用 DFS 又很可能會在找到第一個好的 bound 前，進入很深的 recursive ，甚至會不小心掉進 infinite recursion

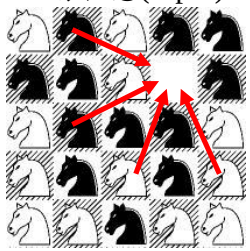
應用

- 應用一: A.10422 Knights in FEN
- 應用二: A.652 Eight (revisit)
- 應用三: A.2240 A Vexing Problem
- 應用四: H.91.2 旅行支票之兌換

應用一: A.10422 Knights in FEN

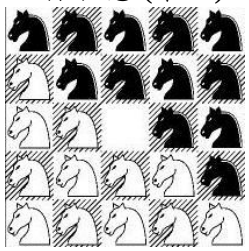
- 給一個 5 * 5 的西洋棋盤，與各 12 隻的黑白騎士，並給予目標的擺法與現在的擺法，利用空白格來移動騎士，請求出不超過 11 步的最少走法

目前狀態(Input)



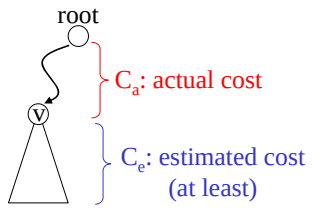
11步之內

目標狀態 (唯一)



Solutions

- BFS on the state graph: ~ 25×2^{24} states (impossible)
  $(25\times C(24, 12)) \approx 10^8$
- brute-force: $O(8\times7^{10}) \approx 10^{10}$ (impossible)
- B&B: (may not be good enough)
  - Bound: the current best b (b = 12 initially)
  - C_a: 目前實際移動步數
  - C_e: ???
- 2-way BFS: $O(8^6)$



應用二: A.652 Eight (revisit)

- 一個 3×3 的魔術盤，上面有 1 到 8 的數字與一個空格，空格可以和上下左右的數字交換。給定起始狀態，請問最少要移動幾次才能從起始狀態變成目標狀態？

2	8	3
1	6	4
7		5

起始狀態

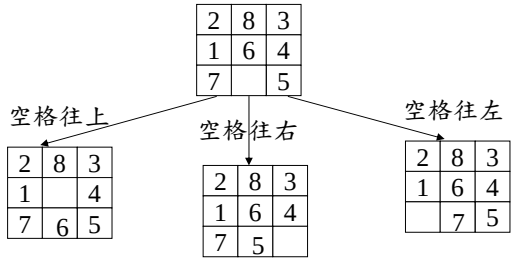
最少幾步?

1	2	3
4	5	6
7	8	

目標狀態

Solution I: (Graph) BFS

- 僅有 9! ($\approx 10^5$) 種 states，可用 (graph) BFS

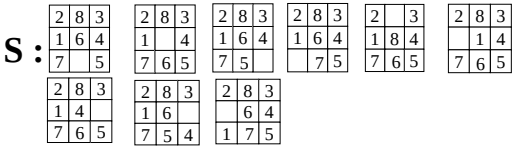


- 困難: 如何用 array 來表式這 9! states?
  - 不能直接數字作為 id ($\approx 10^9$)
  - state 和 id 如何 mapping?
- 解決方法: relabeling (請參考 BFS 講義)

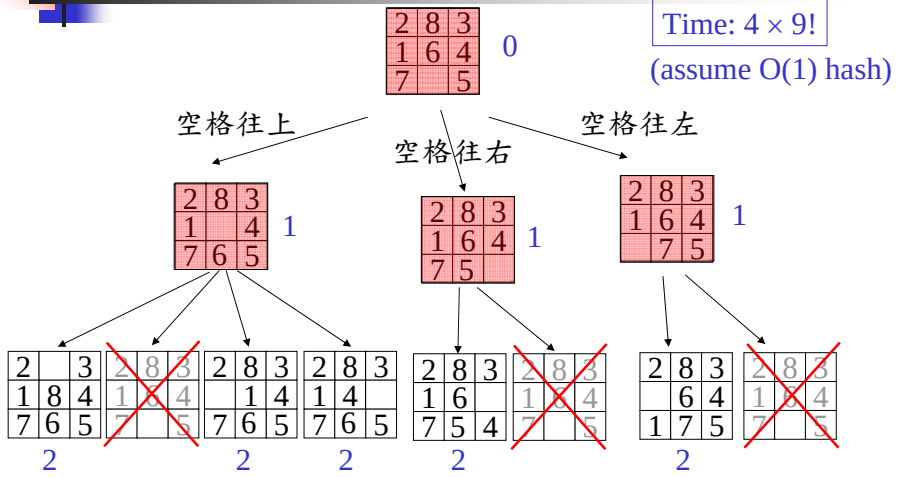
Solution II: Memoized BFS

- 從起始狀態執行 memoized BFS
- 當一個 state s 產生時
  - 先 check s 是否已經在 S 中 (重複出現)
  - 如果是則扔掉 s 避免重複展開 s 下面的 sub-tree

Example

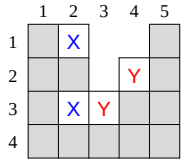


Time: 4 × 9!
(assume O(1) hash)



應用三：A.2240 A Vexing Problem

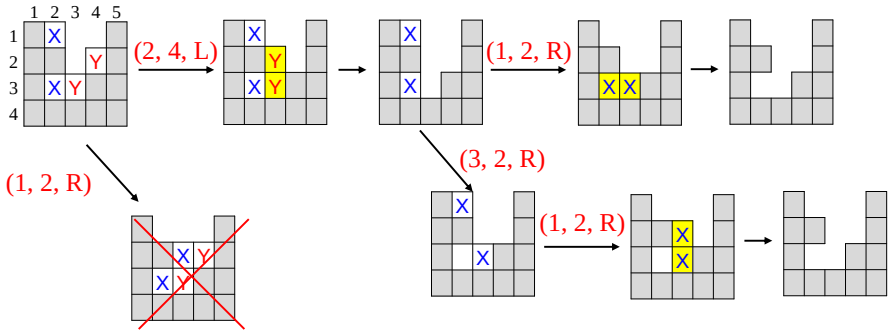
- 一個 M × N (M, N ≤ 9) 的盤子，灰色部分是不可移動的牆壁 (兩邊、底部為牆壁)，其他方塊上有大寫字母，可以移動一步到左右空格上。



- 如果方塊下方沒有其他方塊或牆壁支撐：
  - 方塊會往下降
  - 當沒有方塊降落時，相鄰的方塊有相同文字就可以消掉
- 問題：消掉所有方塊最少需要移動幾步？
  (input 保證不超過 11 步)

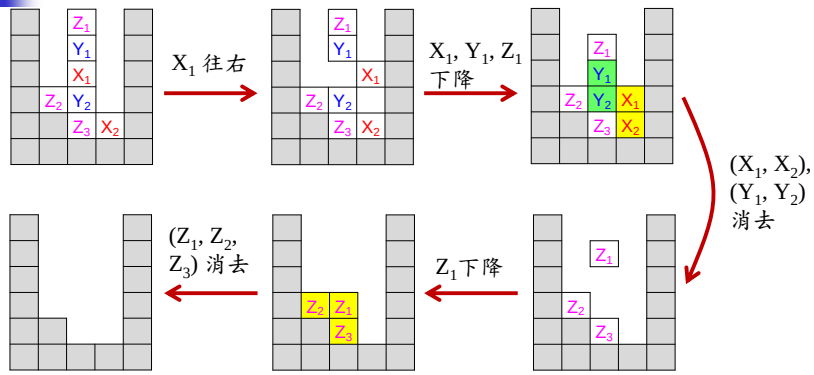
Example 1

移動步數：2



移動步數：3

Example 2



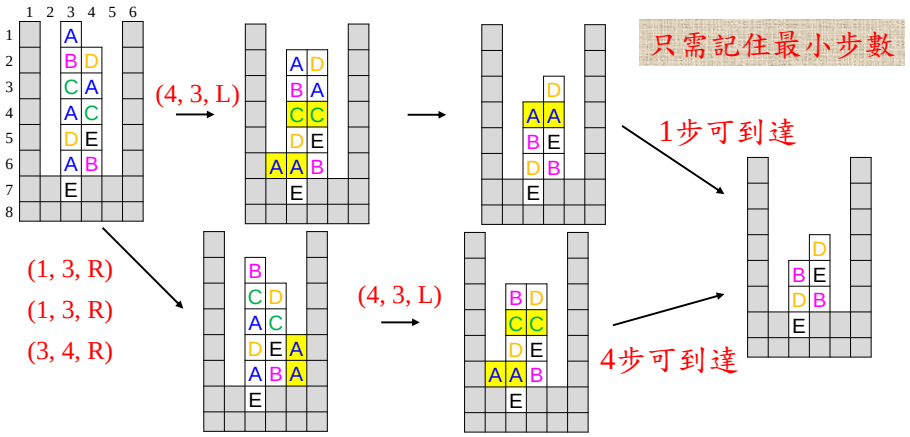
移動步數：1 (最少)

Solution I: B&B

- brute-force: $O((7\times8\times2)^{11})$ (impossible)
- B&B:
  - Bound: the current best b (b = 12 initially)
  - C_a: 目前實際移動步數
  - C_e: ???
  - may not be good enough

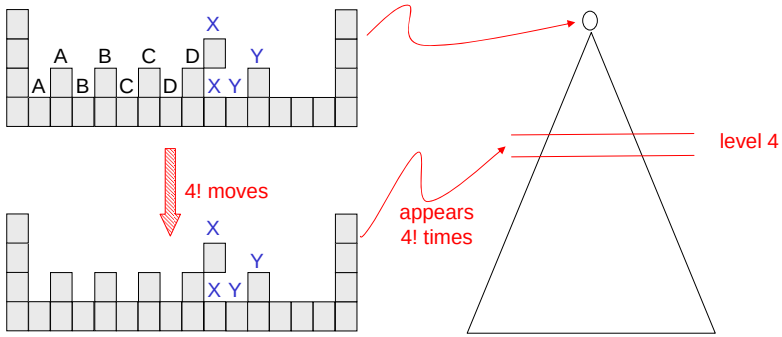
Observation

- 同一個盤面可以由多個不同的移動方式產生



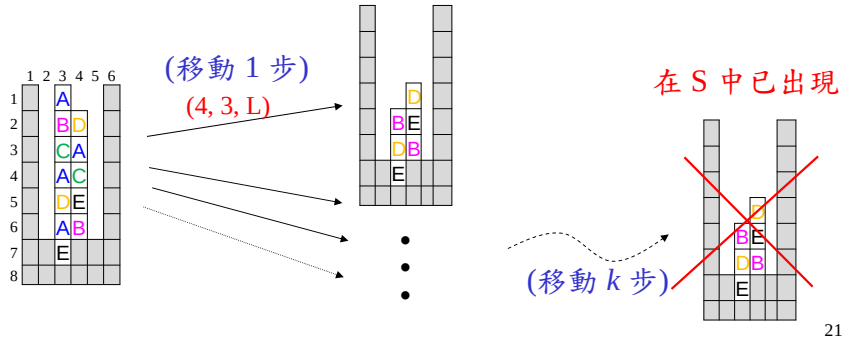
Observation

- 同一個盤面可以由多個不同的移動方式產生



Solution II: Memoized BFS

- 從起始狀態執行 memoized BFS
- 當一個 state s 產生時
  - 先 check s 是否已經在 S 中 (重複出現)
  - 如果是則扔掉 s 避免重複展開 s 下面的 sub-tree

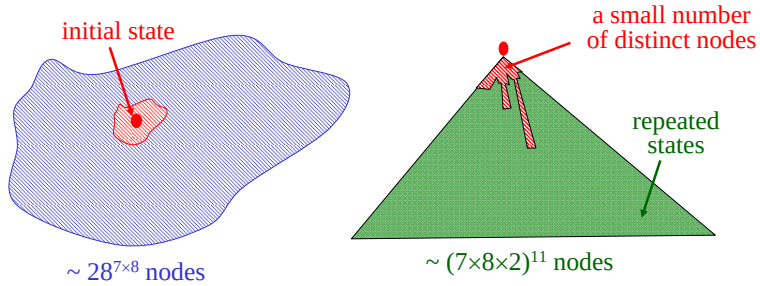


Usages of Memoized BFS

- Usage 0: avoid relabeling in graph BFS
- Usage 1: make brute-force BFS more efficient
  - A speedup technique (like B&B)
  - may be combined with B&B

Time for Usage 1

- It is expected that the number of reachable nodes is small
- However, the search tree is of huge size
  - a state may appear many times (or loops of states)



應用四: H.91.2 旅行支票之兌換

http://www.cs.tku.edu.tw/info_race2002/codeq.htm

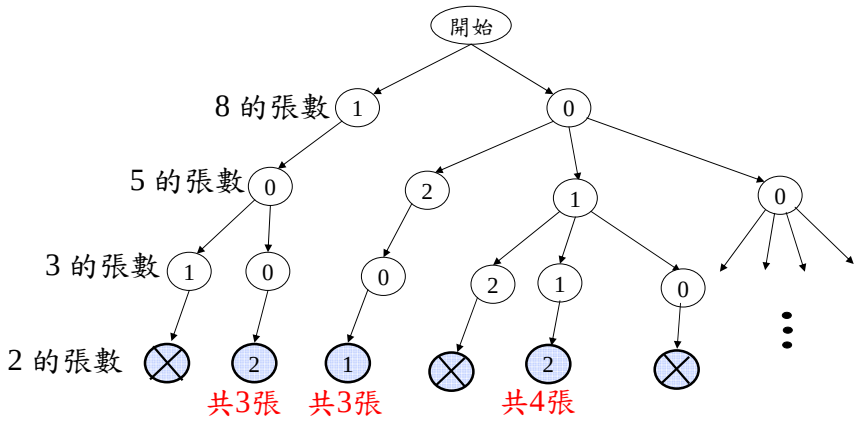
- 給 k 種鈔票的面額 A={a_1, a_2, ..., a_k}，再給定一個目標數 N，找出所有能湊成 N 且張數最少的方法
- $k \le 30$ (range of N: ???, range of a_i: ???)
- Example: A = {2, 3, 5, 8}, N = 12

2 + 2 + 8 = 12 ⇒ 3 張
3 + 3 + 3 + 3 = 12 ⇒ 4 張
2 + 5 + 5 = 12 ⇒ 3 張
…

3 張:
2 + 2 + 8
2 + 5 + 5

A brute-force solution

Example: N =12 , A = {2, 3, 5, 8}



A B&B solution

- for very small k (guess that k is very small in all test cases)
- Bound: 已知最少的張數 b
  - C_a: 已經使用張數
  - C_e: ???
- Problem: How to report all solutions ???

Report all best solutions

- 方法一: 記錄更新
  - ~~5 → (30 + 30 + 20 + 20 + 20) → (80 + 10 + 10 + 10 + 10)~~
  - ~~3 → (100 + 10 + 10) → (80 + 30 + 10) → (110 + 5 + 5)~~
  - 2 ← 100 + 20
- 方法二: 兩次搜尋法
  - 先執行一次程式求出最少張數 x (= 2)
  - 再執行一次程式，這次所有符合張數 x 的換法即是答案 (用第一次所得的 x 作為 initial bound)

作業與自我挑戰

- 作業
  - 練習題
    - A.10422 Knights in FEN
      http://uva.onlinejudge.org/external/104/10422.html
  - 挑戰題
    - A.704 Colour Hash (Hint: 2-way BFS)
      http://uva.onlinejudge.org/external/7/704.html
- 自我挑戰
  - A.2240 A Vexing Problem (Hint: memoized BFS)
  - A.10274 Fans and Gems (Hint: memoized BFS)
  - A.165 Stamps (DP + B&B)
  - A.166 Making Changes
  - A.10384 The Wall Pushers (Hint: BFS+DFS)
  - A.10447 Sum-up the Primes (II)
- 其它有趣題目:
  - AF2004D Insecure in Prague
  - A.10419 Sum-up the Primes