主題: Branch-and-bound (I)

- 基礎
- 應用
- 作業與自我挑戰

基礎

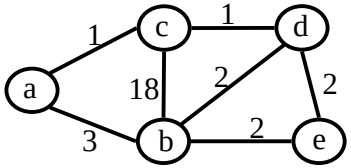
- Brute-force search
- Branch-and-bound

Brute-force Search

- 把所有的可能都產生出來，加以檢查後找出答案
- 最簡單也是最不得已的最後手段

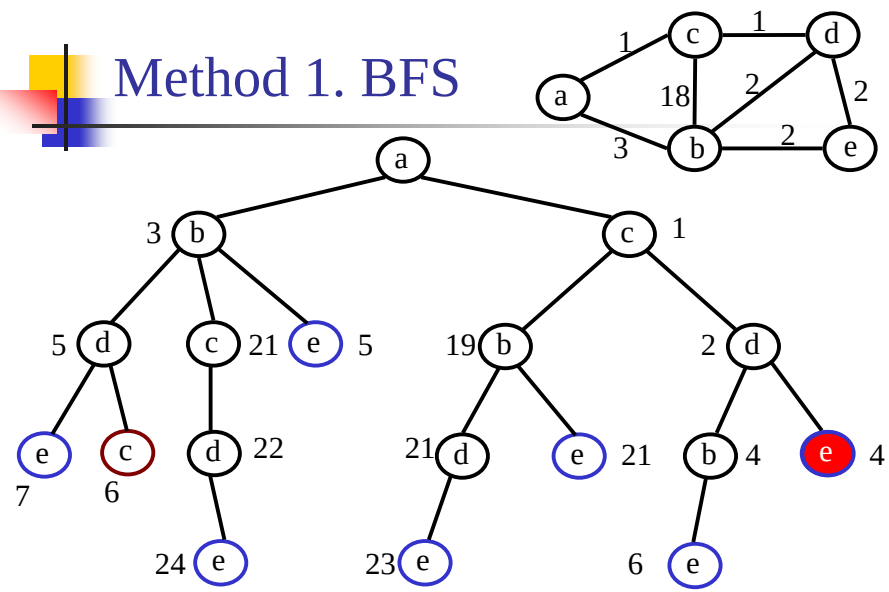
Example

Find a shortest path from a to e

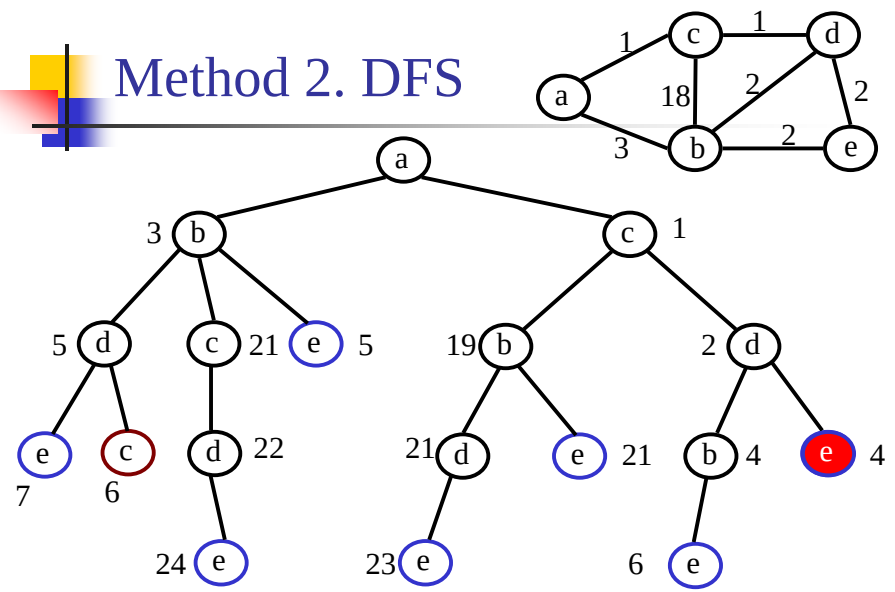


- Brute-force solution: produce all simple paths from a to e and find the best one
- Problem: How to produce all simple paths ?

Method 1. BFS



Method 2. DFS

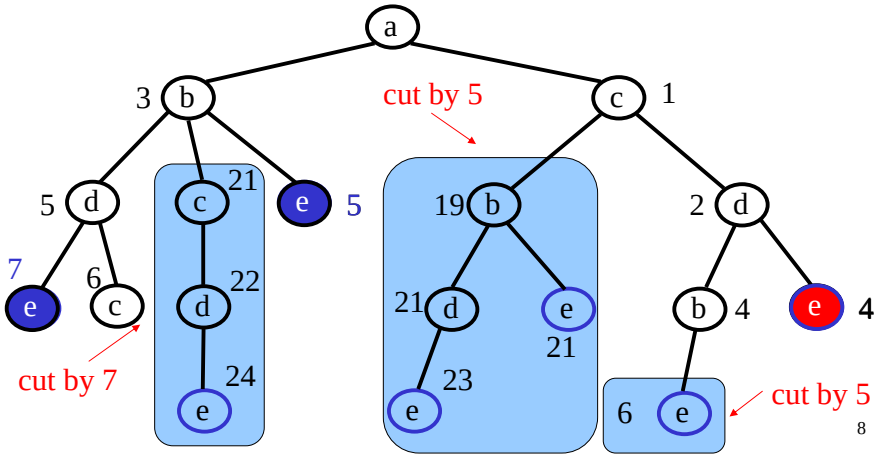


BFS vs. DFS

- DFS
  - 通常使用 DFS，因為比較簡單好寫
    - recursive 的 stack 由系統提供
  - smaller storage
- When DFS needs a stack of size > 10^6
  - write a non-recursive version (maintain a stack by yourself)
  - use BFS

Branch-and-bound

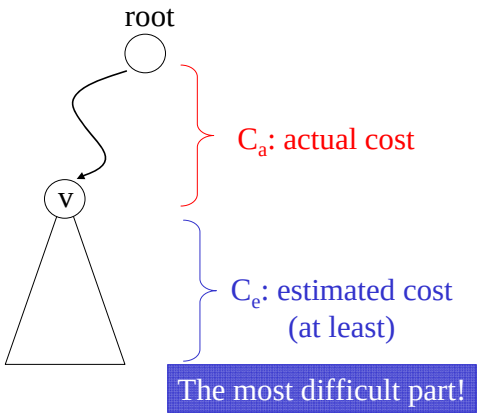
- Branch-and-bound: Brute-force + intelligent cuts



The idea of branch and bound

(**for minimization**)

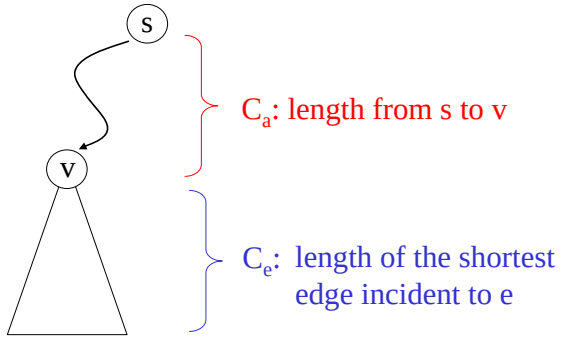
- b: current best
  (initially, b = ∞)

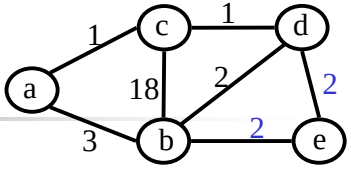


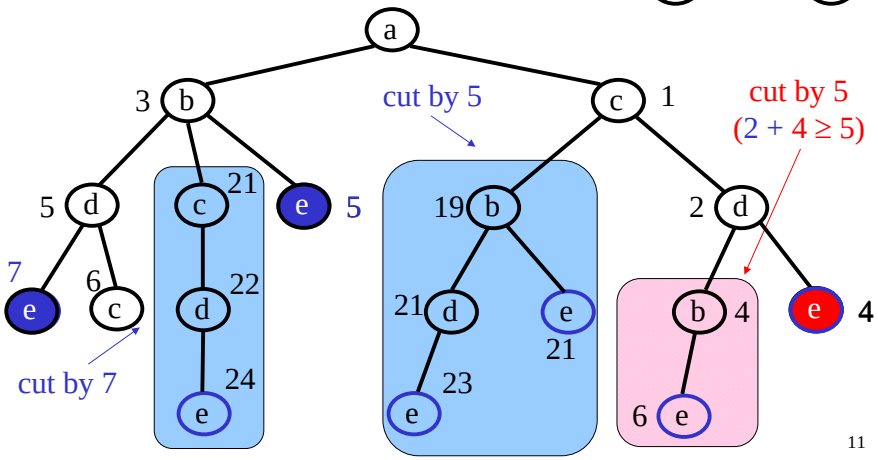
- backtrack at v if
  - v is a leaf, or
  - $C_e + C_a \geq b$
- if a better solution is found, replace b with it.

Example

- Finding the shortest path from s to e

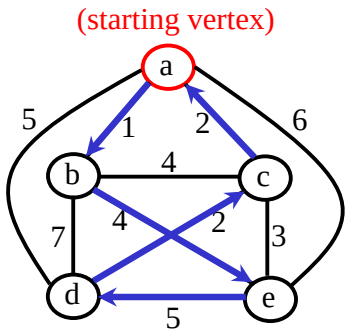






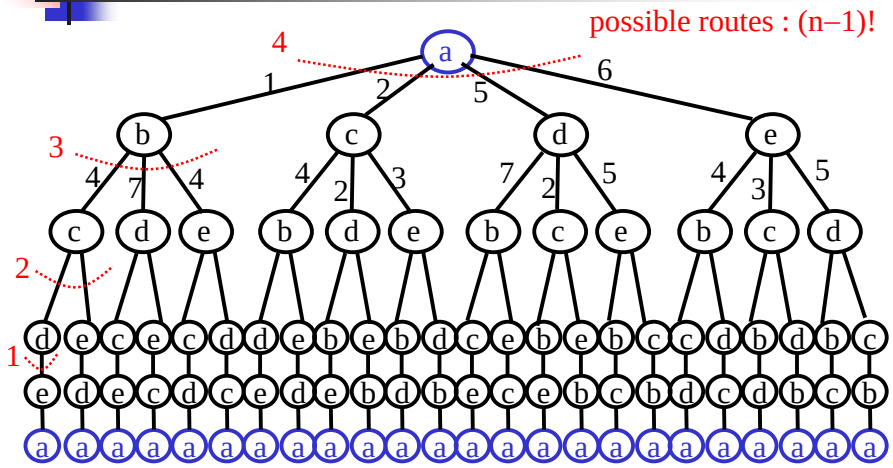
Example: TSP

- Find a shortest route which visits every vertex exactly once and returns to the starting vertex.

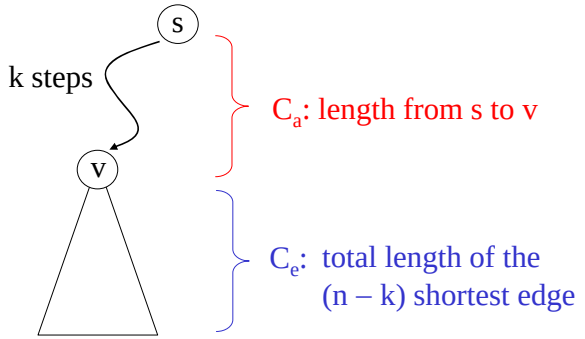


length of shortest route: 14

Brute-force search



A branch-and-bound solution

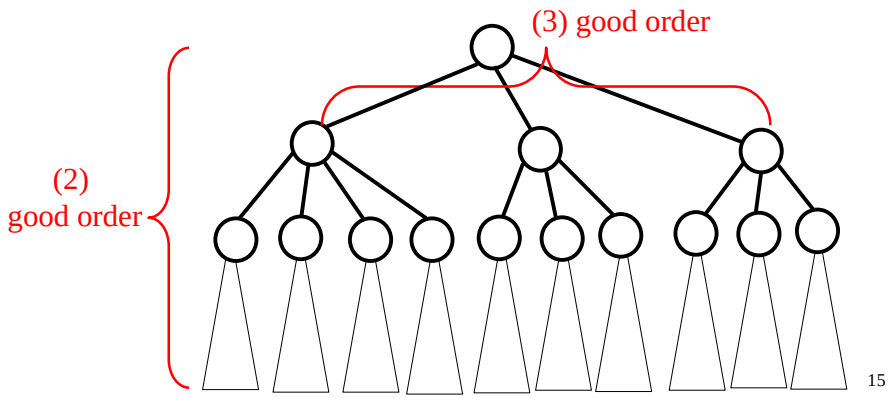


- (1, 2, 2, 3, 4, 4, 5, 5, 6, 7): C_e = **5** for k = 2 (or n - k = 3)
- (1, 3, 5, 8, 12, 16, 21, 26, 32, 59): prefix sums

Speedup by using heuristics

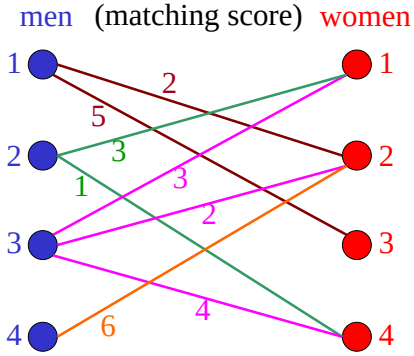
- Three places

(1) get a good initial bound b (instead of b = ∞).



Example: Optimal matching

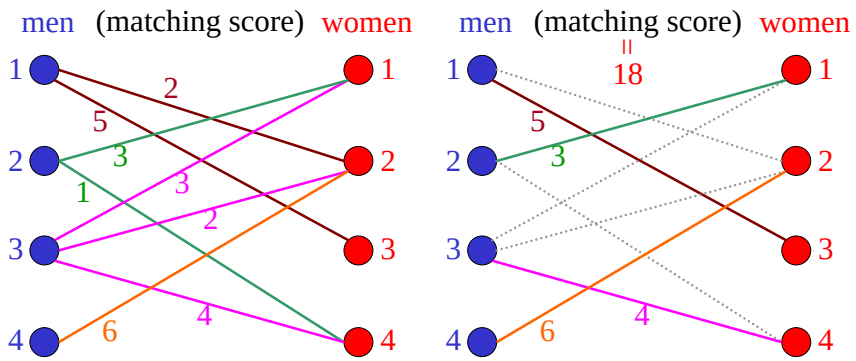
- **Input**: A[i, j]: the weight of matching man i and woman j



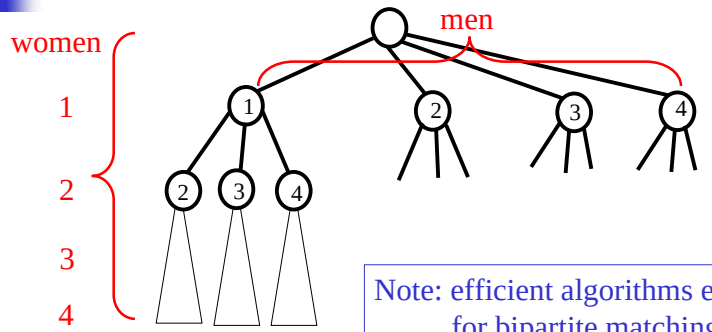
	1	2	3	4
1	0	2	5	0
2	3	0	0	1
3	3	2	0	4
4	0	6	0	0

Example: Optimal matching

- **Problem**: Find a matching that maximizes the matching scores.



Example: Optimal matching



Note: efficient algorithms exist for bipartite matching

- (1): get a good b initially
- (2), (3): sort men and women according to popularity
- C_e: ???

When to use brute-force search?

- 題目要求「列出所有的可能解」
- Problem size 很小，檢查所有可能的時間不會很長
- 國內辦的比賽每一題都可以試試看，因為 problem size 大多騙人，test case 通常很小
  (即使題目上說 n 是 infinite ???!!!)

When to use branch-and-bound?

- Optimization problems
- 除暴力法，想不出任何方法
- 單純暴力法時間一定會超過

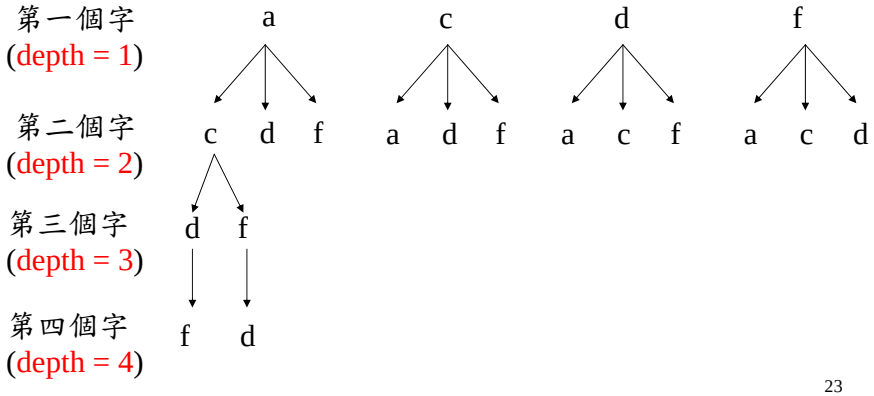
應用

- 應用一: A.10098 Generating Fast, Sorted Permutation
- 應用二: A.441 Lotto
- 應用三: A.167 The Sultan's Successors
- 應用四: H.91.6 專題選課
- 應用五: A. 10318 Security panel
- 應用六: 整數的分割方式
- 應用七: A.574 Sum it Up

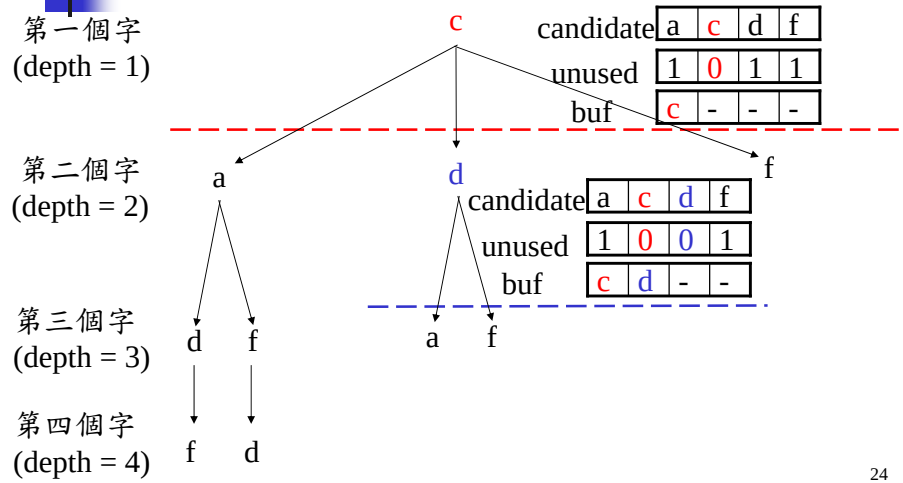
應用一: A. 10098 Generating Fast, Sorted Permutation

- 給 n 個英文字母，請條列出所有能由這 n 個英文字母排列成的字串
- 以 lexicographical 順序輸出
  - 例: acfd (n = 4)
    acdf ⇒ acfd ⇒ adcf ⇒ adfc ⇒ afcd ⇒ afdc ⇒ …
- n ≤ 10
  - 有 n! ≈ 10^6 組解
- Solution: brute-force (recursive DFS)

Example: acdf (sorted)



Example: acdf



Pseudo code

candidate	a	c	d	f
unused	1	1	1	1
buf	-	-	-	-

```
void perm(int depth)        ← recursive depth=1, 2, ..., n
{
   int i;
   for (i = 0; i < n; i++)      ← 按字母順序加入每一個可能的字母
      if (unused[i] == 1) {     ← 檢查這個字母是否還沒有被使用過
         buf[depth – 1] = candidate[i];  ← 把字母加到目前的解中
         unused[i] = 0;         ← 這字母已經使用，設定 flag

         if (depth == length) myoutput(buf);  ← 如果已經夠長就輸出
         else perm(depth + 1);  ← recursive call，深度加一

         unused[i] = 1;         ← 這字母在這位置的所有解已經列出，
                                   要換成別的字母，reset flag
      }
}
```

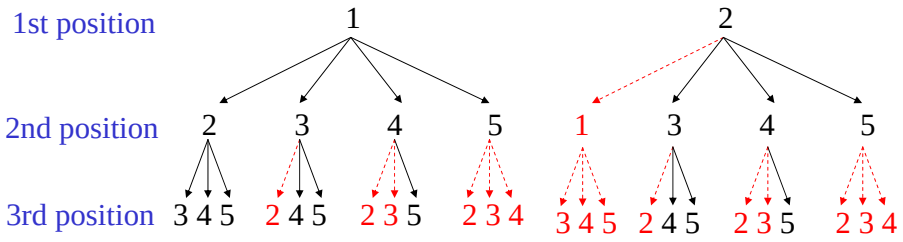

應用二: A.441 Loto

- 給 n 個整數，請列出由這 n 個數字中挑 m 個數字的所有組合方式
- 以 lexicographical 順序輸出
- $n \le 12$
  - C(n, m) 組解
- Solution: brute-force (recursive DFS)
- 困難: 怎樣避免重複的組合?

Example: {1, 2, 3, 4, 5} 取 3 個

- Three positions
  - select a number for each position

如何避免重複: 數字越選越大



Pseudo code for small m

Assume that inputs are sorted in num[n].

- **m = 2**

```
int i, j;
for (i = 0; i < n; i++)
   for (j = i + 1; j < n; j++)
      printf("%d %d\n", num[i], num[j]);
```

- **m = 3**

```
int i, j, k;
for (i = 0; i < n; i++)
   for (j = i + 1; j < n; j++)
      for (k = j + 1; k < n; j++)
         printf("%d %d %d\n", num[i], num[j], num[k]);
```

Pseudo code for all m

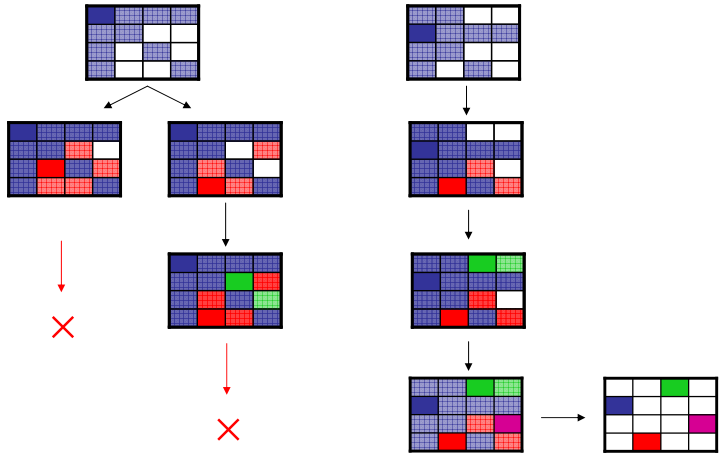
```
void comb(int base, int depth)   ← base 是要從哪一個 number 往後選
{                                  depth 是現在選到第幾個 position
   int i;
   for (i = base; i < n; i++)        ← 從 base 開始往後選
   {
     buf[depth - 1] = num[i];        ← 把目前 number 加到解中

     if (depth  == m ) myoutput(buf);  ← 如果已經夠長就輸出
     else if ((n - i) < (m – depth + 1)) break;  ← 如果剩下的 numbers
     else comb(i+1, depth+1);          ← recursive   不夠填滿剩下的解
   }                                                 ，就不繼續進行
}
```


應用三: A.167 The Sultan's Successors

- 在一個 N × N 的棋盤上，要放置 N 個 queens
- 每一格有一個 number ，表示分數
- 請找出使這 N 個 queens 吃不到彼此 (每一行、每一列及每一對角線上都最多只有一個 queen) 而且得分最高的擺法
- N = 8
- 每個 column 只能放一個，只有 8! ≈ 40000 個擺法要檢查
- Solution: brute-force search (recursive DFS)

Example: N = 4



應用四: H.91.6 專題選課

- 有 n 位老師與 m 位同學 (m 是 n 的整數倍)，每位同學需要選專題老師，每位老師收一樣多的學生
- 現在每個同學以選填志願的方式將老師排序，請找出所有排法中平均志願最佳 (志願總和最小) 的組合
- $n \le 6$, $m \le 12$

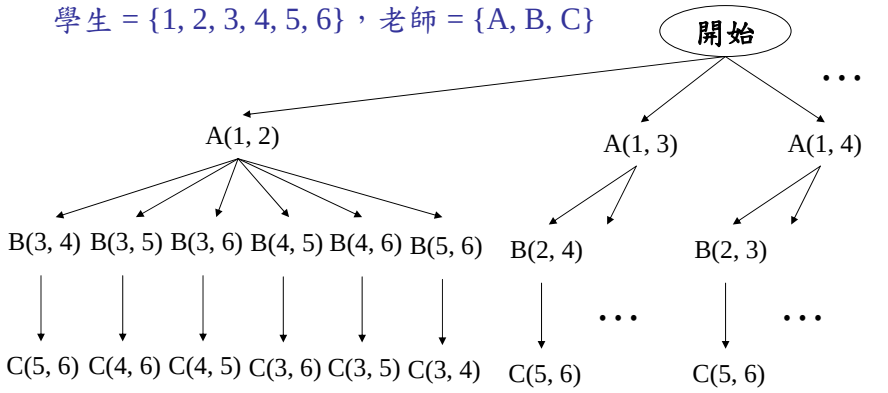
Solutions

- 若 n = 6, m = 12，則每位老師收兩個學生
  - 可看成有 12 個不同的球，要丟入 6 個不同的箱子中，每個箱子要丟兩個球
  - 有 $12! / (2!)^6 < 10^7$ 種可能
- Brute-force search is enough
- B&B
  - b: current best
  - C_e: ???

註: (1) use DP: $O(m*2^m)$ time
(2) use min-cost max-flow: $O(m^4)$-time

Example

學生 = {1, 2, 3, 4, 5, 6}，老師 = {A, B, C}

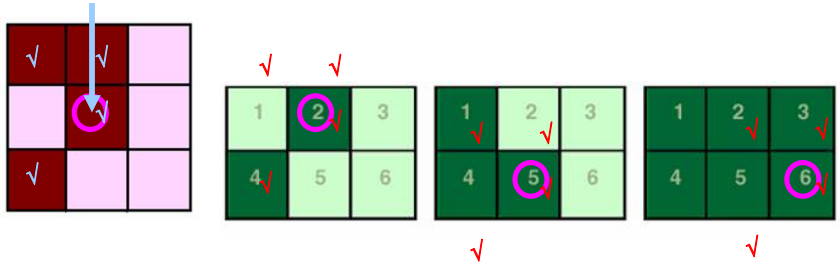


應用五: A. 10318 Security panel

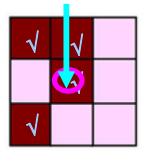
- 給一個 r × c (r, c ≤ 5) 的矩陣面板，以及按下某格後周圍格子的變化，判斷是否存在能把所有格子都啟動的按法，若有，則輸出最少次的按法 (一開始所有格子都是 off)

按下後的變化 (switch)

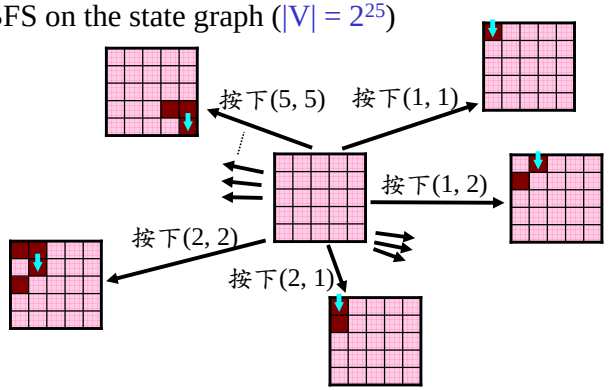
2 × 3 的面板，按下 2, 5, 6



A BFS solution



- BFS on the state graph ($|V| = 2^{25}$)



- $O(E) = O(25 \times 2^{25}) \approx 8 \times 10^8$
- not good enough

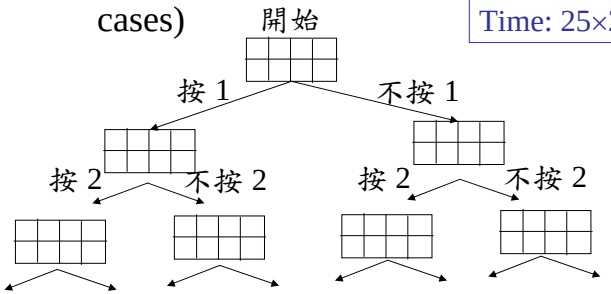
Observation

- 最多按 r × c 次
  - 每個格子若按偶數次則等於沒按，奇數次則等於按一次
  - 每個格子按或不按，與順序無關

A brute-force solution

- Brute-force:
  - $O(2^{25}) \approx 3 \times 10^7$ combinations
  - not good enough (can not process more than 10 cases)

Time: 25×2^{25} or 2^{25} ???

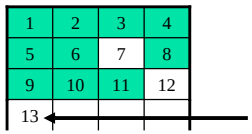


A B&B solution

- Bound: 目前已知最少按法 b (current best)
- C_e: ???

其它 C_e: ???

  - 以 1, 2, 3, ... 順序
  - 當處理格子 k 時，若離 k 上方超過兩行以上有任何格子未啟動，則此按法必定不是正確按法 ($C_e = \infty$)
  - 因為每個格子的影響範圍只有上下一行



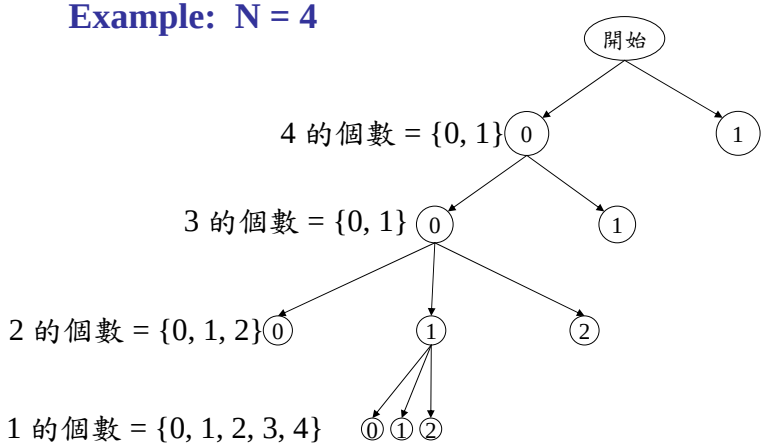
接下來沒有格子可以啟動格子 7

應用六: 整數的分割方式

- 給一正整數 N，把所有將 N 分解成若干 (1 ~ N) 個正整數的方法條列出來 (數字由大到小輸出)
- 例: N = 4
  = 3 + 1 = 2 + 2
  = 2 + 1 + 1
  = 1 + 1 + 1 + 1
- 類題:
  - 給 k 種不同的整數 $\{a_1, a_2, ..., a_k\}$，再給定一個目標數 N，請列出所有能湊出 N 的方法

A brute-force solution

Example: N = 4

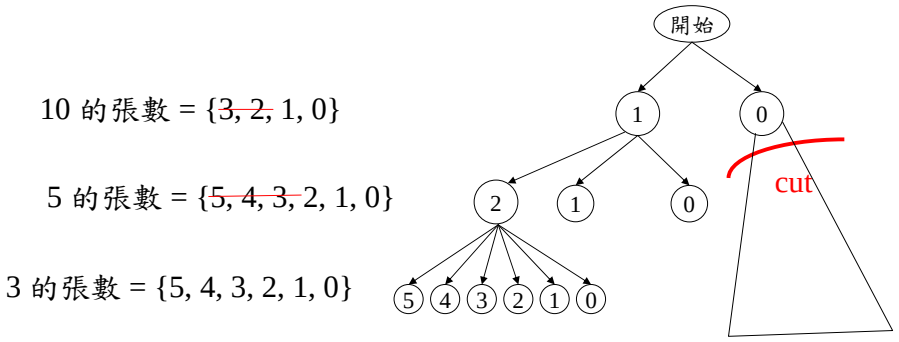


應用七: A.574 Sum it Up

- 給 k 種鈔票的面額，與每一種面額可用張數的限制，再給定一個目標數 N，請列出所有能湊出 N 的方法
- 與前一個主題類似，只是能用的整數與每個整數的個數有限制
- A brute-force solution: recursive DFS (should be good enough)
- A B&B solution
  - C_e = 剩餘所有鈔票的總金額
  - backtrack if $C_a + C_e < N$

N = 37

- 面額 = {10, 5, 3, 1} (sorted)，張數 = {1, 2, 5, 3}



作業與自我挑戰

作業

- 練習題
  - A. 10318 Security Panel
    http://uva.onlinejudge.org/external/103/10318.html
- 挑戰題
  - A. 818 Cutting Chains
    http://uva.onlinejudge.org/external/8/818.html

自我挑戰

- A. 10492 Optimal Mastermind Strategy

其它有趣題目

- A. 10344 23 out of 5
  http://uva.onlinejudge.org/external/103/10344.html
- A. 291 The House Of Santa Claus
- A. 840 Deadlock Detection
- A. 10068 The Treasure Hunt
- A. 838 Worm World (Hint: DFS in (D, R, U, L) order)