ZeroJudge q183. 3.重組問題

Turnpike Reconstruction problem

https://zerojudge.tw/ShowProblem?problemid=q183

APCS 114/01

Pei-yih Ting

題目說明

題目說明

- 定義：一個非負整數數列 $a_1, a_2, \ldots, a_n$，其中 $a_1=0$，$a_i \neq a_j$，
  任意兩個元素差的絕對值集合 $\Delta=\{d_1, d_2, \ldots, d_k\}$, $k=n(n-1)/2$

題目說明

- 定義：一個非負整數數列 $a_1, a_2, \ldots, a_n$，其中 $a_1=0$，$a_i \neq a_j$，
  任意兩個元素差的絕對值集合 $\Delta=\{d_1, d_2, \ldots, d_k\}$, $k=n(n-1)/2$
  例如: 數列 0, 3, 7，兩兩之間差的絕對值集合 $\Delta=\{3,4,7\}$

題目說明

- 定義：一個非負整數數列 $a_1, a_2, \ldots, a_n$，其中 $a_1=0$，$a_i \neq a_j$，
  任意兩個元素差的絕對值集合 $\Delta=\{d_1, d_2, \ldots, d_k\}$, $k=n(n-1)/2$
  例如: 數列 0, 3, 7，兩兩之間差的絕對值集合 $\Delta=\{3,4,7\}$

- 問題：指定一個元素差的絕對值集合 Δ，希望找出
  字典序最小以及字典序最大的兩個非負整數數列

題目說明

- 定義：一個非負整數數列 $a_1, a_2, \ldots, a_n$，其中 $a_1=0$，$a_i \neq a_j$，
  任意兩個元素差的絕對值集合 $\Delta=\{d_1, d_2, \ldots, d_k\}$, $k=n(n-1)/2$
  例如: 數列 0, 3, 7，兩兩之間差的絕對值集合 $\Delta=\{3,4,7\}$
- 問題：指定一個元素差的絕對值集合 Δ，希望找出
  字典序最小以及字典序最大的兩個非負整數數列
- 輸入有兩列：第一列是一個正整數 n, $(1 \leq n \leq 25)$，第二列有 $n(n-1)/2$ 個正整數是數列元素差的絕對值集合 Δ， $1 \leq d_i \leq 100$

題目說明

- 定義：一個非負整數數列 $a_1, a_2, \ldots, a_n$，其中 $a_1=0$，$a_i \neq a_j$，
  任意兩個元素差的絕對值集合 $\Delta=\{d_1, d_2, \ldots, d_k\}$, $k=n(n-1)/2$
  例如: 數列 0, 3, 7，兩兩之間差的絕對值集合 $\Delta=\{3,4,7\}$
- 問題：指定一個元素差的絕對值集合 Δ，希望找出
  字典序最小以及字典序最大的兩個非負整數數列
- 輸入有兩列：第一列是一個正整數 n, $(1 \leq n \leq 25)$，第二列有 $n(n-1)/2$ 個正整數是數列元素差的絕對值集合 Δ， $1 \leq d_i \leq 100$
- 輸出也是兩列：分別是字典序最小以及字典序最大的數列

題目說明

- 定義：一個非負整數數列 $a_1, a_2, \ldots, a_n$，其中 $a_1=0$，$a_i \neq a_j$，
  任意兩個元素差的絕對值集合 $\Delta=\{d_1, d_2, \ldots, d_k\}$, $k=n(n-1)/2$
  例如: 數列 0, 3, 7，兩兩之間差的絕對值集合 $\Delta=\{3,4,7\}$
- 問題：指定一個元素差的絕對值集合 Δ，希望找出
  字典序最小以及字典序最大的兩個非負整數數列
- 輸入有兩列：第一列是一個正整數 n, $(1 \leq n \leq 25)$，第二列有 $n(n-1)/2$ 個正整數是數列元素差的絕對值集合 Δ， $1 \leq d_i \leq 100$
- 輸出也是兩列：分別是字典序最小以及字典序最大的數列

輸入測資

```
3
3 4 7
```

題目說明

- 定義：一個非負整數數列 $a_1, a_2, \ldots, a_n$，其中 $a_1=0$，$a_i \neq a_j$，
  任意兩個元素差的絕對值集合 $\Delta=\{d_1, d_2, \ldots, d_k\}$, $k=n(n-1)/2$
  例如: 數列 0, 3, 7，兩兩之間差的絕對值集合 $\Delta=\{3,4,7\}$
- 問題：指定一個元素差的絕對值集合 Δ，希望找出
  字典序最小以及字典序最大的兩個非負整數數列
- 輸入有兩列：第一列是一個正整數 n, $(1 \leq n \leq 25)$，第二列有 $n(n-1)/2$ 個正整數是數列元素差的絕對值集合 Δ， $1 \leq d_i \leq 100$
- 輸出也是兩列：分別是字典序最小以及字典序最大的數列

輸入測資	輸出測資
3	0 3 7
3 4 7	0 4 7

題目說明

- 定義：一個非負整數數列 $a_1, a_2, \ldots, a_n$，其中 $a_1=0$，$a_i \neq a_j$，
  任意兩個元素差的絕對值集合 $\Delta=\{d_1, d_2, \ldots, d_k\}$, $k=n(n-1)/2$
  例如: 數列 0, 3, 7，兩兩之間差的絕對值集合 $\Delta=\{3,4,7\}$
- 問題：指定一個元素差的絕對值集合 Δ，希望找出
  字典序最小以及字典序最大的兩個非負整數數列
- 輸入有兩列：第一列是一個正整數 n, $(1 \leq n \leq 25)$，第二列有 $n(n-1)/2$ 個正整數是數列元素差的絕對值集合 Δ， $1 \leq d_i \leq 100$
- 輸出也是兩列：分別是字典序最小以及字典序最大的數列

輸入測資	輸出測資	輸入測資
3	0 3 7	5
3 4 7	0 4 7	1 2 3 3 5 5 6 8 10 11

題目說明

- 定義：一個非負整數數列 $a_1, a_2, \ldots, a_n$，其中 $a_1=0$，$a_i \neq a_j$，
  任意兩個元素差的絕對值集合 $\Delta=\{d_1, d_2, \ldots, d_k\}$, $k=n(n-1)/2$
  例如: 數列 0, 3, 7，兩兩之間差的絕對值集合 $\Delta=\{3,4,7\}$
- 問題：指定一個元素差的絕對值集合 Δ，希望找出
  字典序最小以及字典序最大的兩個非負整數數列
- 輸入有兩列：第一列是一個正整數 n, $(1 \leq n \leq 25)$，第二列有 $n(n-1)/2$ 個正整數是數列元素差的絕對值集合 Δ， $1 \leq d_i \leq 100$
- 輸出也是兩列：分別是字典序最小以及字典序最大的數列

輸入測資	輸出測資
3 3 4 7	0 3 7 0 4 7

輸入測資	輸出測資
5 1 2 3 3 5 5 6 8 10 11	0 1 3 6 11 0 5 8 10 11

問題分析

- 字典序：由英文字典中的單字的排序方法延伸而來

問題分析

- 字典序：由英文字典中的單字的排序方法延伸而來

  例如：比較兩個整數數列 $A = a_1\ a_2\ a_3\ a_4$ 與 $B = b_1\ b_2\ b_3$

問題分析

- 字典序：由英文字典中的單字的排序方法延伸而來

  例如：比較兩個整數數列 $A = a_1\ a_2\ a_3\ a_4$ 與 $B = b_1\ b_2\ b_3$

  由左至右依序比較 a_i 與 b_i

問題分析

- 字典序：由英文字典中的單字的排序方法延伸而來

例如：比較兩個整數數列 $A = a_1\ a_2\ a_3\ a_4$ 與 $B = b_1\ b_2\ b_3$

由左至右依序比較 a_i 與 b_i

$a_i == b_i$ 則繼續比較 a_{i+1} 與 b_{i+1}

$a_i > b_i$ 則 $A > B$

$a_i < b_i$ 則 $A < B$

問題分析

- 字典序：由英文字典中的單字的排序方法延伸而來

例如：比較兩個整數數列 $A = a_1\ a_2\ a_3\ a_4$ 與 $B = b_1\ b_2\ b_3$

由左至右依序比較 a_i 與 b_i

$a_i == b_i$ 則繼續比較 a_{i+1} 與 b_{i+1}

$a_i > b_i$ 則 $A > B$

$a_i < b_i$ 則 $A < B$

如果一直都相等，但是其中一字串比較短，則較長的字串較大，上例中如 $a_1=b_1$, $a_2=b_2$,$a_3=b_3$ 則 $A>B$

問題分析

- 字典序：由英文字典中的單字的排序方法延伸而來

  例如：比較兩個整數數列 $A = a_1\ a_2\ a_3\ a_4$ 與 $B = b_1\ b_2\ b_3$

  由左至右依序比較 a_i 與 b_i

  $a_i == b_i$ 則繼續比較 a_{i+1} 與 b_{i+1}

  $a_i > b_i$ 則 $A > B$

  $a_i < b_i$ 則 $A < B$

  如果一直都相等，但是其中一字串比較短，則較長的字串較大，上例中如 $a_1=b_1$, $a_2=b_2$,$a_3=b_3$ 則 $A>B$

- 撰寫程式前的第一步就是先手動分析範例，沒有人能夠在不太曉得怎麼手動計算出來之前就寫出程式的

問題分析

問題: n, k=n(n-1)/2, $0 < d_1 \le d_2 \le \ldots < d_k$

求 $a_1 < a_2 < a_3 < \ldots < a_n$, 其中 $a_1=0$

問題分析

- 基本性質

問題: n, k=n(n-1)/2, $0 < d_1 \le d_2 \le \ldots < d_k$

求 $a_1 < a_2 < a_3 < \ldots < a_n$, 其中 $a_1=0$

問題分析

- 基本性質

問題: n, k=n(n-1)/2, $0 < d_1 \le d_2 \le \ldots < d_k$

求 $a_1 < a_2 < a_3 < \ldots < a_n$, 其中 $a_1=0$

1. 任何 a_i 與其它數字的距離中以 a_1 或 a_{n-1} 距離為最遠

問題分析

- 基本性質

問題: n, k=n(n-1)/2, $0 < d_1 \le d_2 \le \ldots < d_k$

求 $a_1 < a_2 < a_3 < \ldots < a_n$, 其中 $a_1=0$

1. 任何 a_i 與其它數字的距離中以 a_1 或 a_{n-1} 距離為最遠
2. d_k 必須大於 d_{k-1} 且 $a_n = d_k$ (由最大的 d_k 可以推出最大的 a_n)

問題分析

- 基本性質

問題: n, k=n(n-1)/2, $0 < d_1 \le d_2 \le \ldots < d_k$

求 $a_1 < a_2 < a_3 < \ldots < a_n$, 其中 $a_1=0$

1. 任何 a_i 與其它數字的距離中以 a_1 或 a_{n-1} 距離為最遠
2. d_k 必須大於 d_{k-1} 且 $a_n = d_k$ (由最大的 d_k 可以推出最大的 a_n)
3. 新增第三個數，要得到第二大的 d_{k-1}，則此數必為 $a_2=a_n-d_{k-1}$
   或是 $a_{n-1}=a_1+d_{k-1}$

問題分析

- 基本性質

問題: n, k=n(n-1)/2, $0 < d_1 \le d_2 \le \ldots < d_k$

求 $a_1 < a_2 < a_3 < \ldots < a_n$, 其中 $a_1=0$

1. 任何 a_i 與其它數字的距離中以 a_1 或 a_{n-1} 距離為最遠
2. d_k 必須大於 d_{k-1} 且 $a_n = d_k$ (由最大的 d_k 可以推出最大的 a_n)
3. 新增第三個數，要得到第二大的 d_{k-1}，則此數必為 $a_2=a_n-d_{k-1}$
   或是 $a_{n-1}=a_1+d_{k-1}$，因為 $\forall\, i>2, a_n-a_2 > a_n-a_i$ 且 $\forall j<n-1, a_{n-1}-a_1 > a_j-a_1$

問題分析

- 基本性質

問題: n, k=n(n-1)/2, $0 < d_1 \le d_2 \le \ldots < d_k$

求 $a_1 < a_2 < a_3 < \ldots < a_n$, 其中 $a_1=0$

1. 任何 a_i 與其它數字的距離中以 a_1 或 a_{n-1} 距離為最遠
2. d_k 必須大於 d_{k-1} 且 $a_n = d_k$ (由最大的 d_k 可以推出最大的 a_n)
3. 新增第三個數，要得到第二大的 d_{k-1}，則此數必為 $a_2=a_n-d_{k-1}$
   或是 $a_{n-1}=a_1+d_{k-1}$，因為 $\forall\, i>2, a_n-a_2 > a_n-a_i$ 且 $\forall j<n-1, a_{n-1}-a_1 > a_j-a_1$

也就是說由第二大的 d_{k-1} 可以推出 a_2 或是 a_{n-1} 二者之一

問題分析

- 基本性質

問題: n, k=n(n-1)/2, $0 < d_1 \le d_2 \le \ldots < d_k$

求 $a_1 < a_2 < a_3 < \ldots < a_n$, 其中 $a_1=0$

1. 任何 a_i 與其它數字的距離中以 a_1 或 a_{n-1} 距離為最遠
2. d_k 必須大於 d_{k-1} 且 $a_n = d_k$ (由最大的 d_k 可以推出最大的 a_n)
3. 新增第三個數，要得到第二大的 d_{k-1}，則此數必為 $a_2=a_n-d_{k-1}$
   或是 $a_{n-1}=a_1+d_{k-1}$，因為 $\forall\, i>2, a_n-a_2 > a_n-a_i$ 且 $\forall j<n-1, a_{n-1}-a_1 > a_j-a_1$

也就是說由第二大的 d_{k-1} 可以推出 a_2 或是 a_{n-1} 二者之一

如何判斷是哪一個呢?

問題分析

- 基本性質

問題: n, k=n(n-1)/2, $0 < d_1 \le d_2 \le \ldots < d_k$

求 $a_1 < a_2 < a_3 < \ldots < a_n$, 其中 $a_1=0$

1. 任何 a_i 與其它數字的距離中以 a_1 或 a_{n-1} 距離為最遠
2. d_k 必須大於 d_{k-1} 且 $a_n = d_k$ (由最大的 d_k 可以推出最大的 a_n)
3. 新增第三個數，要得到第二大的 d_{k-1}，則此數必為 $a_2=a_n-d_{k-1}$
   或是 $a_{n-1}=a_1+d_{k-1}$，因為 $\forall\, i>2, a_n-a_2 > a_n-a_i$ 且 $\forall j<n-1, a_{n-1}-a_1 > a_j-a_1$

也就是說由第二大的 d_{k-1} 可以推出 a_2 或是 a_{n-1} 二者之一

如何判斷是哪一個呢?

例如 $a_2 = a_n - d_{k-1}$ 則 $\{a_2-a_1, a_i-a_2, i=3,\ldots,n-1\} \subset \Delta\backslash\{d_{k-1},d_k\}$

但是 $a_i, i=3,\ldots,n-1$ 都是未知，所以至少需要滿足 $a_2-a_1 \in \Delta\backslash\{d_{k-1},d_k\}$，究竟 a_2 是不是 $a_n - d_{k-1}$ 需要等 $a_i, i=3,\ldots,n-1$ 都設定之後才能確定

問題分析

- 基本性質

問題: n, k=n(n-1)/2, $0 < d_1 \le d_2 \le \ldots < d_k$

求 $a_1 < a_2 < a_3 < \ldots < a_n$, 其中 $a_1=0$

1. 任何 a_i 與其它數字的距離中以 a_1 或 a_{n-1} 距離為最遠
2. d_k 必須大於 d_{k-1} 且 $a_n = d_k$ (由最大的 d_k 可以推出最大的 a_n)
3. 新增第三個數，要得到第二大的 d_{k-1}，則此數必為 $a_2=a_n-d_{k-1}$
   或是 $a_{n-1}=a_1+d_{k-1}$，因為 $\forall\, i>2,\ a_n-a_2 > a_n-a_i$ 且 $\forall j<n-1,\ a_{n-1}-a_1 > a_j-a_1$

也就是說由第二大的 d_{k-1} 可以推出 a_2 或是 a_{n-1} 二者之一

如何判斷是哪一個呢?

例如 $a_2 = a_n - d_{k-1}$ 則 $\{a_2-a_1, a_i-a_2, i=3,\ldots,n-1\} \subset \Delta\backslash\{d_{k-1},d_k\}$

但是 $a_i, i=3,\ldots,n-1$ 都是未知，所以至少需要滿足 $a_2-a_1 \in \Delta\backslash\{d_{k-1},d_k\}$，究竟 a_2 是不是 $a_n - d_{k-1}$ 需要等 $a_i, i=3,\ldots,n-1$ 都設定之後才能確定

和老鼠走迷宮或是速讀的問題很像

範例解析

範例解析

- 解題過程中會有一個一個的決策點

範例解析

- 解題過程中會有一個一個的決策點 ，有些決策不符合限制可以立刻排除

範例解析

- 解題過程中會有一個一個的決策點 ，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹

範例解析

- 解題過程中會有一個一個的決策點 ，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支

範例解析

- 解題過程中會有一個一個的決策點 ，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支

- 範例一：n=3, Δ={3,4,7}

範例解析

- 解題過程中會有一個一個的決策點 ，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支
- 範例一：n=3, Δ={3,4,7}

  令 d_1=3, d_2=4, d_3=7

範例解析

- 解題過程中會有一個一個的決策點 ，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支
- 範例一：n=3, Δ={3,4,7}

  令 $d_1=3$, $d_2=4$, $d_3=7$

  - $a_1=0$

範例解析

- 解題過程中會有一個一個的決策點 ，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支
- 範例一：n=3, Δ={3,4,7}

令 $d_1=3$, $d_2=4$, $d_3=7$

- $a_1=0$

$d_3=7$

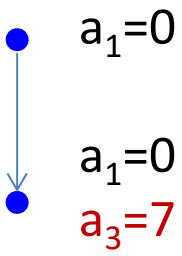
範例解析

- 解題過程中會有一個一個的決策點 ，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支

- 範例一：n=3, Δ={3,4,7}

令 $d_1=3$, $d_2=4$, $d_3=7$

$d_3=7$

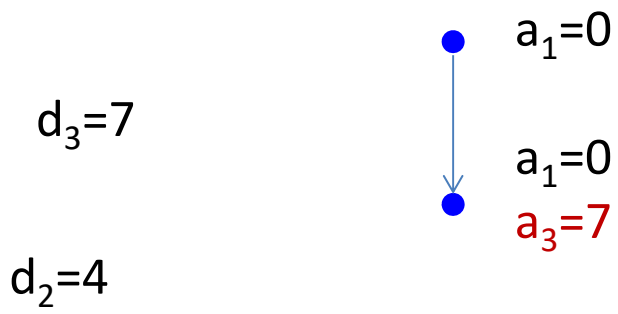


範例解析

- 解題過程中會有一個一個的決策點 ，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支

- 範例一：n=3, Δ={3,4,7}

令 $d_1=3$, $d_2=4$, $d_3=7$

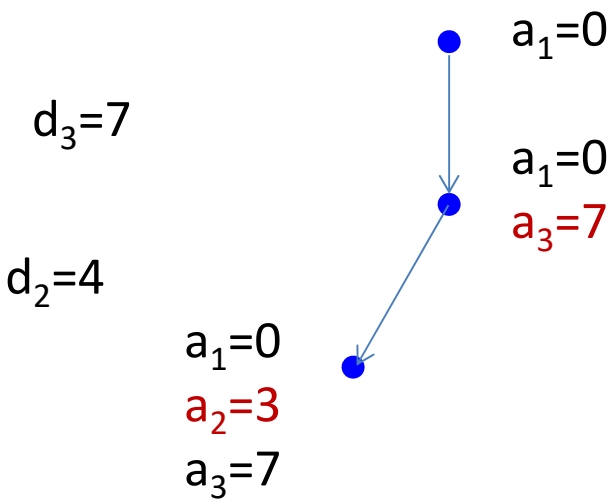


範例解析

- 解題過程中會有一個一個的決策點，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支

- 範例一：n=3, Δ={3,4,7}

令 $d_1=3$, $d_2=4$, $d_3=7$

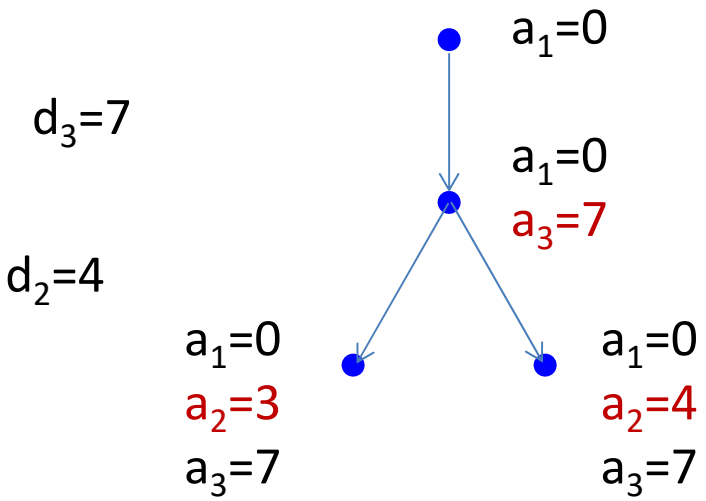


範例解析

- 解題過程中會有一個一個的決策點，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支

- 範例一：n=3, Δ={3,4,7}

令 $d_1=3$, $d_2=4$, $d_3=7$

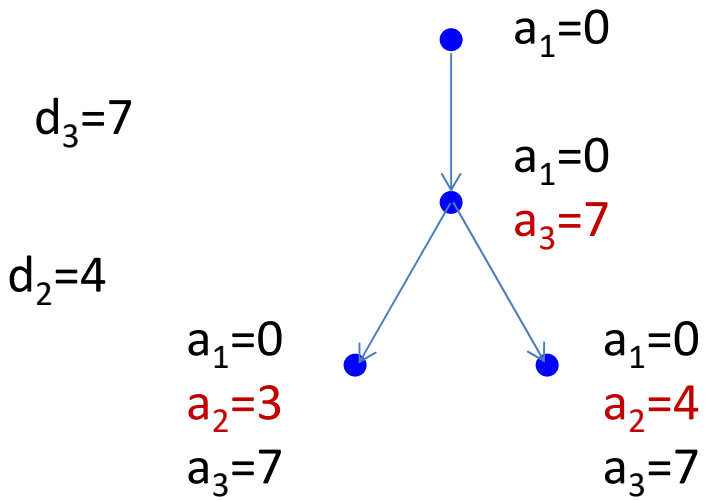


範例解析

- 解題過程中會有一個一個的決策點 ，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支

- 範例一：n=3, Δ={3,4,7}

令 d_1=3, d_2=4, d_3=7



輸出測資

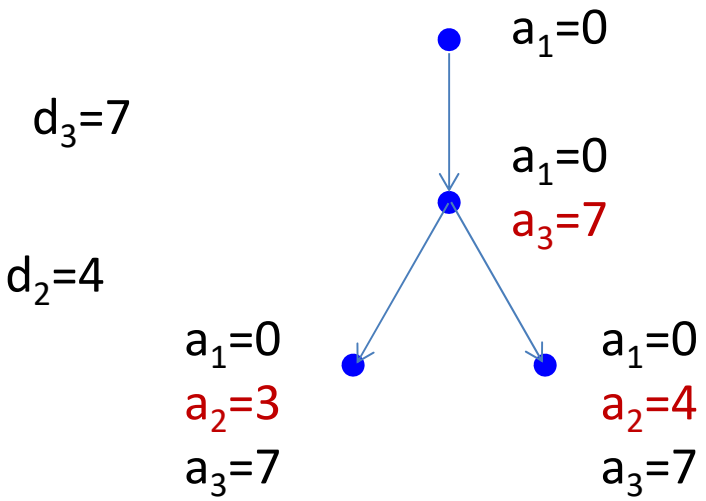
0 3 7
0 4 7

範例解析

- 解題過程中會有一個一個的決策點，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支

- 範例一：n=3, Δ={3,4,7}

令 d_1=3, d_2=4, d_3=7



輸出測資

0 3 7
0 4 7

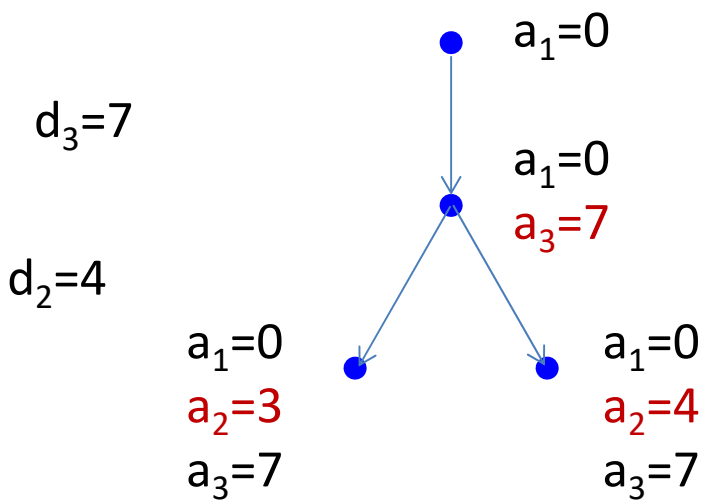
- 如果每次決策都是由 {a_n-d, d} 中比較小的數值 a_n-d 開始測試，那麼第一個找到的答案就是字典序最小的答案

範例解析

- 解題過程中會有一個一個的決策點，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支

- 範例一：n=3, Δ={3,4,7}

令 d_1=3, d_2=4, d_3=7



輸出測資

0 3 7
0 4 7

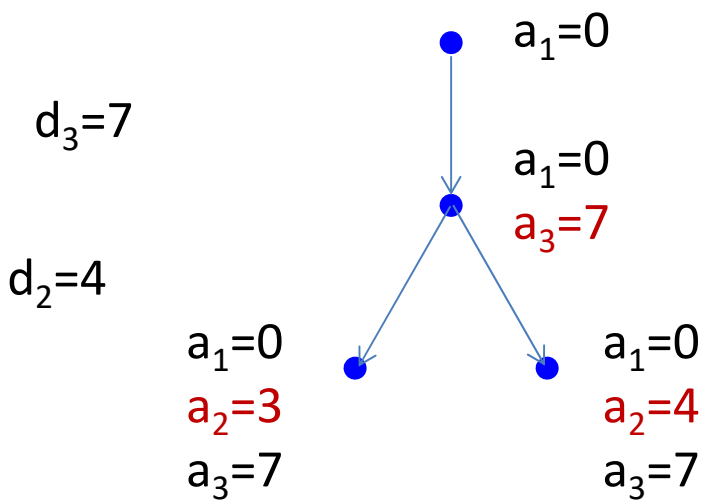
- 如果每次決策都是由 {a_n-d, d} 中比較小的數值 a_n-d 開始測試，那麼第一個找到的答案就是字典序最小的答案
- 雖然在這例子裡第二個答案就是字典序最大的答案，但是這個二元樹隨著 n 變大會有 2^n 個分支，不可能找完所有答案

範例解析

- 解題過程中會有一個一個的決策點，有些決策不符合限制可以立刻排除，但是有些決策需要等到後續一系列決策之後才能夠判斷是否可行 – 決策樹 – **DFS** 基本上是一種嘗試錯誤的搜索方法在決策樹中找尋滿足限制的決策分支

- 範例一：n=3, Δ={3,4,7}

令 d_1=3, d_2=4, d_3=7



輸出測資

0 3 7
0 4 7

- 如果每次決策都是由 {a_n-d, d} 中比較小的數值 a_n-d 開始測試，那麼第一個找到的答案就是字典序最小的答案
- 雖然在這例子裡第二個答案就是字典序最大的答案，但是這個二元樹隨著 n 變大會有 2^n 個分支，不可能找完所有答案
- 而且有些問題不只兩個答案

範例解析

範例解析

- 很容易可以證明這個問題答案的數列間距滿足一種對稱性

範例解析

- 很容易可以證明這個問題答案的數列間距滿足一種對稱性

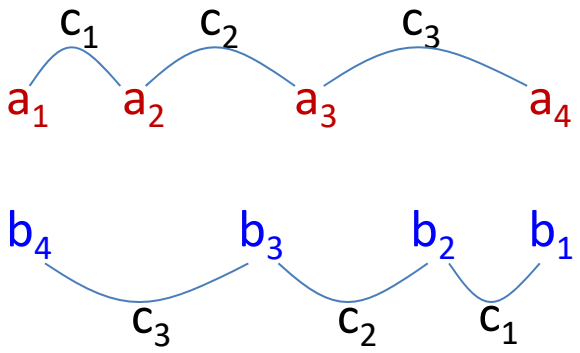
  如果右圖中 a_1,a_2,a_3,a_4 是一組答案

a_1 a_2 a_3 a_4

範例解析

- 很容易可以證明這個問題答案的數列間距滿足一種對稱性

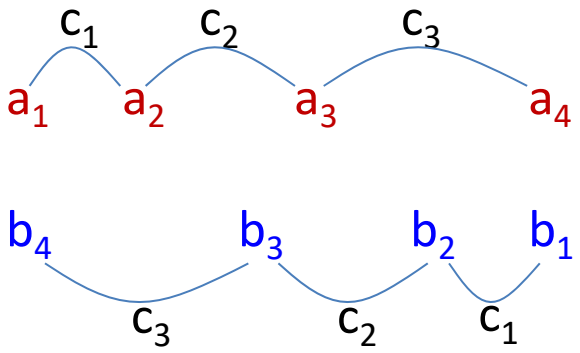
如果右圖中 a_1,a_2,a_3,a_4 是一組答案，
則滿足 $b_4=a_1=0$, $b_1=a_4$ 且 $c_i=a_{i+1}-a_i=b_i-b_{i+1}$,
$i=1,2,3$ 的 b_4,b_3,b_2,b_1 也會是一組答案



範例解析

- 很容易可以證明這個問題答案的數列間距滿足一種對稱性

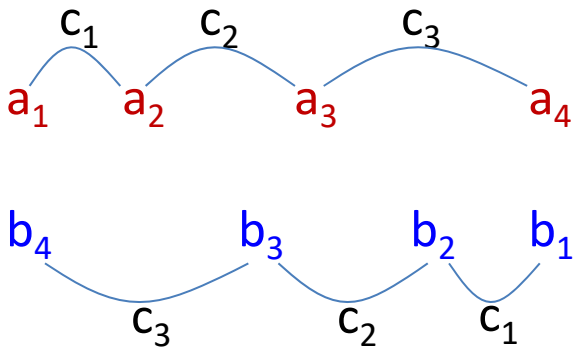
如果右圖中 a_1,a_2,a_3,a_4 是一組答案，
則滿足 $b_4=a_1=0$, $b_1=a_4$ 且 $c_i=a_{i+1}-a_i=b_i-b_{i+1}$,
$i=1,2,3$ 的 b_4,b_3,b_2,b_1 也會是一組答案，
它們差的絕對值集合都是相同的
$\Delta=\{c_1,c_2,c_3,c_1+c_2,c_2+c_3,c_1+c_2+c_3\}$



範例解析

- 很容易可以證明這個問題答案的數列間距滿足一種對稱性

  如果右圖中 a_1,a_2,a_3,a_4 是一組答案，
  則滿足 $b_4=a_1=0$, $b_1=a_4$ 且 $c_i=a_{i+1}-a_i=b_i-b_{i+1}$,
  $i=1,2,3$ 的 b_4,b_3,b_2,b_1 也會是一組答案，
  它們差的絕對值集合都是相同的
  $\Delta=\{c_1,c_2,c_3,c_1+c_2,c_2+c_3,c_1+c_2+c_3\}$

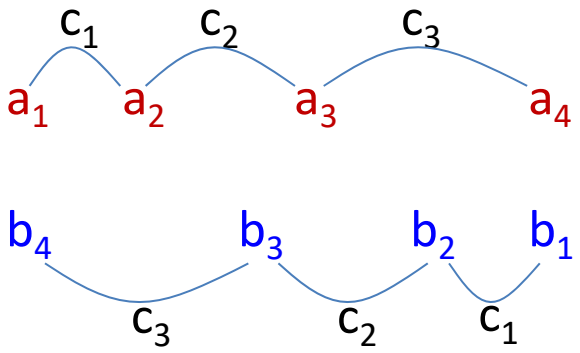


- 對稱於字典序最小的那一組答案一定是字典序最大的答案

範例解析

- 很容易可以證明這個問題答案的**數列間距**滿足一種**對稱性**

  如果右圖中 a_1,a_2,a_3,a_4 是一組答案，
  則滿足 $b_4=a_1=0$, $b_1=a_4$ 且 $c_i=a_{i+1}-a_i=b_i-b_{i+1}$,
  $i=1,2,3$ 的 b_4,b_3,b_2,b_1 也會是一組答案，
  它們差的絕對值集合都是相同的
  $\Delta=\{c_1,c_2,c_3,c_1+c_2,c_2+c_3,c_1+c_2+c_3\}$



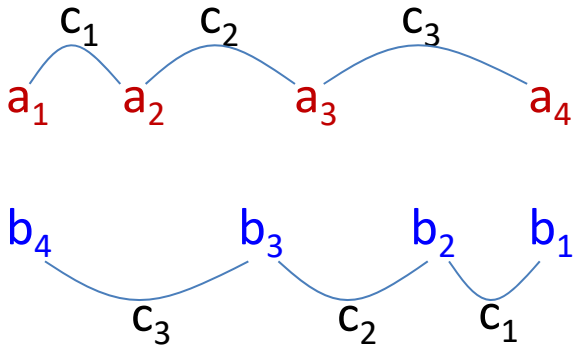
- 對稱於字典序最小的那一組答案一定是字典序最大的答案

  如果 a_1,a_2,a_3,a_4 是一組字典序最小的答案，且 b_4,b_3,b_2,b_1 滿足上面描述的對稱性，則 b_4,b_3,b_2,b_1 會是字典序最大的答案

範例解析

- 很容易可以證明這個問題答案的**數列間距**滿足一種**對稱性**

  如果右圖中 a_1,a_2,a_3,a_4 是一組答案，
  則滿足 $b_4=a_1=0$, $b_1=a_4$ 且 $c_i=a_{i+1}-a_i=b_i-b_{i+1}$,
  $i=1,2,3$ 的 b_4,b_3,b_2,b_1 也會是一組答案，
  它們差的絕對值集合都是相同的
  $\Delta=\{c_1,c_2,c_3,c_1+c_2,c_2+c_3,c_1+c_2+c_3\}$



- 對稱於字典序最小的那一組答案一定是字典序最大的答案

  如果 a_1,a_2,a_3,a_4 是一組字典序最小的答案，且 b_4,b_3,b_2,b_1 滿足上面描述的對稱性，則 b_4,b_3,b_2,b_1 會是字典序最大的答案

  pf: 如果 b_4,b_3,b_2,b_1 字典序不是最大，還有另外一個字典序更大的 $\beta_4,\beta_3,\beta_2,\beta_1$ 的答案，則與其對稱的 $\alpha_1,\alpha_2,\alpha_3,\alpha_4$ 是一個比 a_1,a_2,a_3,a_4 字典序更小的答案。

範例解析

- 範例二：n=5

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好

- a_1=0

範例解析

- $a_1=0$

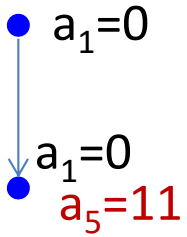
$d_{10}=11$

- 範例二：n=5

$\Delta=\{1,2,3,3,5,5,6,8,10,11\}$

將 Δ 內的數值
由小到大排好

範例解析



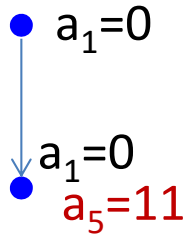
d_{10}=11

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好

範例解析



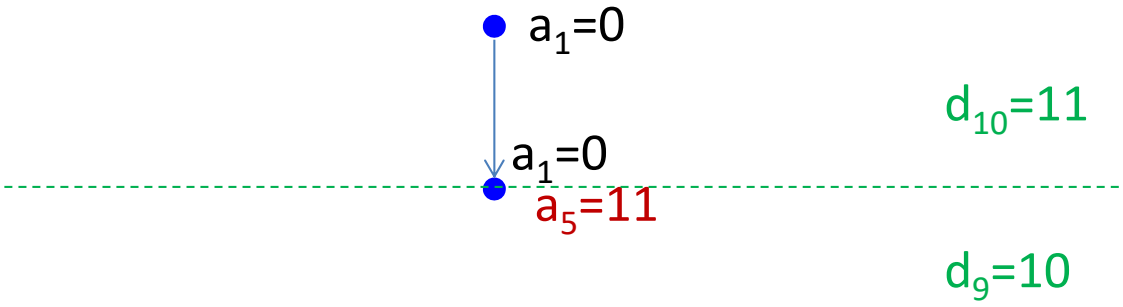
d_{10}=11

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好

範例解析



- 範例二：n=5

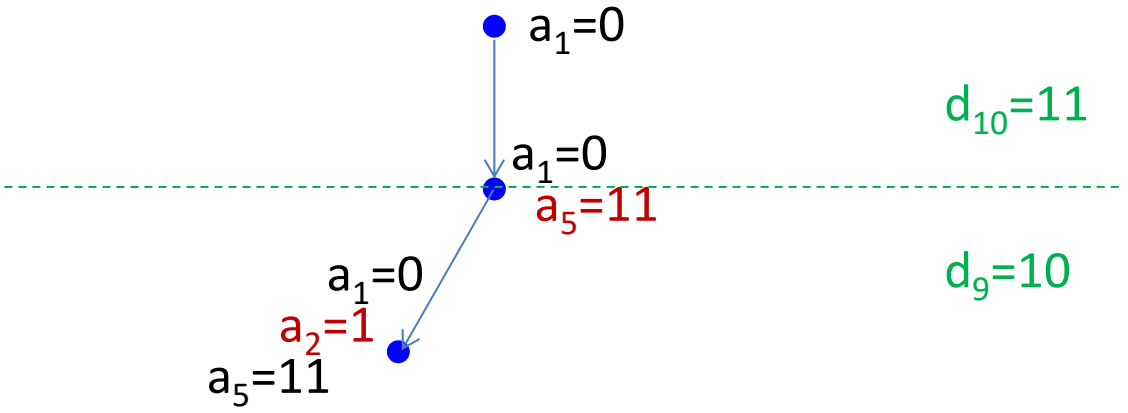
Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}



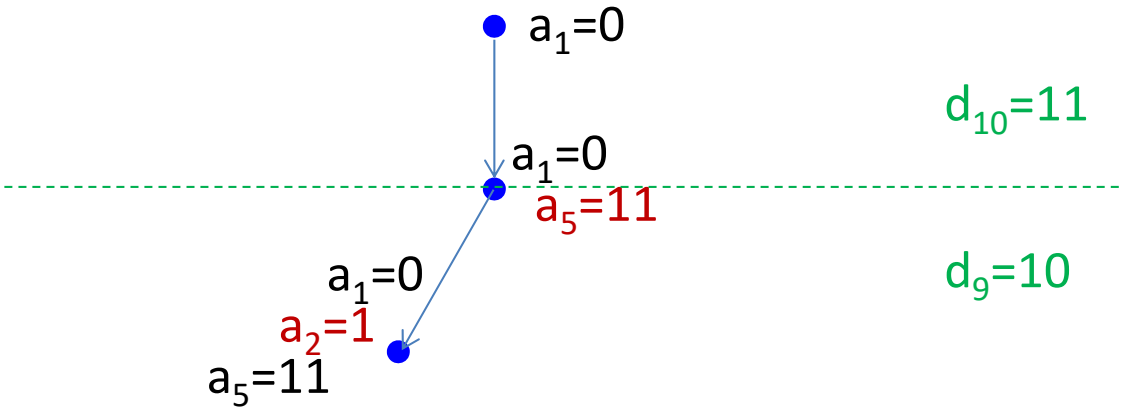
將 Δ 內的數值
由小到大排好

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好

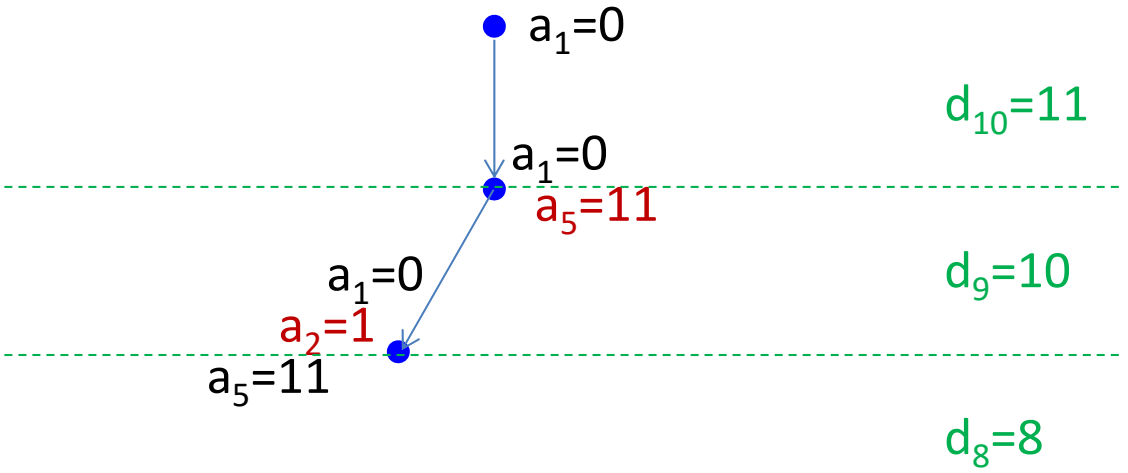


範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

× × ×



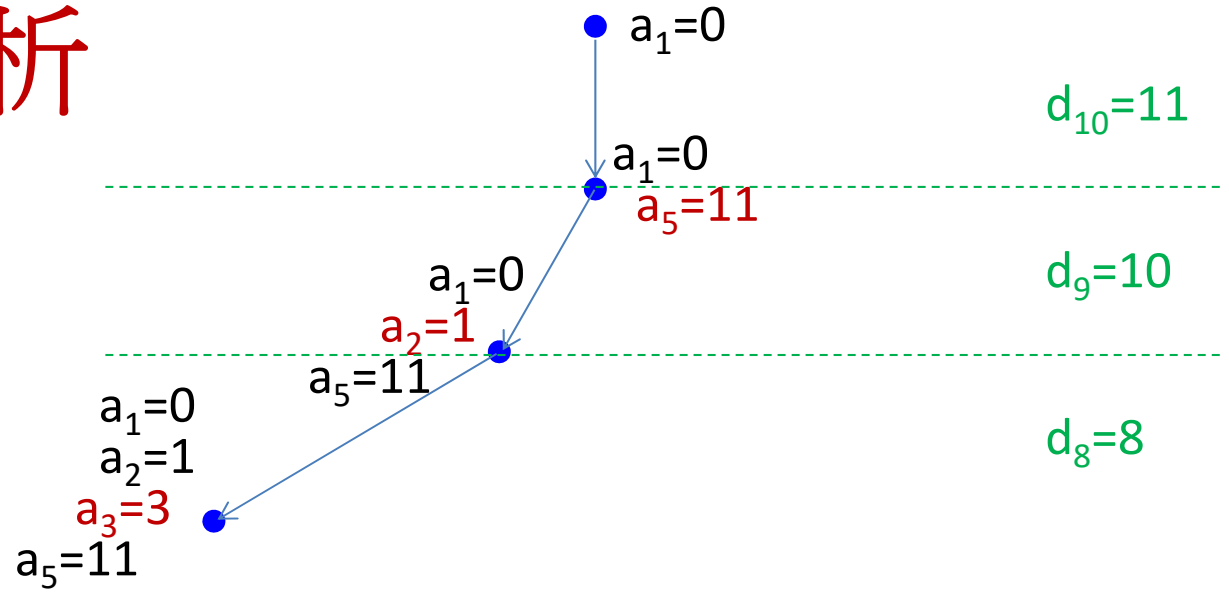
將 Δ 內的數值
由小到大排好

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

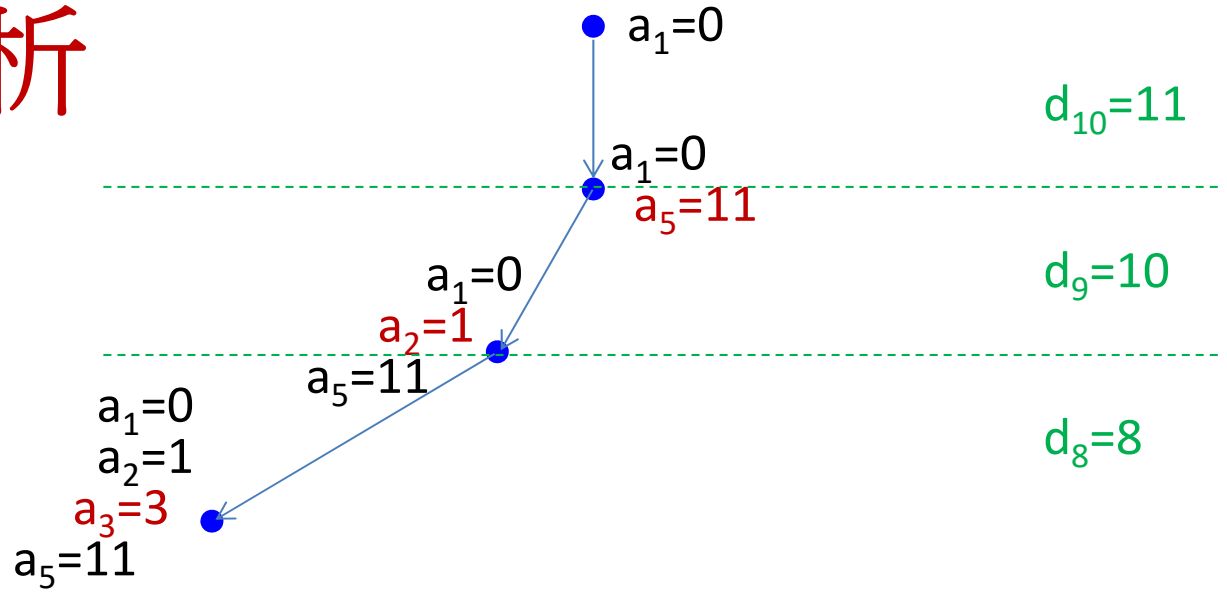
將 Δ 內的數值
由小到大排好



範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}



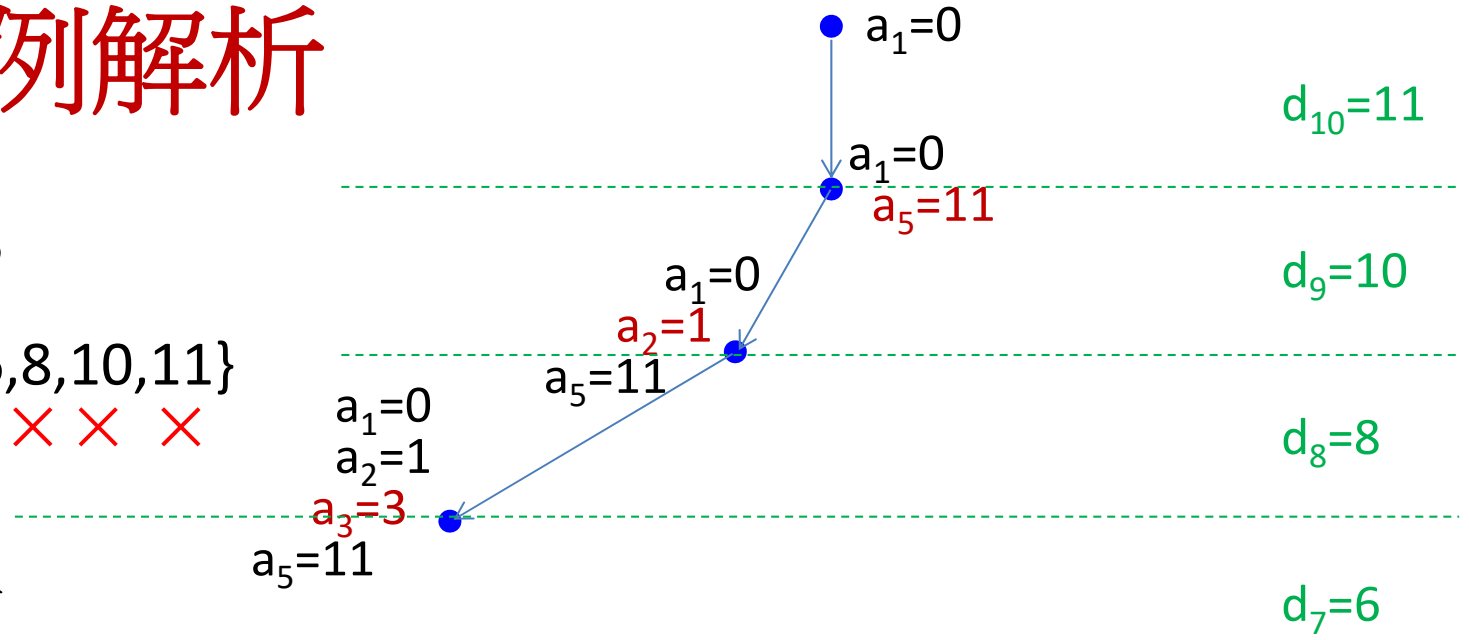
將 Δ 內的數值
由小到大排好

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好

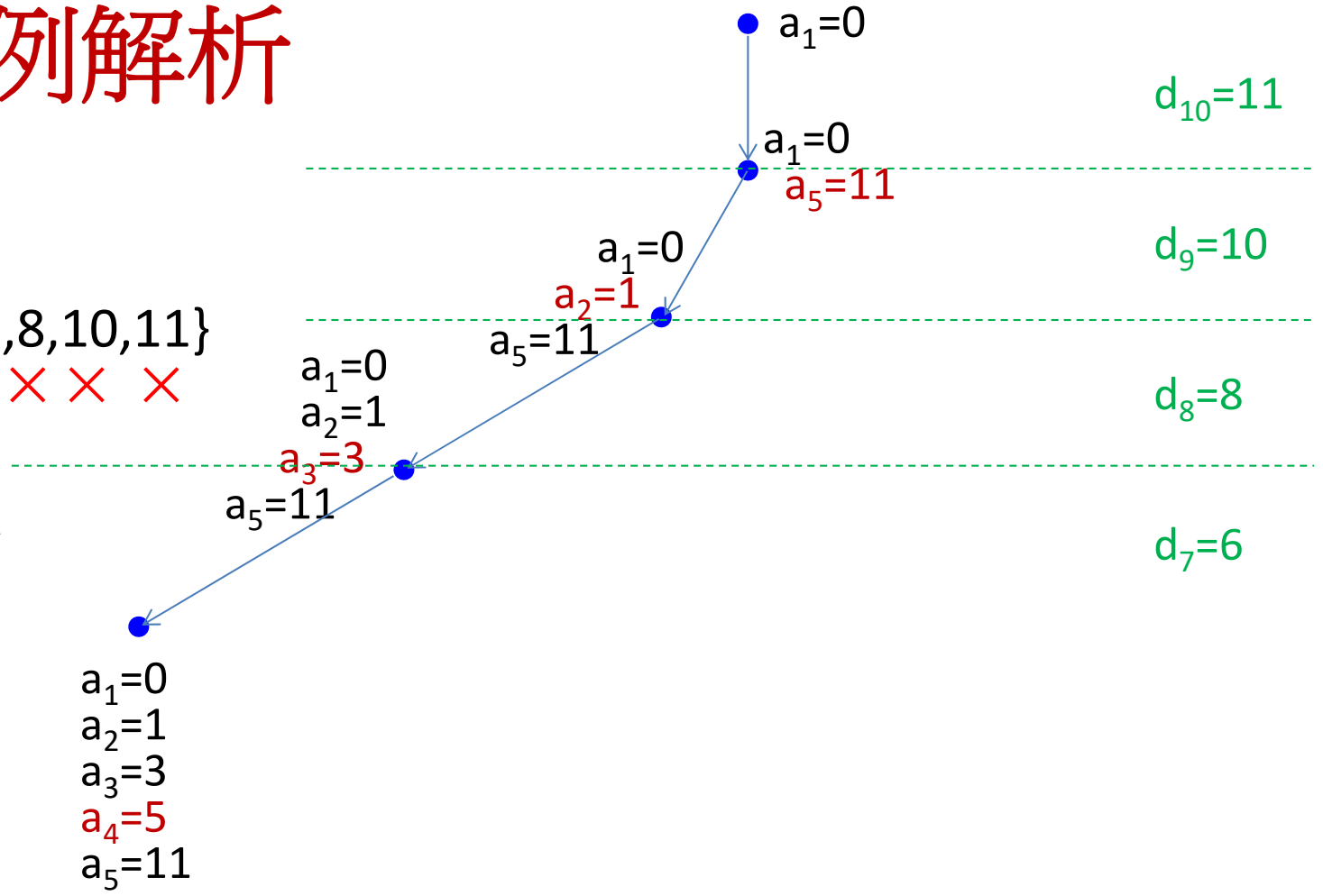


範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



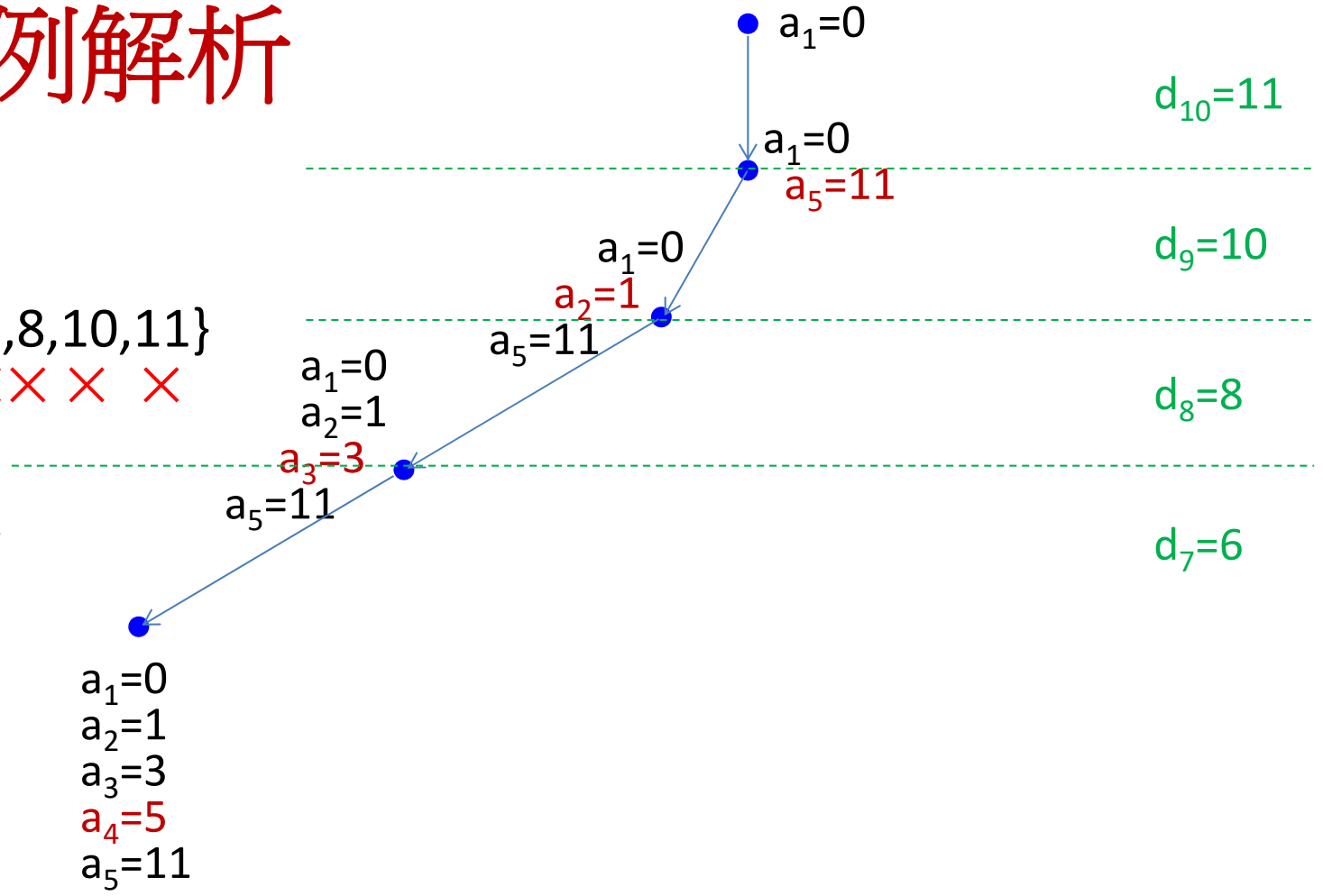
範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

✕✕✕　✕✕✕✕✕

將Δ內的數值
由小到大排好



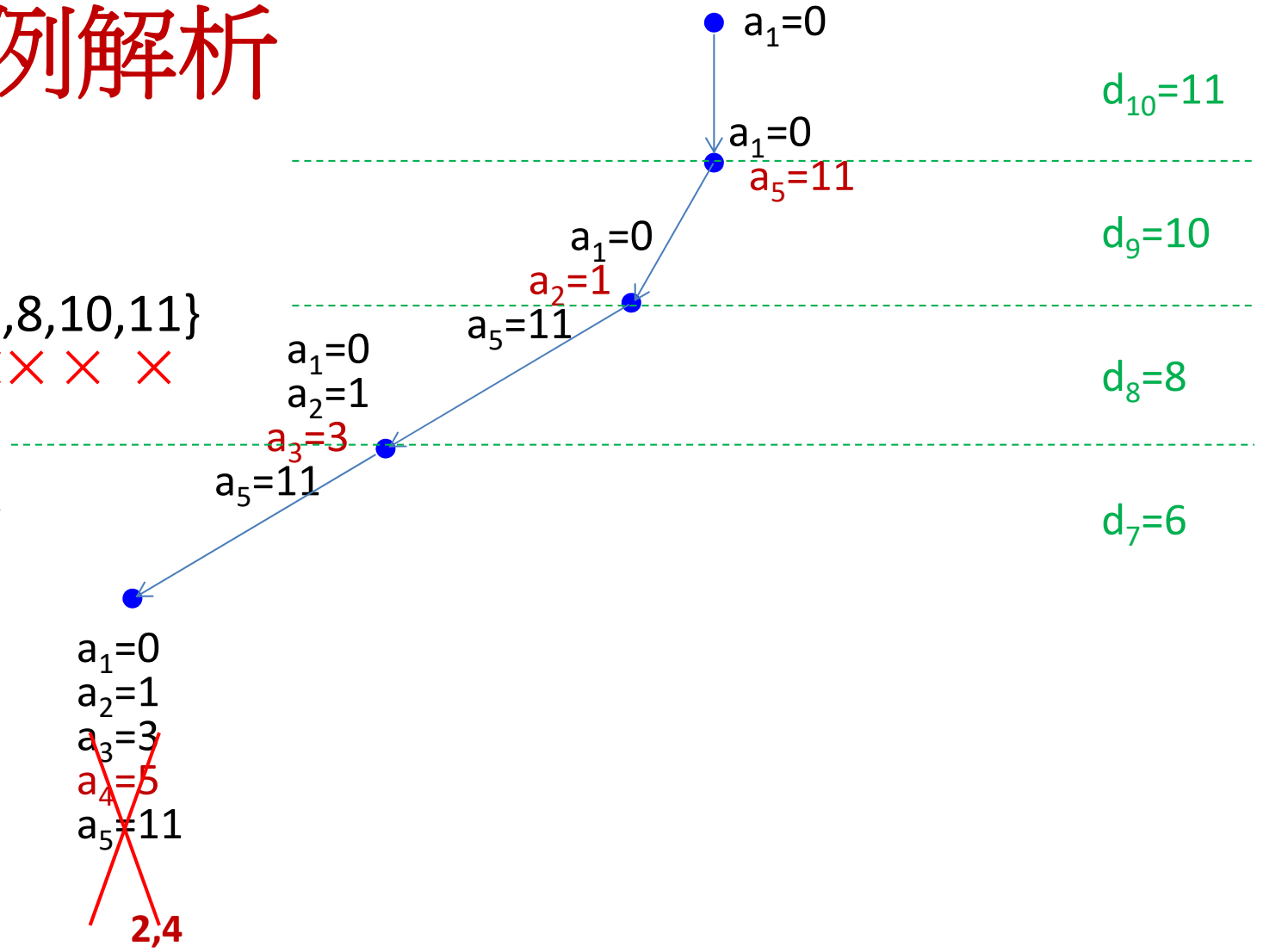
範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

✕✕✕　✕✕✕✕✕

將Δ內的數值
由小到大排好

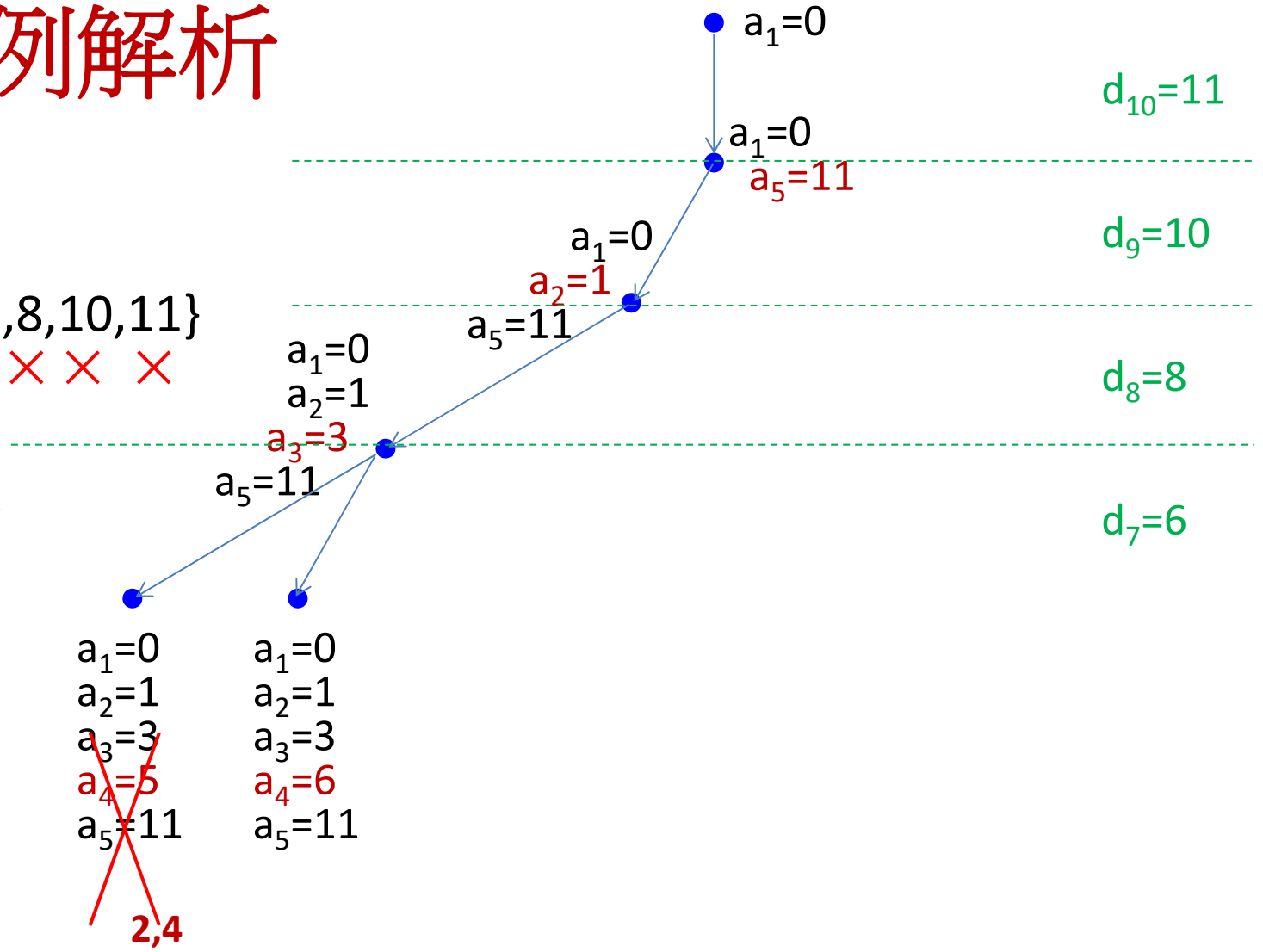


範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



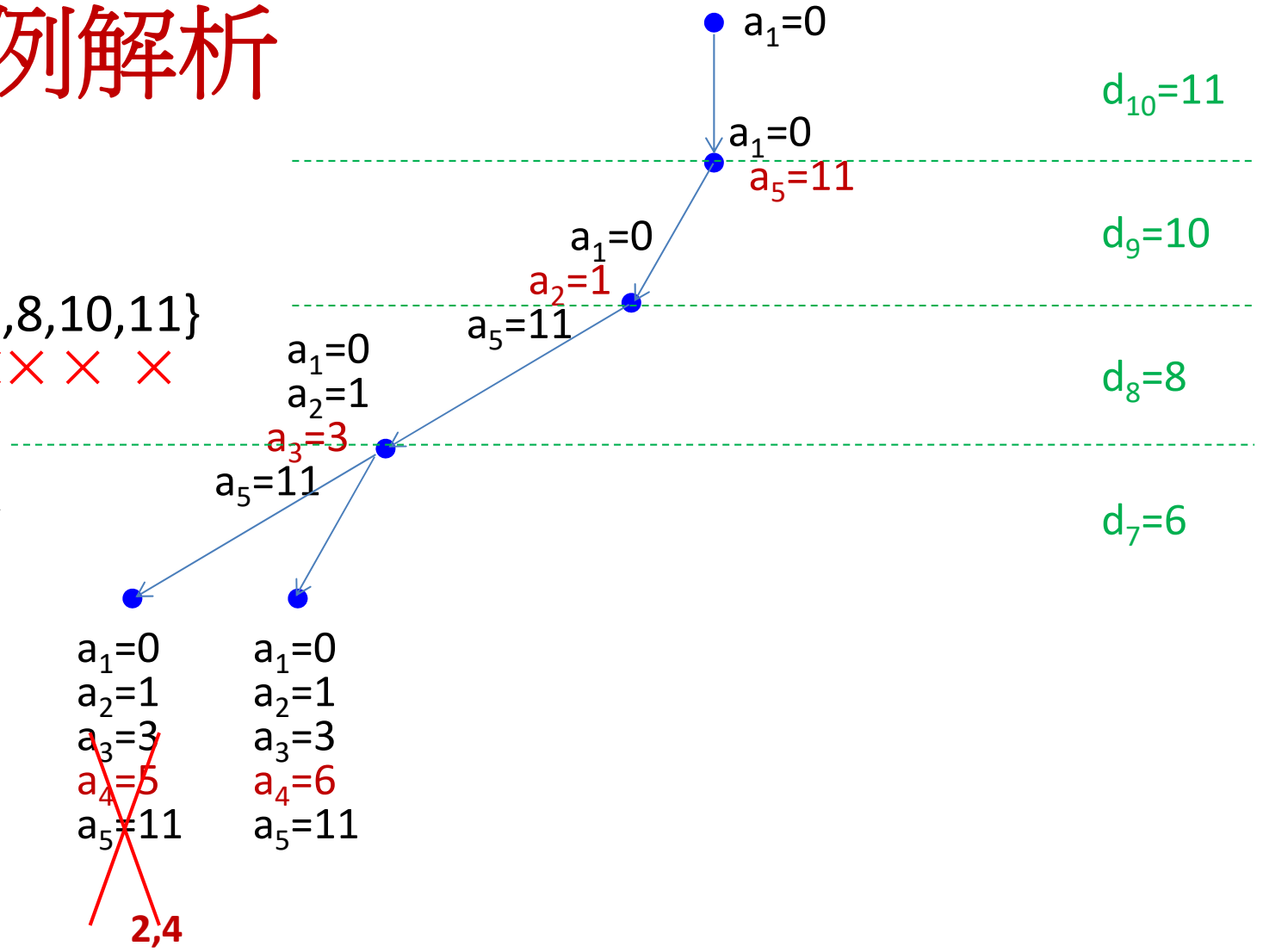
範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

× × × × × × × × × ×

將 Δ 內的數值
由小到大排好



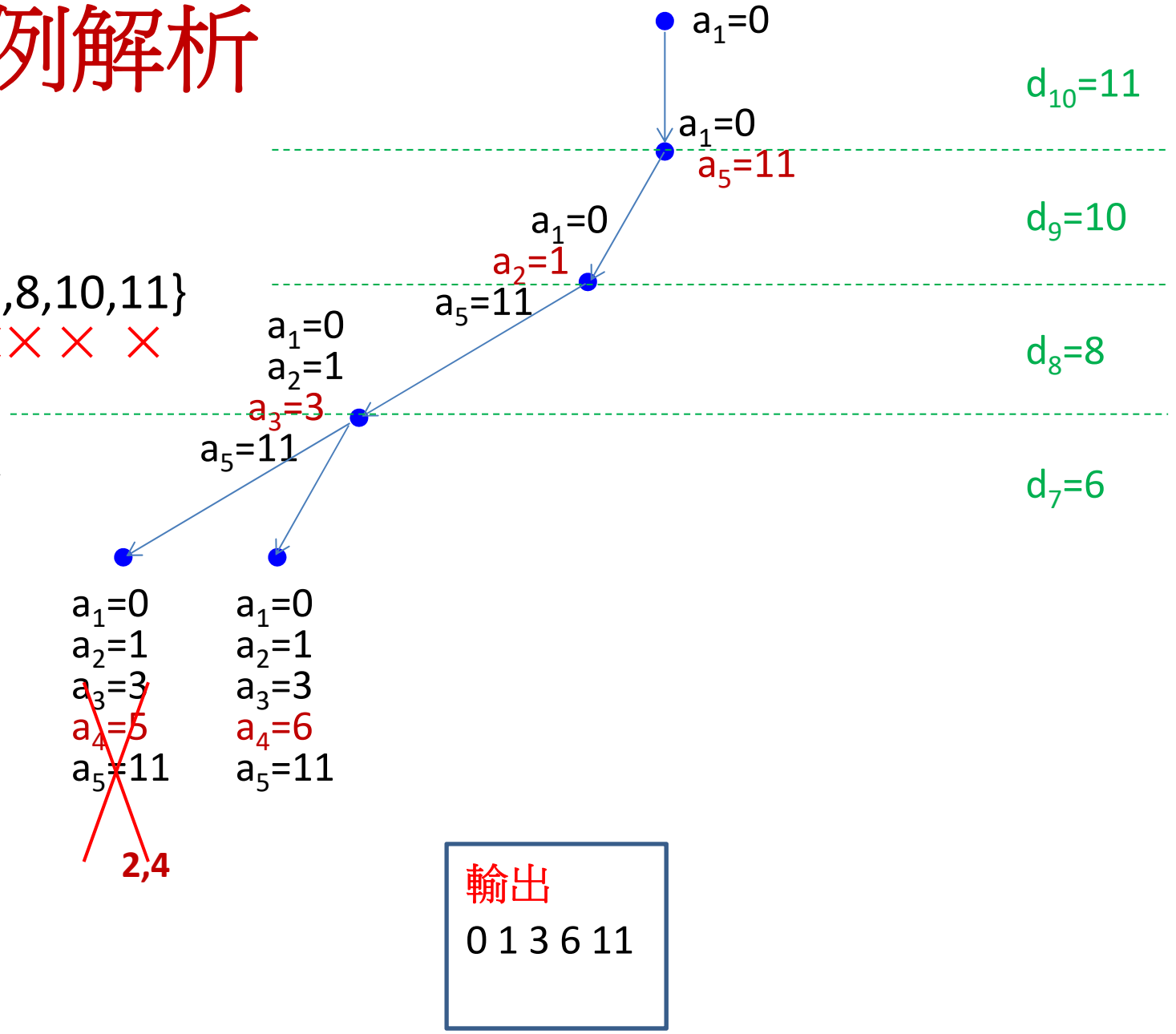
範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

✕✕✕✕✕✕✕✕ ✕ ✕

將 Δ 內的數值
由小到大排好



輸出

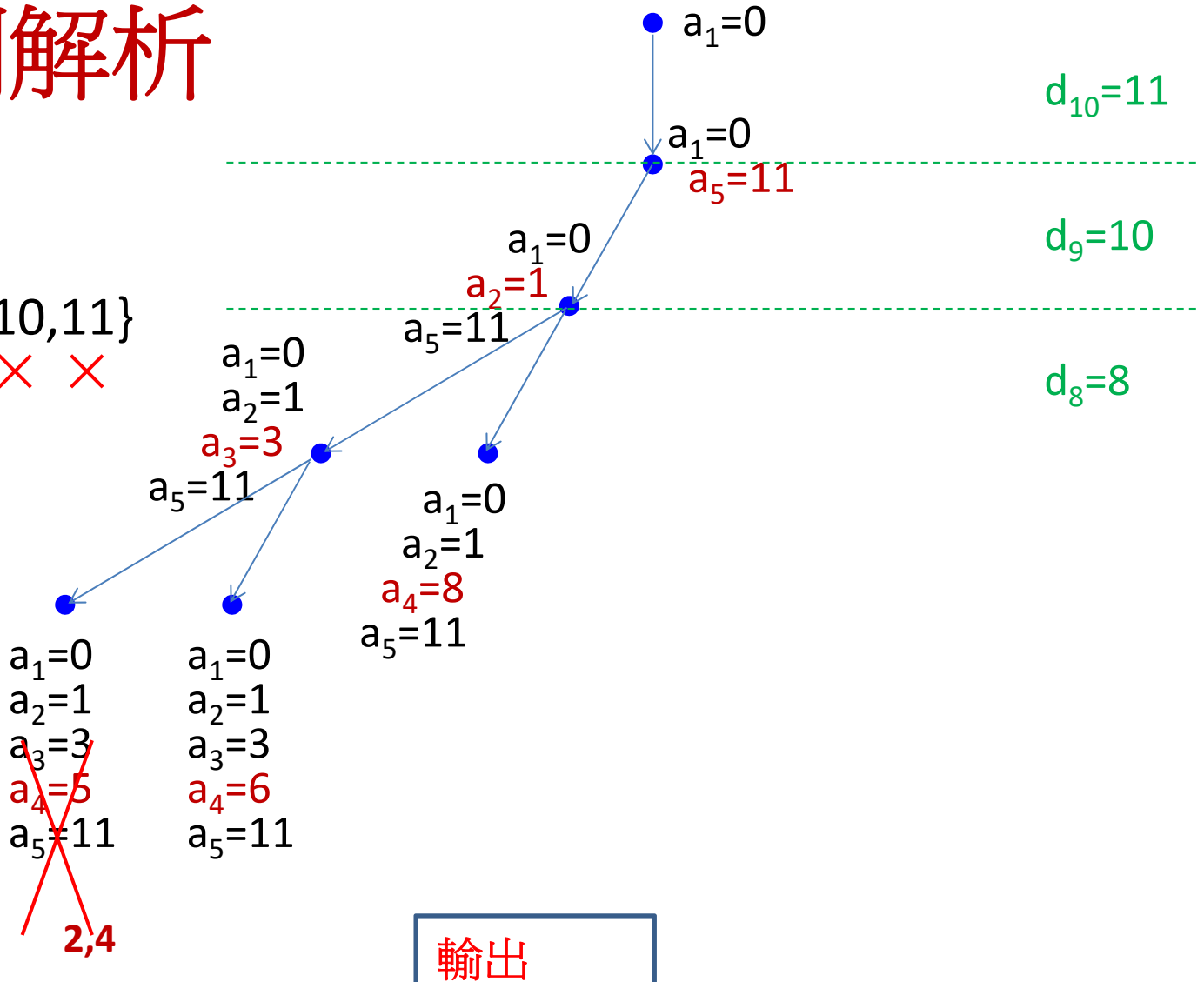
0 1 3 6 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

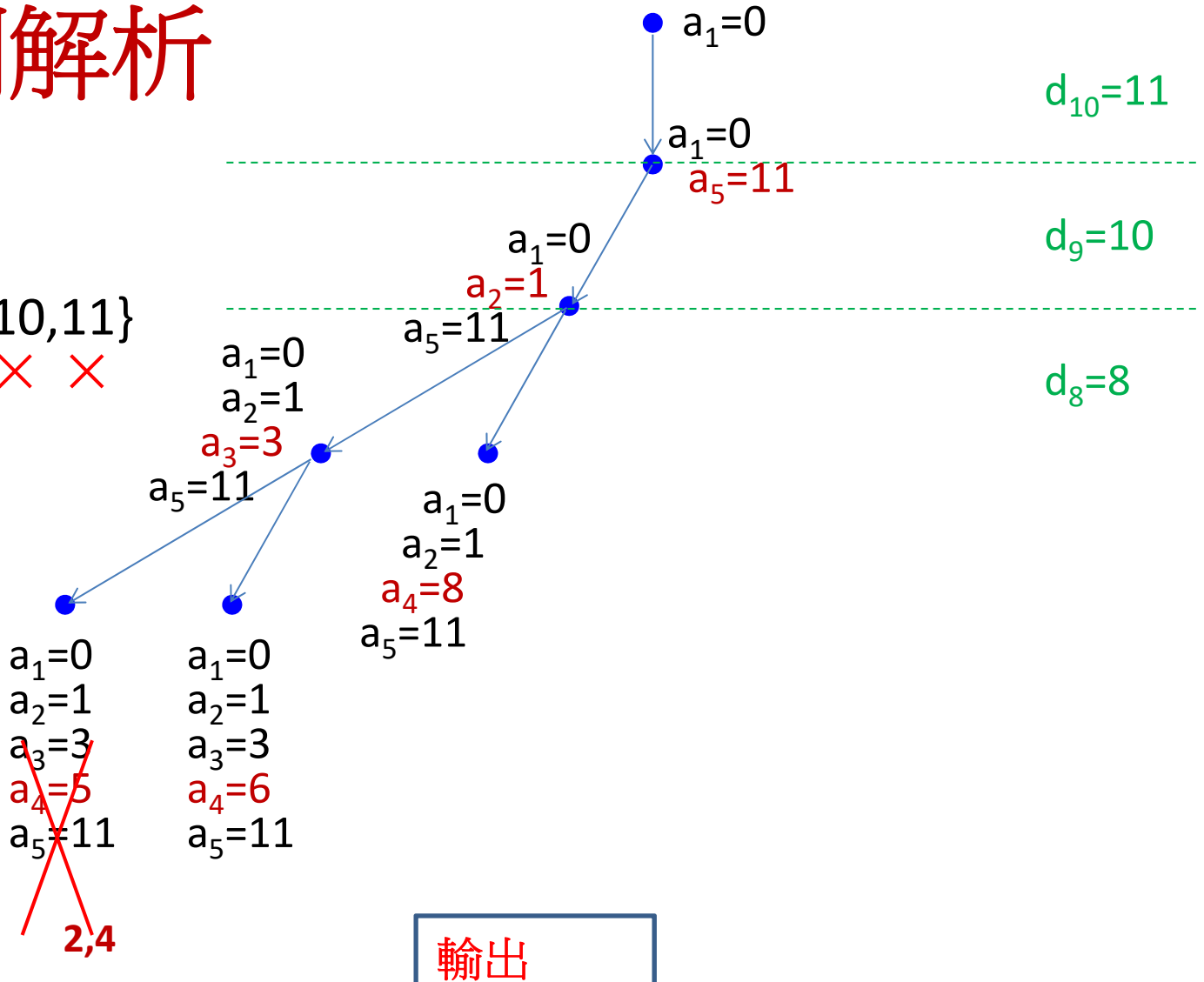
0 1 3 6 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

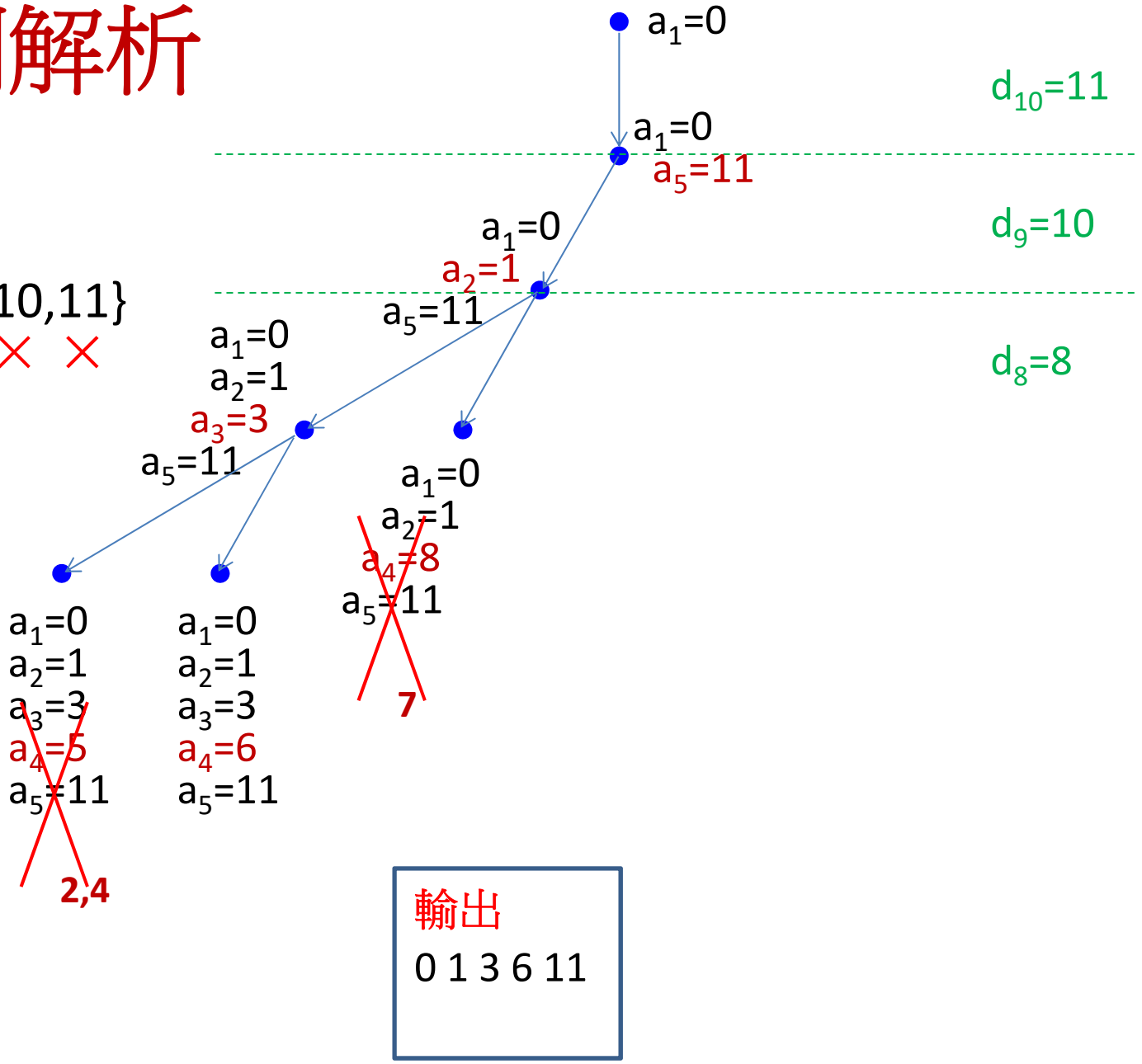
0 1 3 6 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

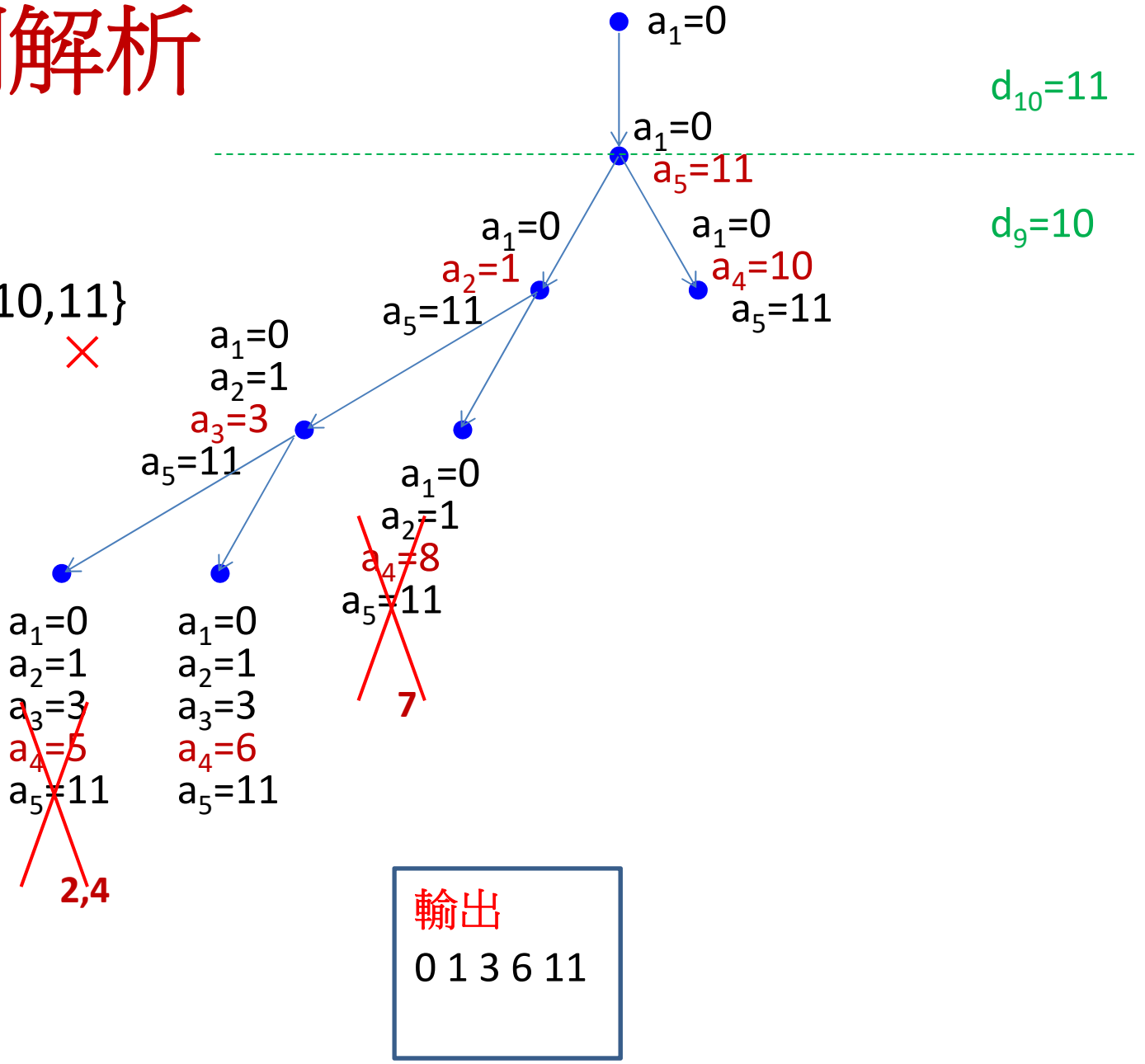
0 1 3 6 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

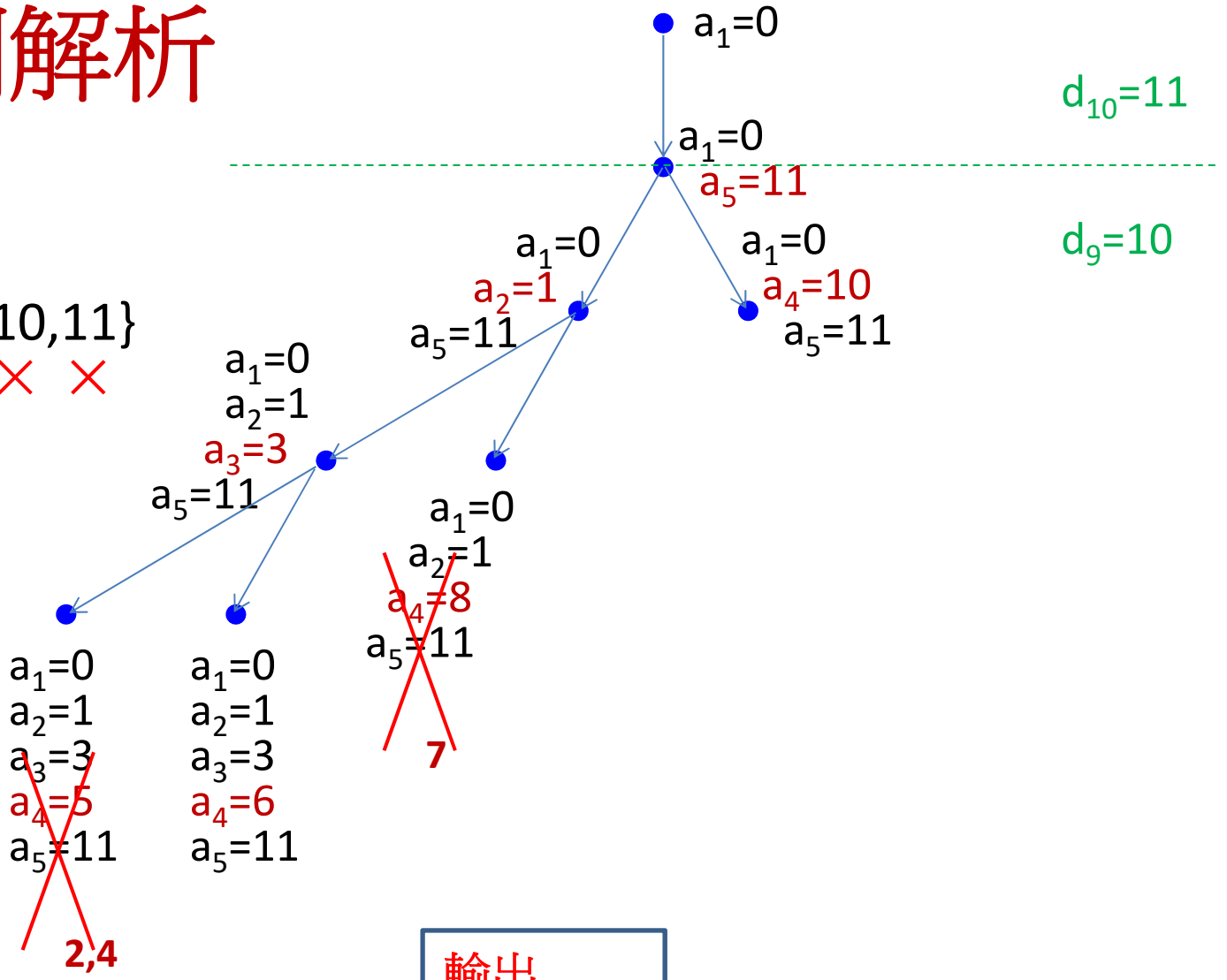
0 1 3 6 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

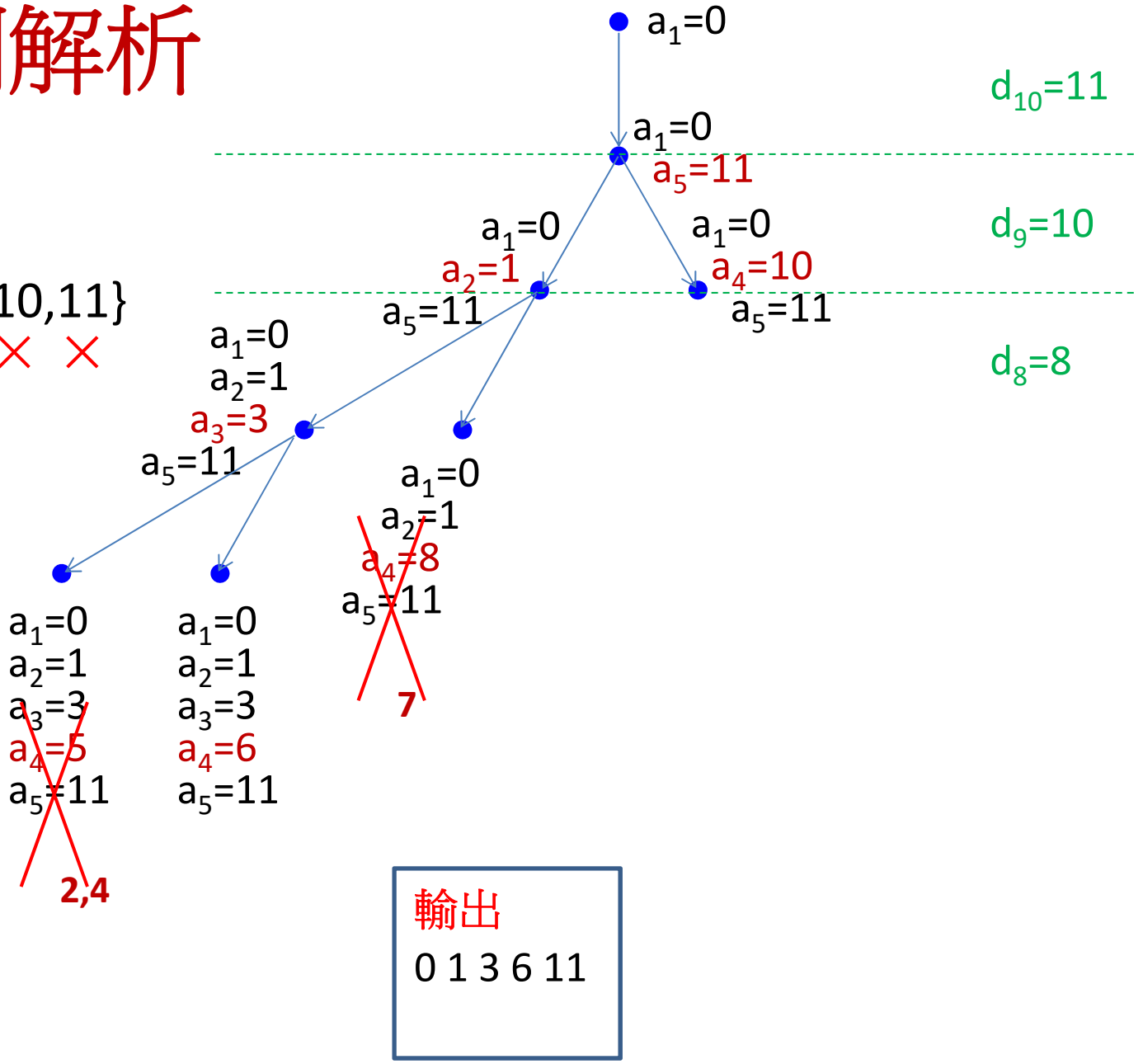
0 1 3 6 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好

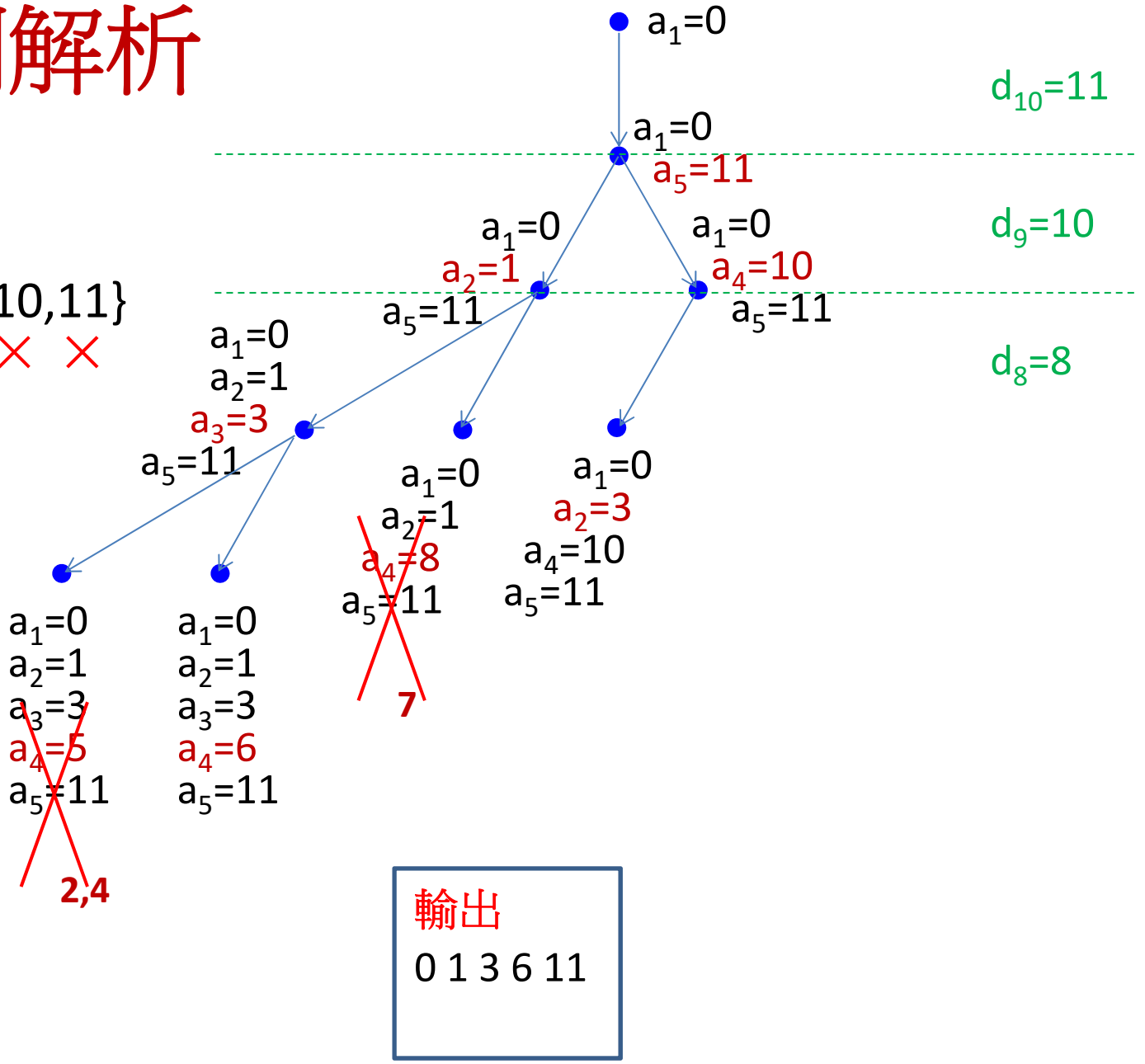


範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

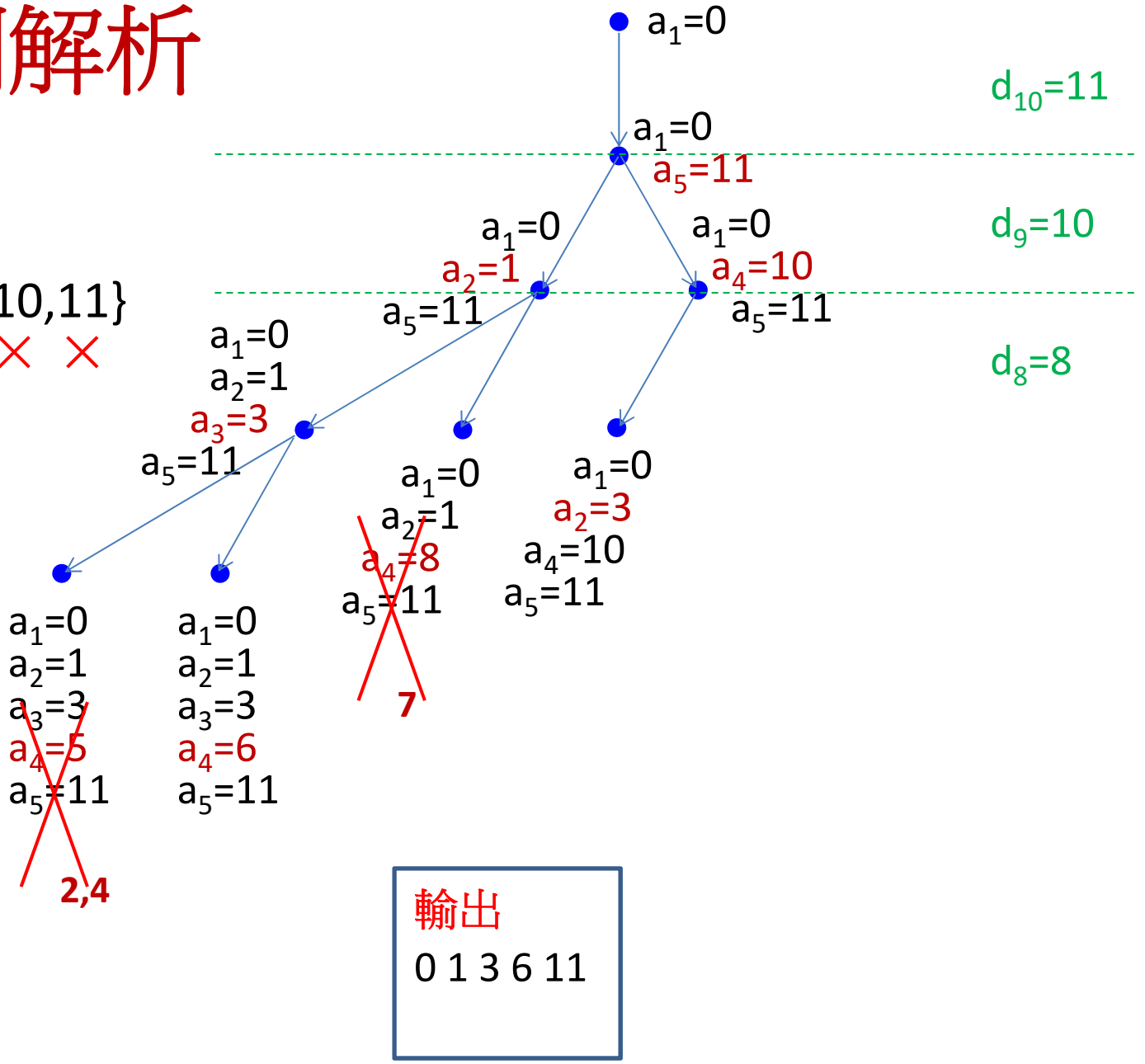
0 1 3 6 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

0 1 3 6 11

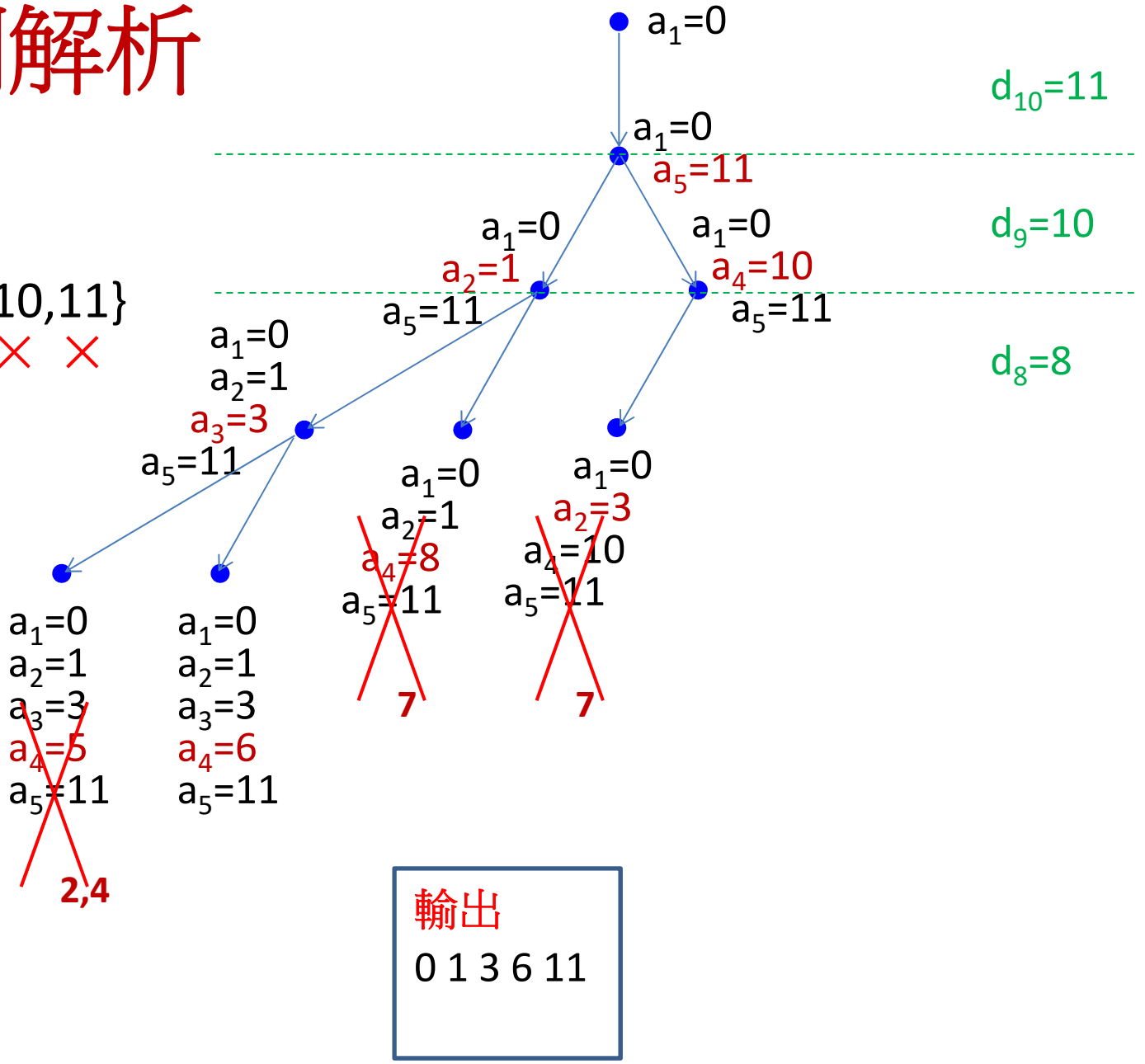
範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

× ×　　× × ×

將 Δ 內的數值
由小到大排好

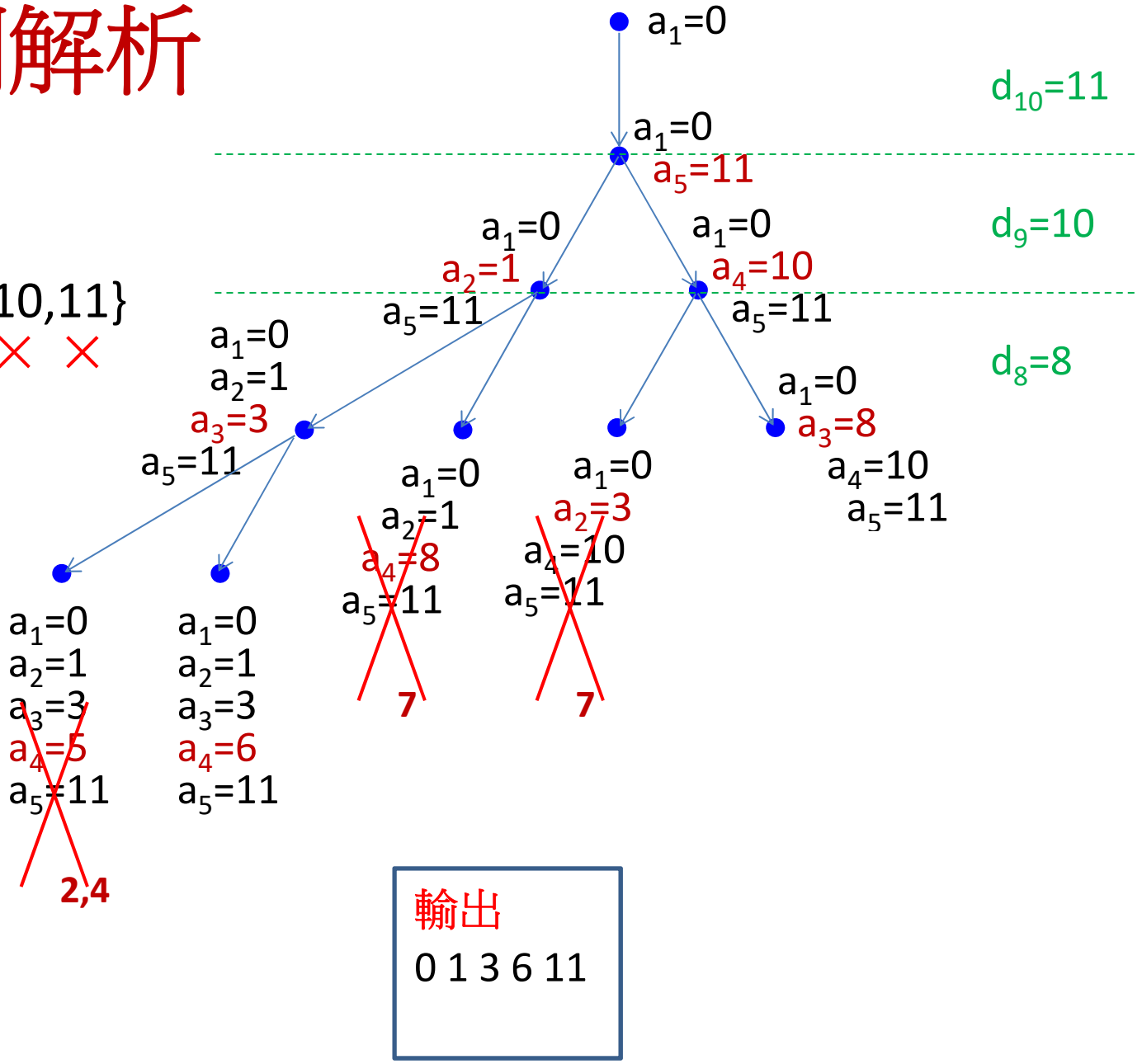


範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

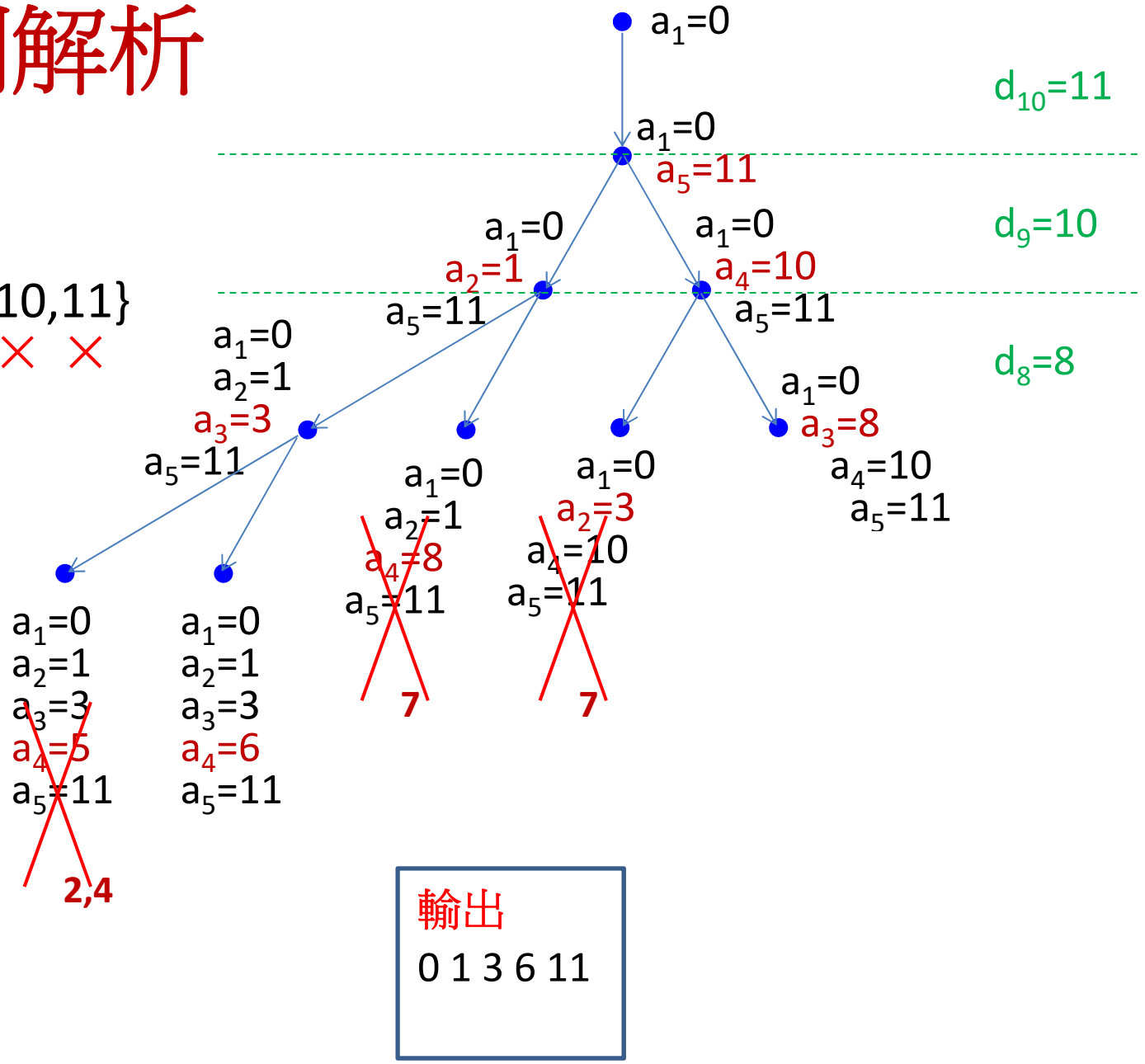
0 1 3 6 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

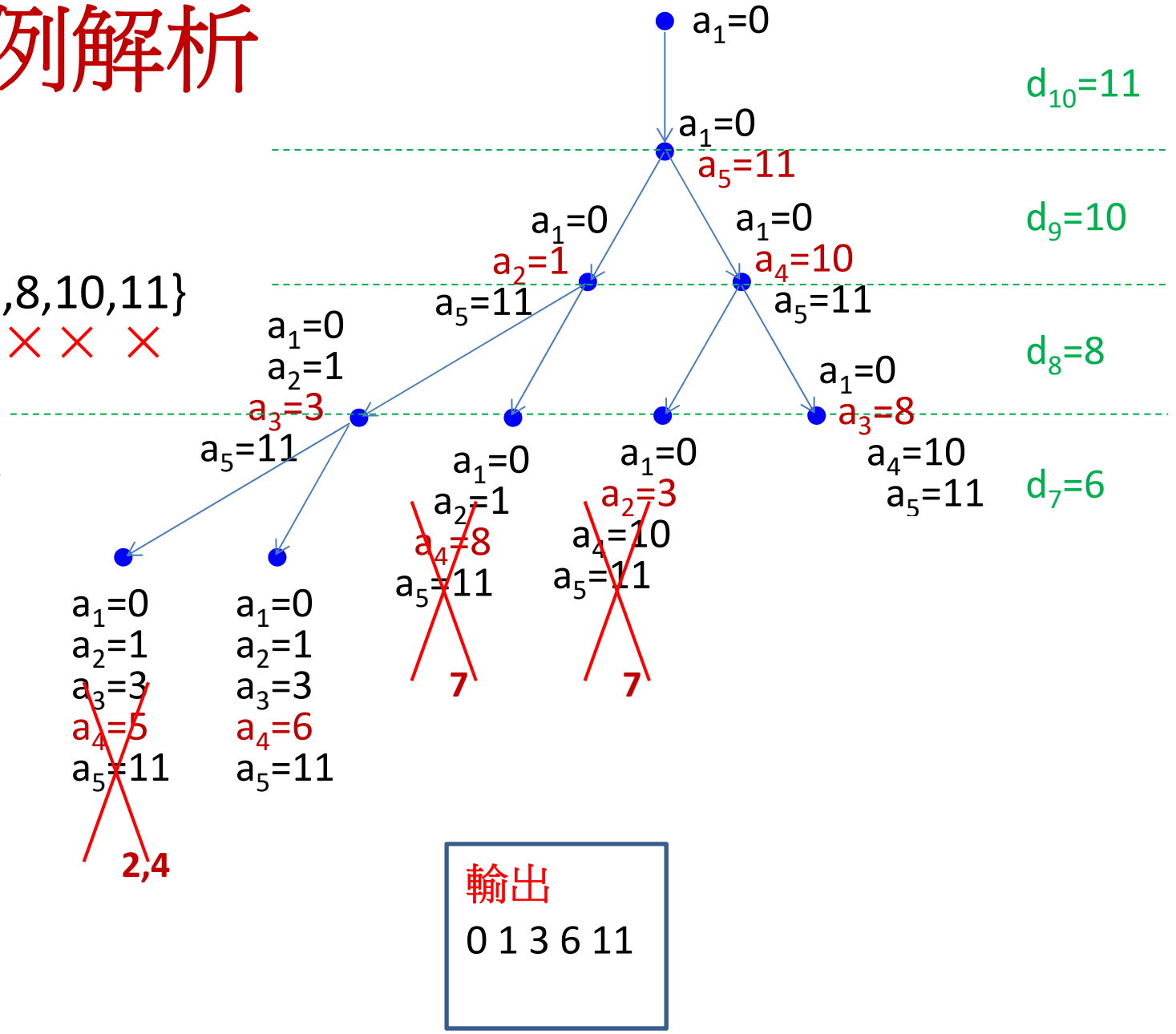
0 1 3 6 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

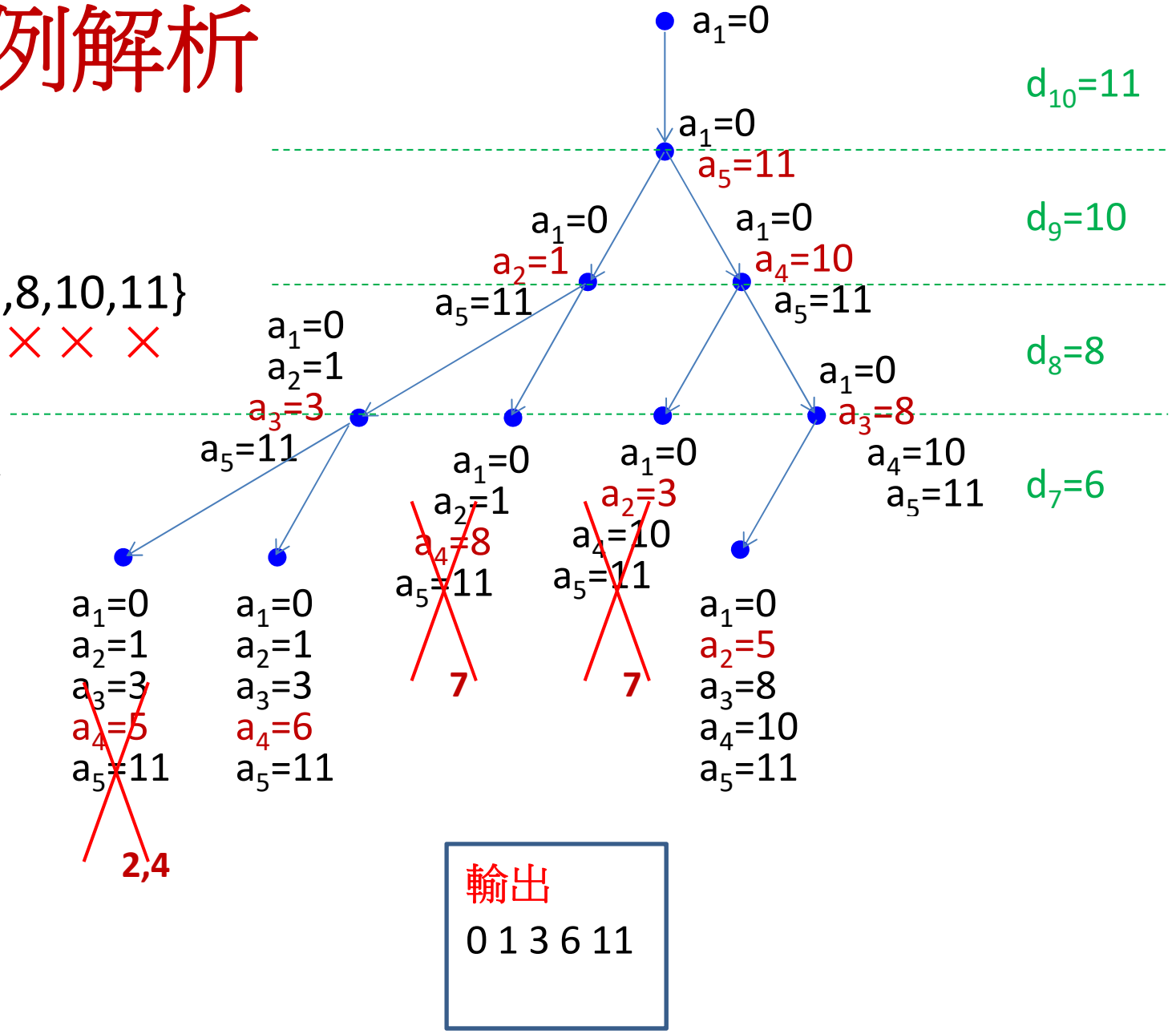
0 1 3 6 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

0 1 3 6 11

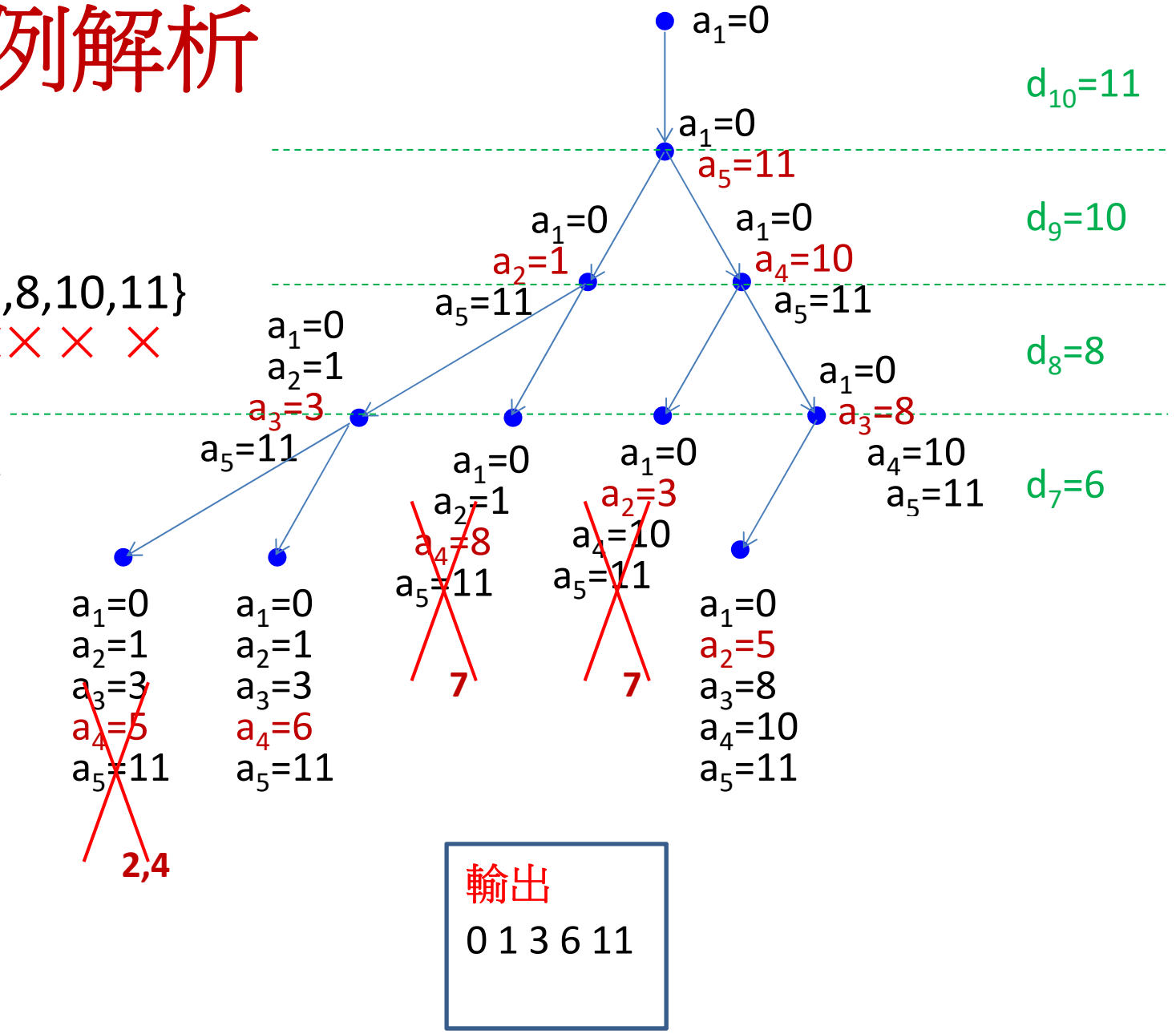
範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

× × × × × × × × × ×

將 Δ 內的數值
由小到大排好



輸出

0 1 3 6 11

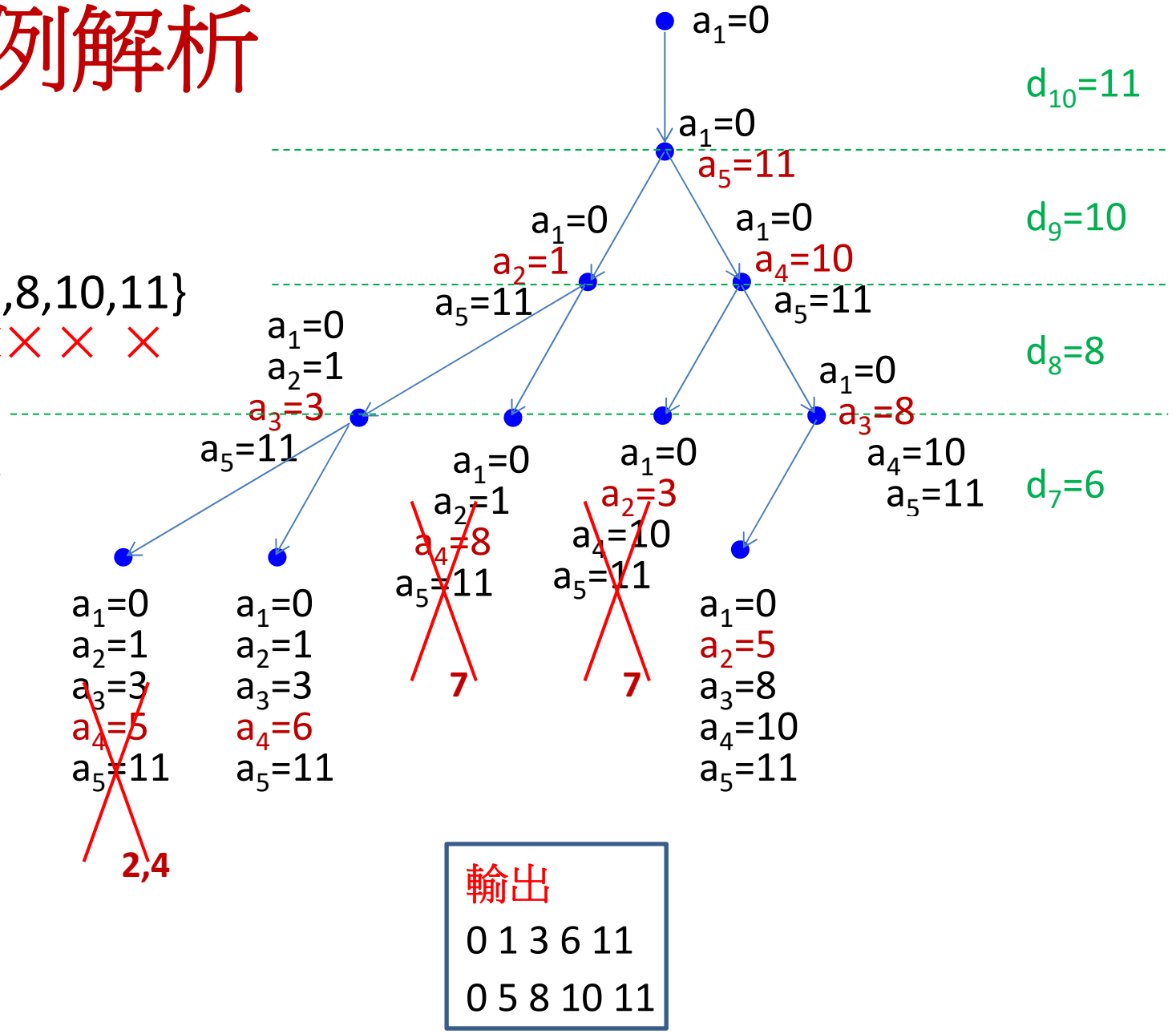
範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

× × × × × × × × × ×

將 Δ 內的數值
由小到大排好



輸出

0 1 3 6 11

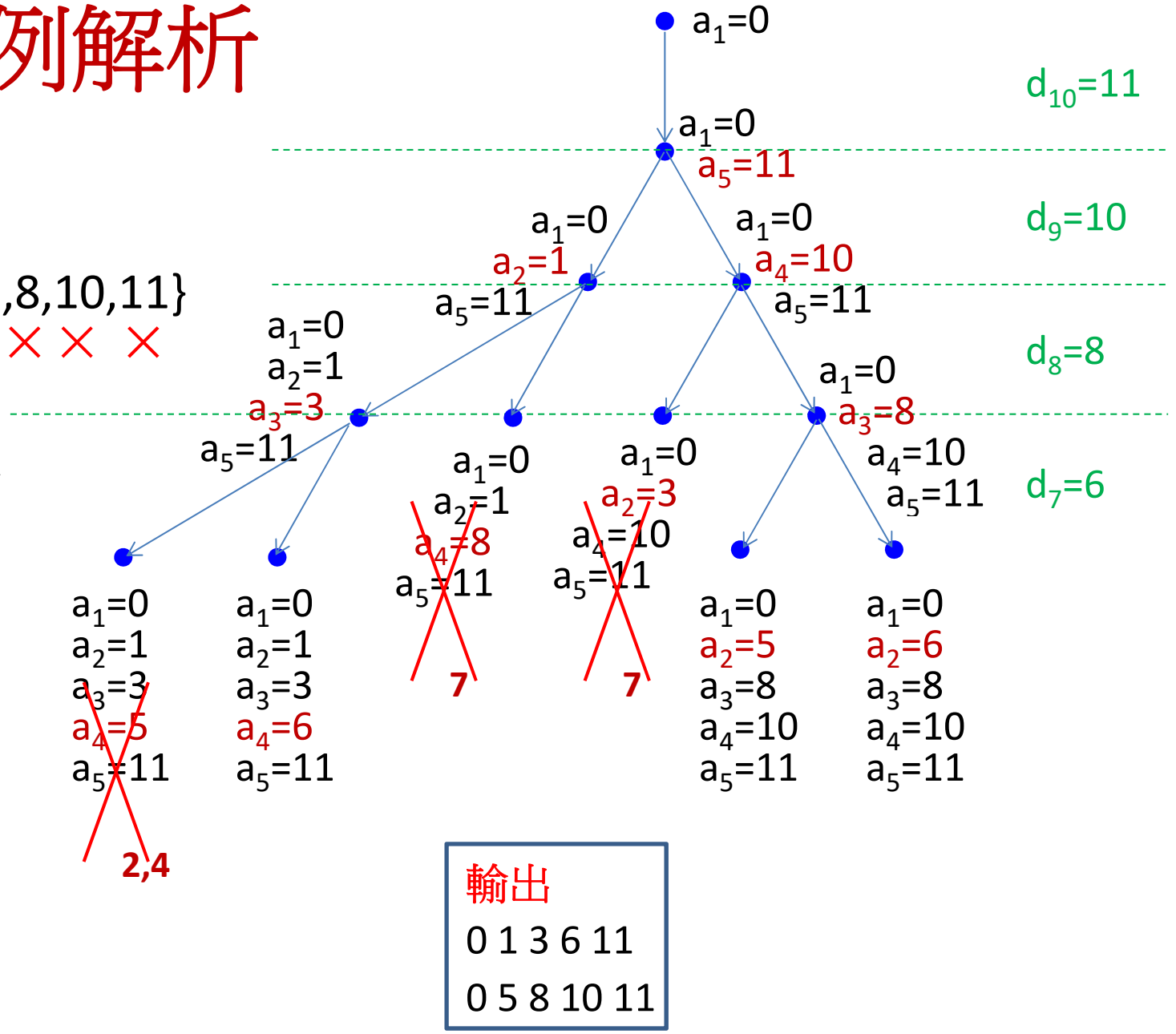
0 5 8 10 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值由小到大排好



輸出

0 1 3 6 11

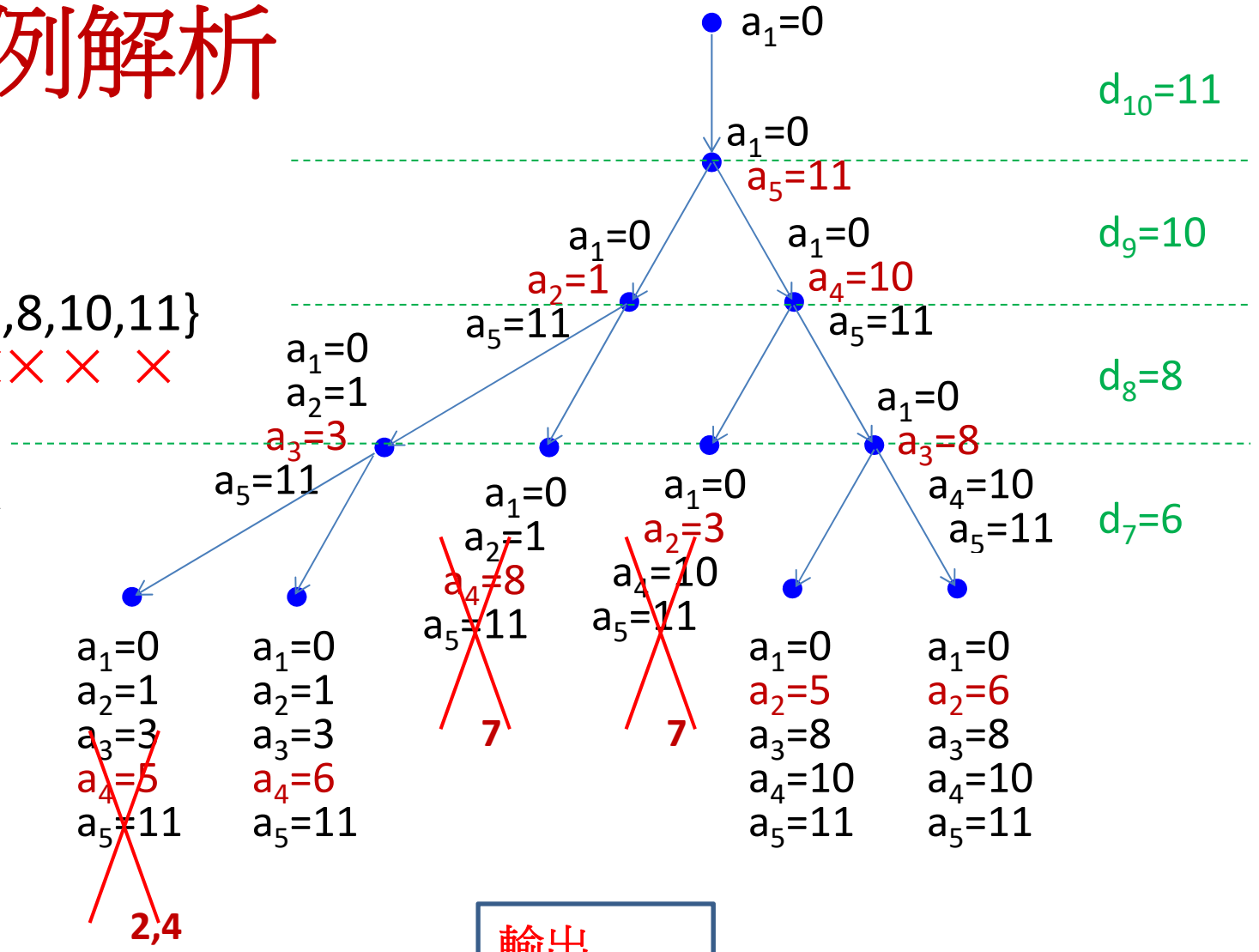
0 5 8 10 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}

將 Δ 內的數值
由小到大排好



輸出

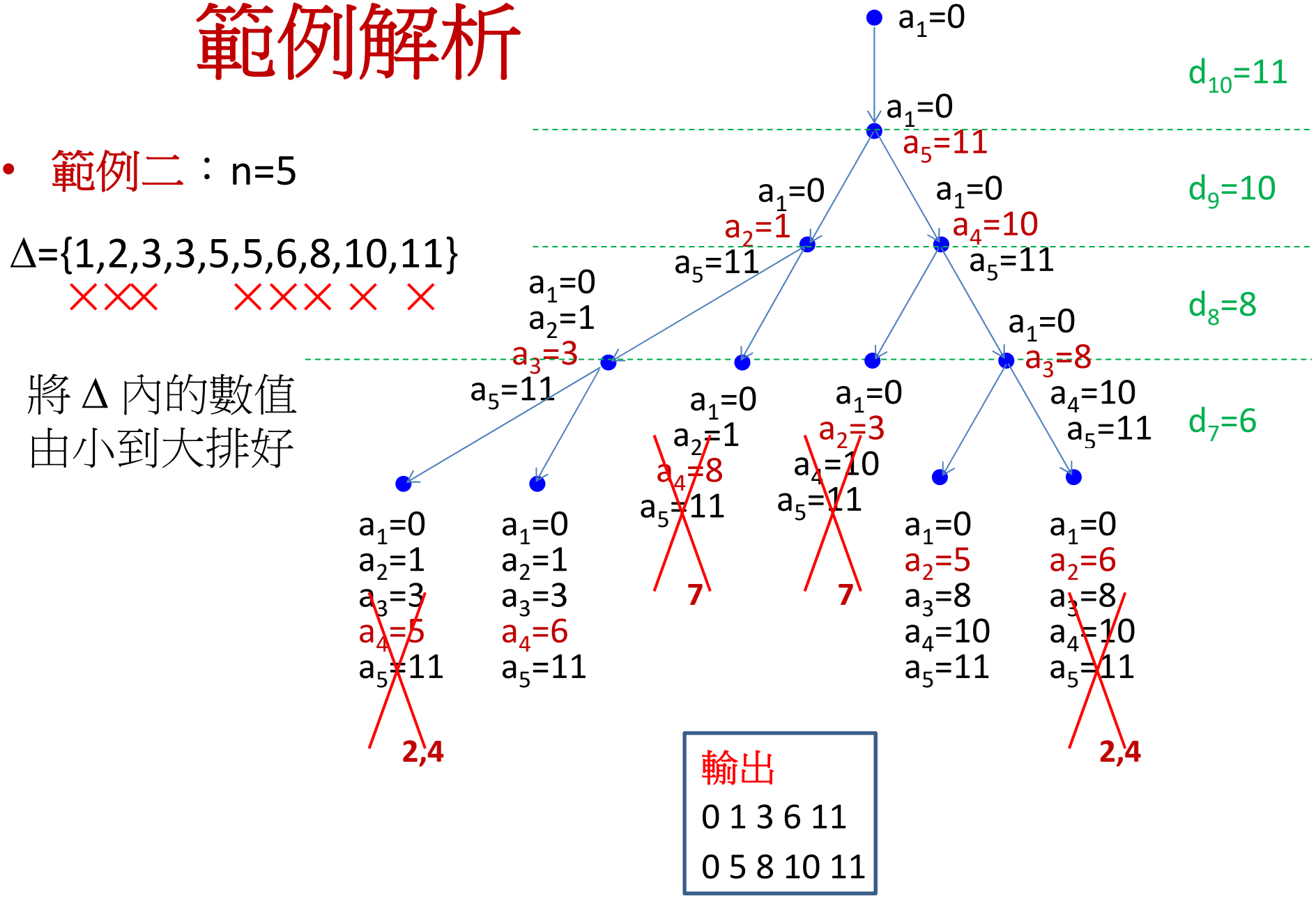
0 1 3 6 11

0 5 8 10 11

範例解析

- 範例二：n=5

Δ={1,2,3,3,5,5,6,8,10,11}



將 Δ 內的數值由小到大排好

輸出

0 1 3 6 11

0 5 8 10 11

合法性検査

合法性檢查

- 每次考量一個新的數字時，有兩種一致性需要檢查

合法性檢查

- 每次考量一個新的數字時，有兩種一致性需要檢查

  1. 這個數字和所有已經決定的數字的距離是否在 Δ 剩下的集合內，如果是的話，把那些距離扣除

合法性檢查

- 每次考量一個新的數字時，有兩種一致性需要檢查

  1. 這個數字和所有已經決定的數字的距離是否在 Δ 剩下的集合內，如果是的話，把那些距離扣除
  2. 這個數字和所有已經決定的數字是否相同 (在 Δ 剩下的集合內的最大值不只一個時需要檢查)

合法性檢查

- 每次考量一個新的數字時，有兩種一致性需要檢查

  1. 這個數字和所有已經決定的數字的距離是否在 Δ 剩下的集合內，如果是的話，把那些距離扣除
  2. 這個數字和所有已經決定的數字是否相同 (在 Δ 剩下的集合內的最大值不只一個時需要檢查)

- 實作：

合法性檢查

- 每次考量一個新的數字時，有兩種一致性需要檢查

  1. 這個數字和所有已經決定的數字的距離是否在 Δ 剩下的集合內，如果是的話，把那些距離扣除
  2. 這個數字和所有已經決定的數字是否相同 (在 Δ 剩下的集合內的最大值不只一個時需要檢查)

- 實作：

  因為 Δ 集合內的元素隨著 DFS 搜尋的進度不斷地減少

合法性檢查

- 每次考量一個新的數字時，有兩種一致性需要檢查

  1. 這個數字和所有已經決定的數字的距離是否在 Δ 剩下的集合內，如果是的話，把那些距離扣除
  2. 這個數字和所有已經決定的數字是否相同 (在 Δ 剩下的集合內的最大值不只一個時需要檢查)

- 實作：

  因為 Δ 集合內的元素隨著 DFS 搜尋的進度不斷地減少，

  如果 Δ 集合用一個陣列來實作，修改時會花比較多時間；

合法性檢查

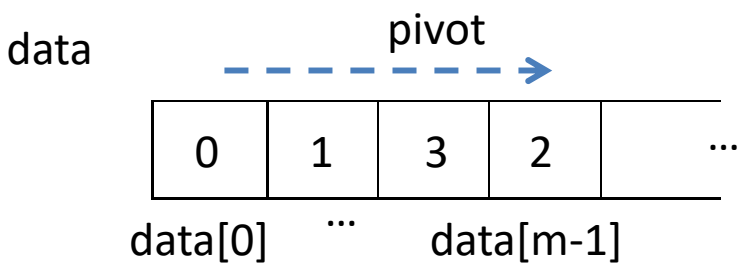
- 每次考量一個新的數字時，有兩種一致性需要檢查

  1. 這個數字和所有已經決定的數字的距離是否在 Δ 剩下的集合內，如果是的話，把那些距離扣除
  2. 這個數字和所有已經決定的數字是否相同 (在 Δ 剩下的集合內的最大值不只一個時需要檢查)

- 實作：

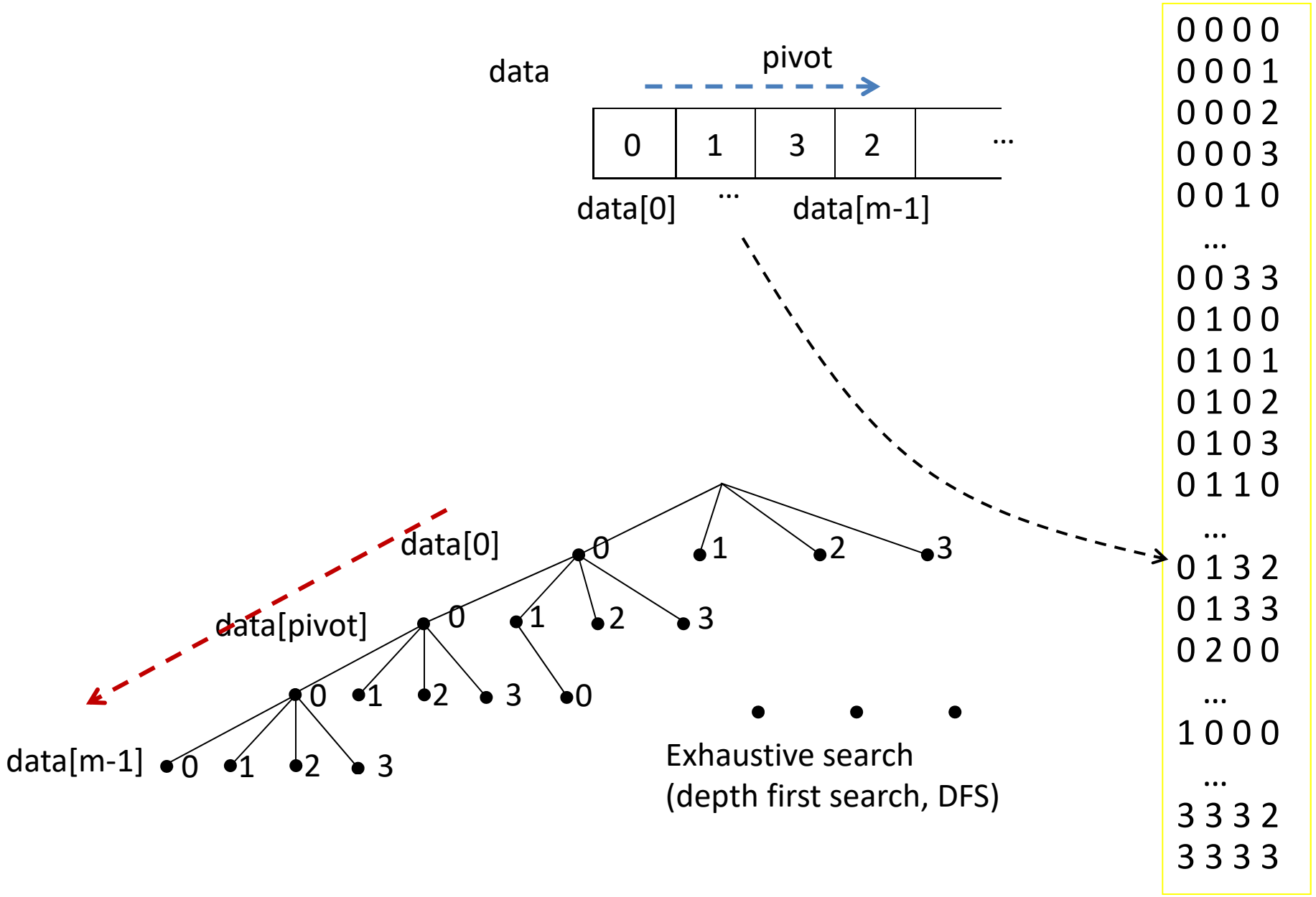
  因為 Δ 集合內的元素隨著 DFS 搜尋的進度不斷地減少，如果 Δ 集合用一個陣列來實作，修改時會花比較多時間；因為 Δ 集合裡面的距離限制在 [1,100] 之間，所以可以用一個整數陣列 int C[101]; 來紀錄每個距離出現的次數，如此可以快速地檢查 d_i 是否在 Δ 集合裡 … if ($C[d_i]>0$)

DFS 遞迴的樣板： 數數字

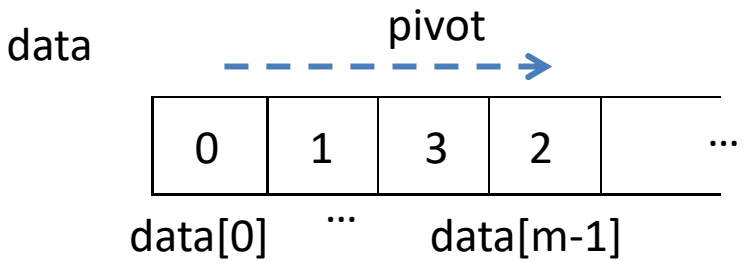


0 0 0 0
0 0 0 1
0 0 0 2
0 0 0 3
0 0 1 0
…
0 0 3 3
0 1 0 0
0 1 0 1
0 1 0 2
0 1 0 3
0 1 1 0
…
0 1 3 2
0 1 3 3
0 2 0 0
…
1 0 0 0
…
3 3 3 2
3 3 3 3

DFS 遞迴的樣板： 數數字



DFS 遞迴的樣板： 數數字




```
01 int main() {
02     int data[10];
03     countUp(4, data, 4, 0);
04     return 0;
05 }
06
07 void countUp(int b, int data[], int m, int pivot) {
08     if (pivot>=m)
09         print(data, m);
10     else
11         for (data[pivot]=0; data[pivot]<b; data[pivot]++)
12             countUp(b, data, m, pivot+1);
13 }
```

base

digits

the running digit

```
0 0 0 0
0 0 0 1
0 0 0 2
0 0 0 3
0 0 1 0
  …
0 0 3 3
0 1 0 0
0 1 0 1
0 1 0 2
0 1 0 3
0 1 1 0
  …
0 1 3 2
0 1 3 3
0 2 0 0
  …
1 0 0 0
  …
3 3 3 2
3 3 3 3
```

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

DFS 實作

x[0]~x[n-1]

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

DFS 實作

x[0]~x[n-1]

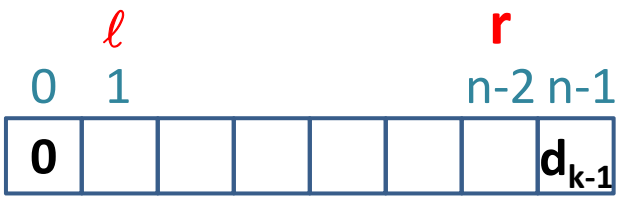
DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

x[0]~x[n-1]




```
int dfs(int n, char d[], int k, int ℓ, int r, char C[], int x[]) {


   if (ℓ>r) return 1;









   return 0;
}
```

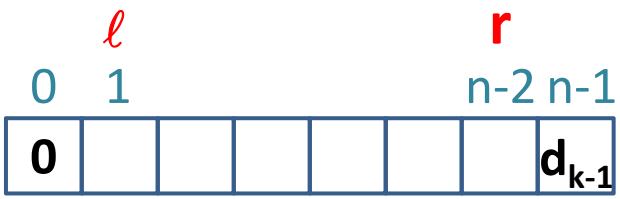
DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

x[0]~x[n-1]




```
int dfs(int n, char d[], int k, int ℓ, int r, char C[], int x[]) {
    int i,j,c;
    while (k>0&&C[d[k]]==0) k--;
    if (ℓ>r) return 1;
    x[ℓ] = x[n-1]-d[k];
    if (!isBad(n,x[ℓ],ℓ,r,C,x)) {
        modifyC(n,x[ℓ],ℓ,r,C,x,-1);
        if (dfs(n,d, k-1,ℓ+1,r,C,x)) return 1;
        modifyC(n,x[ℓ],ℓ,r,C,x,1);
    }
```

左子樹

```
    return 0;
}
```

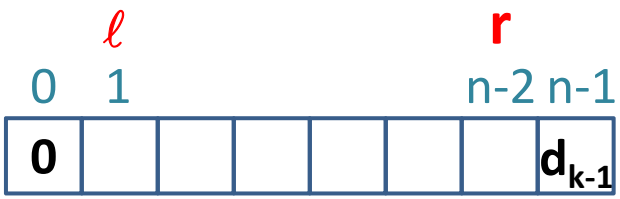
DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

x[0]~x[n-1]




```
int dfs(int n, char d[], int k, int ℓ, int r, char C[], int x[]) {
    int i,j,c;
    while (k>0&&C[d[k]]==0) k--;
    if (ℓ>r) return 1;
    x[ℓ] = x[n-1]-d[k];
    if (!isBad(n,x[ℓ],ℓ,r,C,x)) {
        modifyC(n,x[ℓ],ℓ,r,C,x,-1);
        if (dfs(n,d, k-1,ℓ+1,r,C,x)) return 1;
        modifyC(n,x[ℓ],ℓ,r,C,x,1);
    }

    return 0;
}
```

原則上由大到小一個一個考量 **d[k]**，但是有的時候已經從 Δ 中移除

左子樹

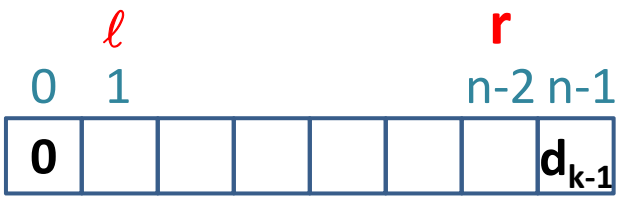
DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

x[0]~x[n-1]




```
int dfs(int n, char d[], int k, int ℓ, int r, char C[], int x[]) {
    int i,j,c;
    while (k>0&&C[d[k]]==0) k--;
    if (ℓ>r) return 1;
    x[ℓ] = x[n-1]-d[k];
    if (!isBad(n,x[ℓ],ℓ,r,C,x)) {
        modifyC(n,x[ℓ],ℓ,r,C,x,-1);
        if (dfs(n,d, k-1,ℓ+1,r,C,x)) return 1;
        modifyC(n,x[ℓ],ℓ,r,C,x,1);
    }

    return 0;
}
```

原則上由大到小一個一個考量 **d[k]**，但是有的時候已經從 Δ 中移除

新增 **x[ℓ]** 時需要符合 Δ 的限制

左子樹

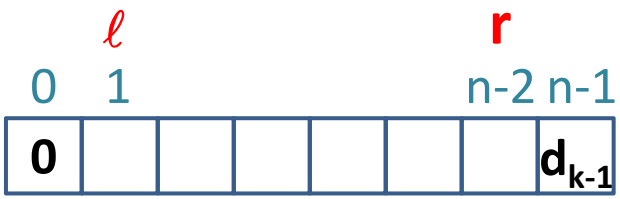
DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

x[0]~x[n-1]




```
int dfs(int n, char d[], int k, int ℓ, int r, char C[], int x[]) {
   int i,j,c;
   while (k>0&&C[d[k]]==0) k--;
   if (ℓ>r) return 1;
   x[ℓ] = x[n-1]-d[k];
   if (!isBad(n,x[ℓ],ℓ,r,C,x)) {
      modifyC(n,x[ℓ],ℓ,r,C,x,-1);
      if (dfs(n,d, k-1,ℓ+1,r,C,x)) return 1;
      modifyC(n,x[ℓ],ℓ,r,C,x,1);
   }
   return 0;
}
```

原則上由大到小一個一個考量 **d[k]**，但是有的時候已經從 Δ 中移除

新增 **x[ℓ]** 時需要符合 Δ 的限制

從 Δ 中移除因 **x[ℓ]** 產生的距離

左子樹

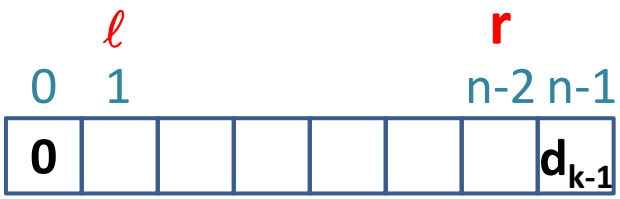
DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

x[0]~x[n-1]




```
int dfs(int n, char d[], int k, int ℓ, int r, char C[], int x[]) {
   int i,j,c;
   while (k>0&&C[d[k]]==0) k--;
   if (ℓ>r) return 1;
   x[ℓ] = x[n-1]-d[k];
   if (!isBad(n,x[ℓ],ℓ,r,C,x)) {
      modifyC(n,x[ℓ],ℓ,r,C,x,-1);
      if (dfs(n,d, k-1,ℓ+1,r,C,x)) return 1;
      modifyC(n,x[ℓ],ℓ,r,C,x,1);
   }
   return 0;
}
```

原則上由大到小一個一個考量 **d[k]**，但是有的時候已經從 Δ 中移除

新增 **x[ℓ]** 時需要符合 Δ 的限制

從 Δ 中移除因 **x[ℓ]** 產生的距離

將 **x[ℓ]** 產生的距離加回 Δ 集合中

左子樹

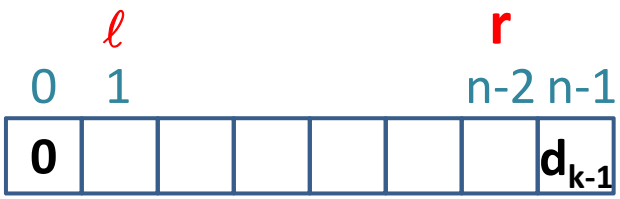
DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

x[0]~x[n-1]




```
int dfs(int n, char d[], int k, int ℓ, int r, char C[], int x[]) {
   int i,j,c;
   while (k>0&&C[d[k]]==0) k--;
   if (ℓ>r) return 1;
   x[ℓ] = x[n-1]-d[k];
   if (!isBad(n,x[ℓ],ℓ,r,C,x)) {
      modifyC(n,x[ℓ],ℓ,r,C,x,-1);
      if (dfs(n,d, k-1,ℓ+1,r,C,x)) return 1;
      modifyC(n,x[ℓ],ℓ,r,C,x,1);
   }
   x[r] = d[k];
   if (isBad(n,x[r],ℓ,r,C,x)) return 0;
   modifyC(n,x[r],ℓ,r,C,x,-1);
   if (dfs(n,d, k-1,ℓ,r-1,C,x)) return 1;
   modifyC(n,x[r],ℓ,r,C,x,1);
   return 0;
}
```

原則上由大到小一個一個考量 **d[k]**，但是有的時候已經從 Δ 中移除

新增 **x[ℓ]** 時需要符合 Δ 的限制

從 Δ 中移除因 **x[ℓ]** 產生的距離

將 **x[ℓ]** 產生的距離加回 Δ 集合中

左子樹

右子樹

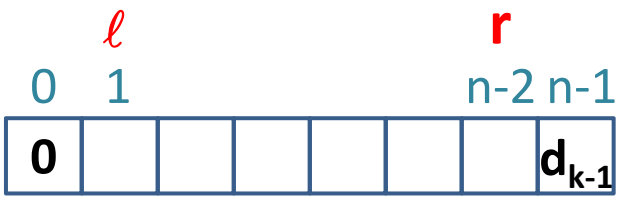
DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

x[0]~x[n-1]




```
int dfs(int n, char d[], int k, int ℓ, int r, char C[], int x[]) {
   int i,j,c;
   while (k>0&&C[d[k]]==0) k--;
   if (ℓ>r) return 1;
   x[ℓ] = x[n-1]-d[k];
   if (!isBad(n,x[ℓ],ℓ,r,C,x)) {
      modifyC(n,x[ℓ],ℓ,r,C,x,-1);
      if (dfs(n,d, k-1,ℓ+1,r,C,x)) return 1;
      modifyC(n,x[ℓ],ℓ,r,C,x,1);
   }
   x[r] = d[k];
   if (isBad(n,x[r],ℓ,r,C,x)) return 0;
   modifyC(n,x[r],ℓ,r,C,x,-1);
   if (dfs(n,d, k-1,ℓ,r-1,C,x)) return 1;
   modifyC(n,x[r],ℓ,r,C,x,1);
   return 0;
}
```

原則上由大到小一個一個考量 **d[k]**，但是有的時候已經從 Δ 中移除

新增 **x[ℓ]** 時需要符合 Δ 的限制

從 Δ 中移除因 **x[ℓ]** 產生的距離

將 **x[ℓ]** 產生的距離加回 Δ 集合中

左子樹

新增 **x[r]** 時需要符合 Δ 的限制

右子樹

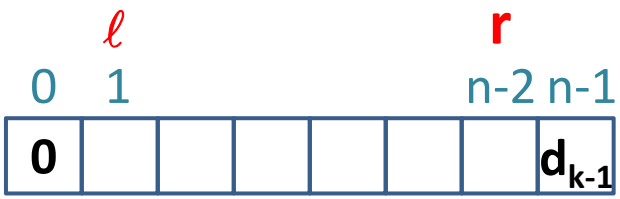
DFS 實作

ascending d_i

d[0]~d[k-1], k=n(n-1)/2

C[1]~C[100] count of d_i

x[0]~x[n-1]




```
int dfs(int n, char d[], int k, int ℓ, int r, char C[], int x[]) {
    int i,j,c;
    while (k>0&&C[d[k]]==0) k--;
    if (ℓ>r) return 1;
    x[ℓ] = x[n-1]-d[k];
    if (!isBad(n,x[ℓ],ℓ,r,C,x)) {
        modifyC(n,x[ℓ],ℓ,r,C,x,-1);
        if (dfs(n,d, k-1,ℓ+1,r,C,x)) return 1;
        modifyC(n,x[ℓ],ℓ,r,C,x,1);
    }
    x[r] = d[k];
    if (isBad(n,x[r],ℓ,r,C,x)) return 0;
    modifyC(n,x[r],ℓ,r,C,x,-1);
    if (dfs(n,d, k-1,ℓ,r-1,C,x)) return 1;
    modifyC(n,x[r],ℓ,r,C,x,1);
    return 0;
}
```

原則上由大到小一個一個考量 **d[k]**，但是有的時候已經從 Δ 中移除

新增 **x[ℓ]** 時需要符合 Δ 的限制

從 Δ 中移除因 **x[ℓ]** 產生的距離

將 **x[ℓ]** 產生的距離加回 Δ 集合中

左子樹

新增 **x[r]** 時需要符合 Δ 的限制

右子樹

x[n-1]=d[k-1], C[d[k-1]]=0, dfs(n,d, k-2,1,n-2,C,x);

擴充功能 - 找出所有答案

擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋

擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案

擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式

擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

```
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
  70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
  41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
  32 14 55 18 23 41
```

擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序

1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86

擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86
0

擴充功能 - 找出所有答案

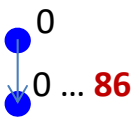
- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86
●0
d_{65}=86

擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86

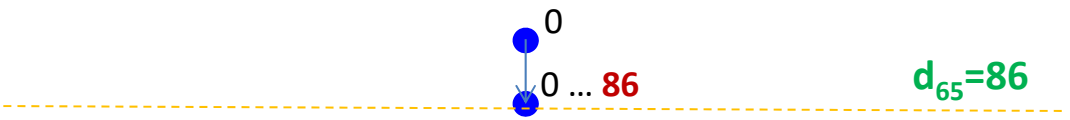


d_{65}=86

擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

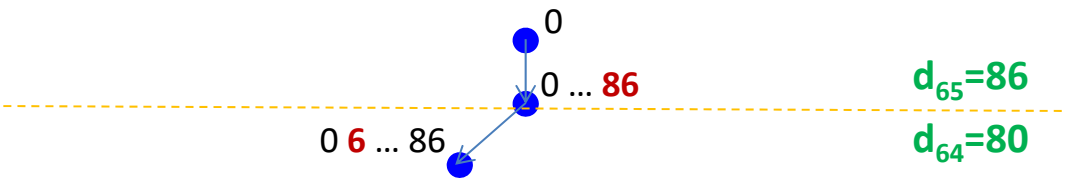
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

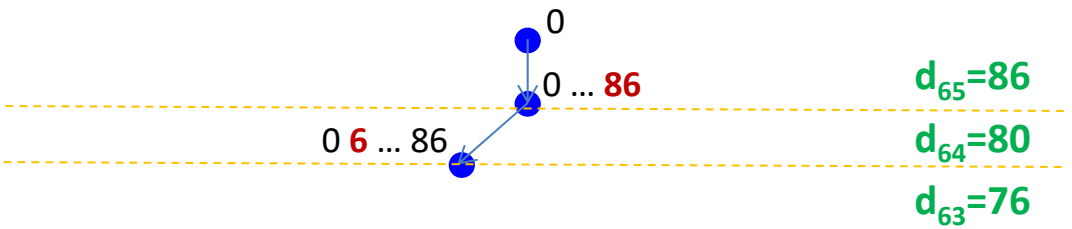
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

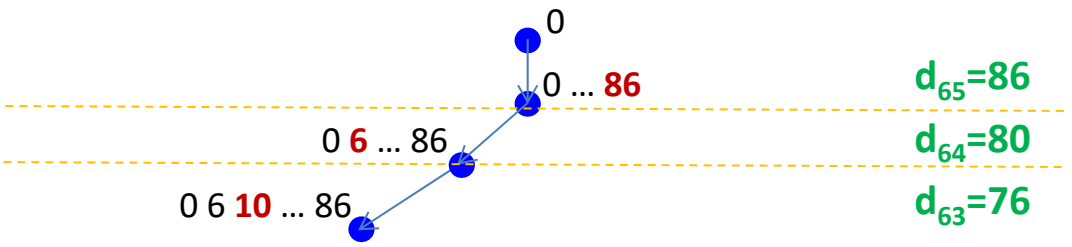
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

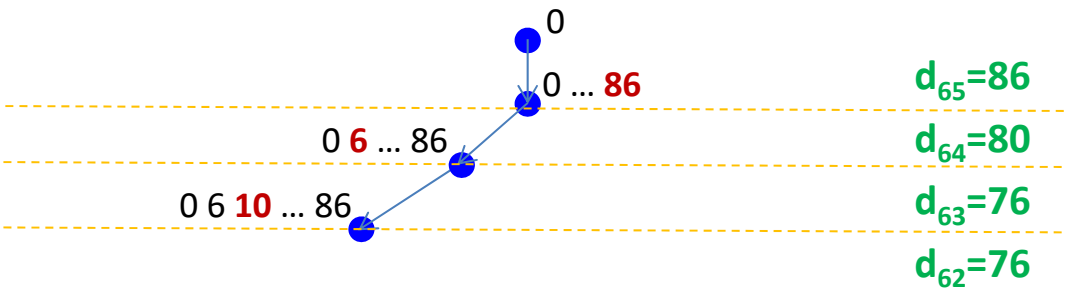
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

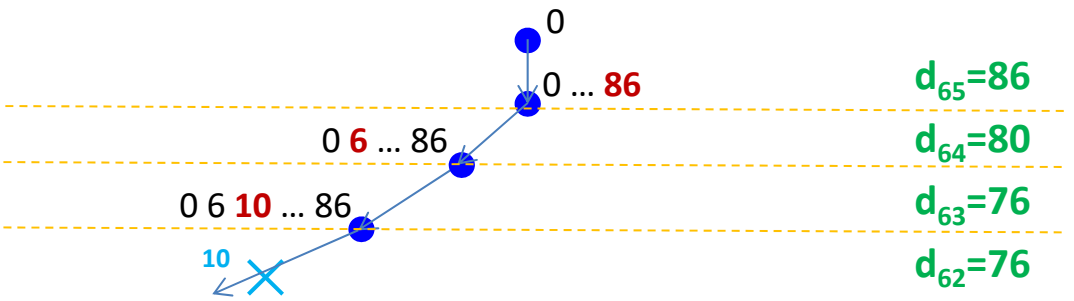
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

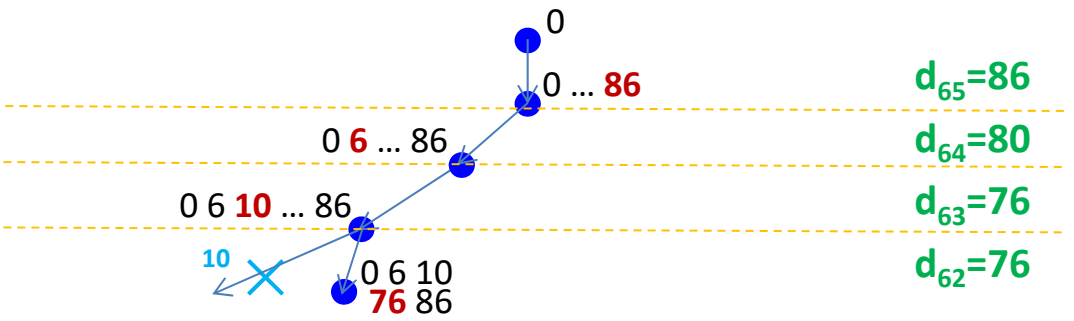
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

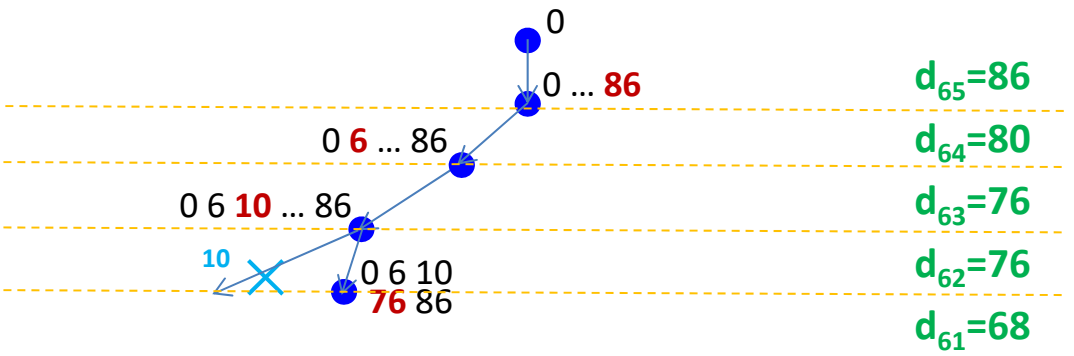
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

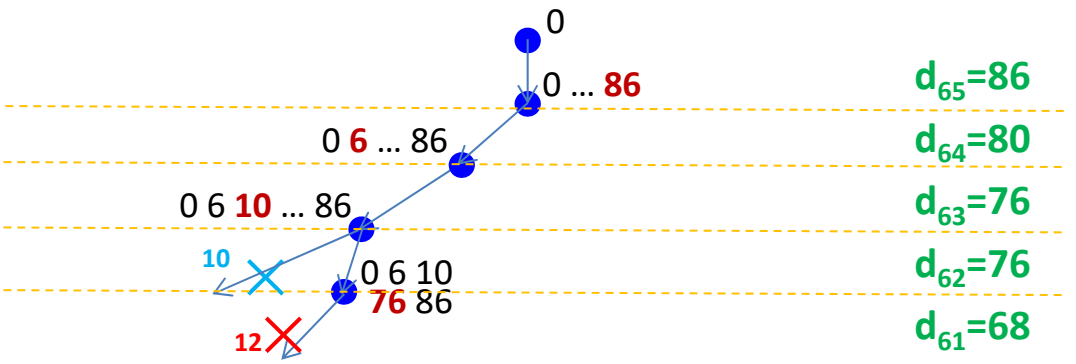
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

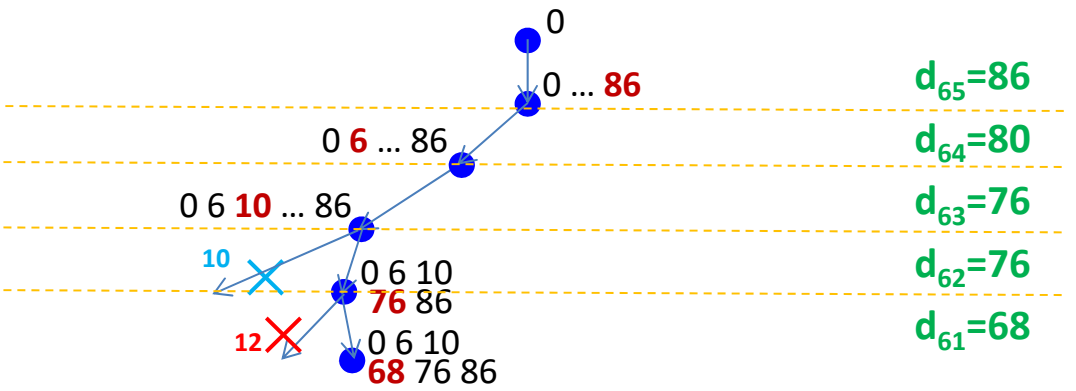
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

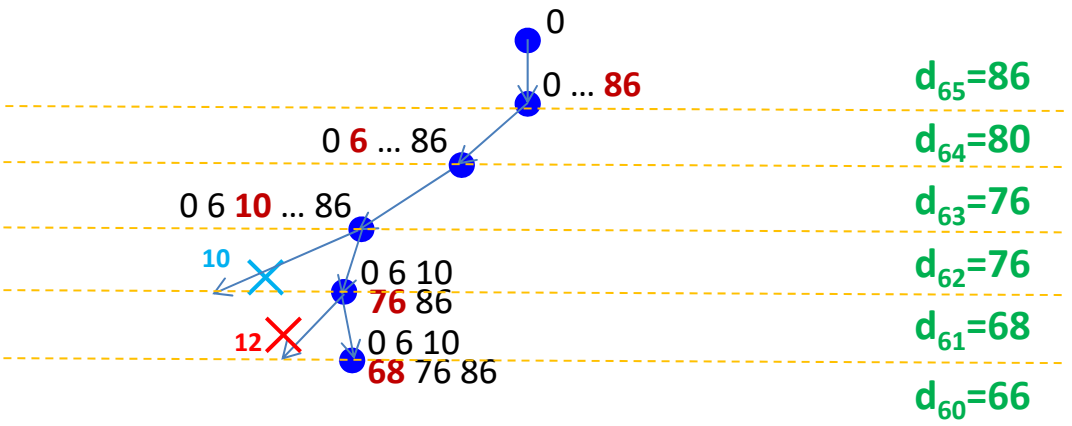
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

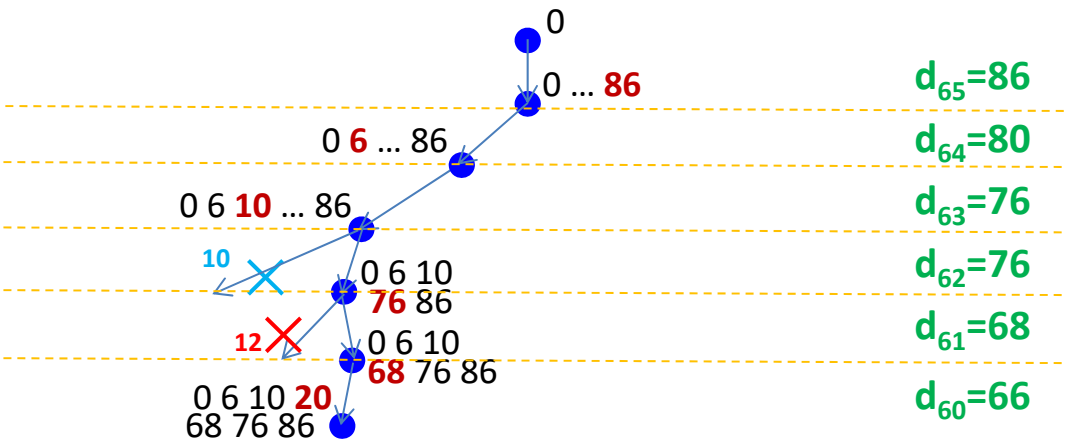
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

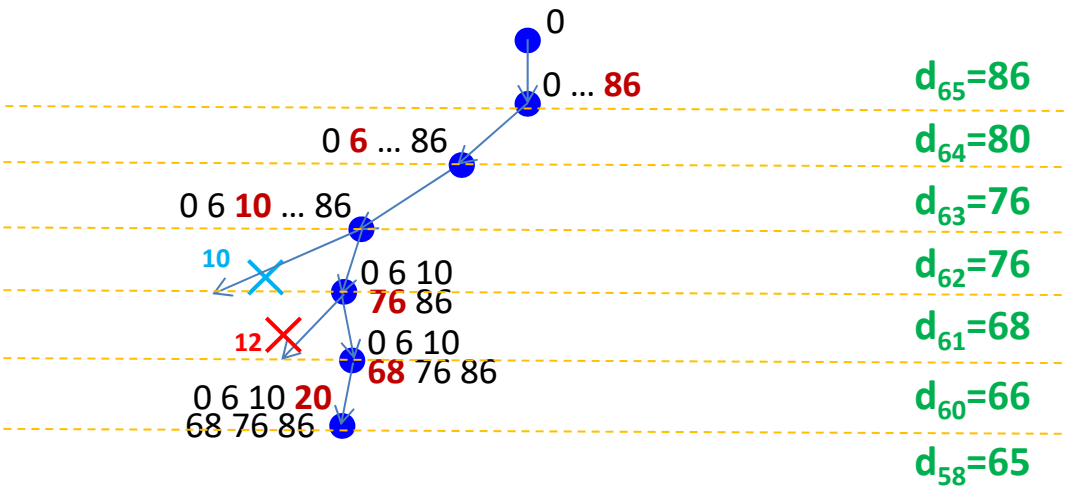
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

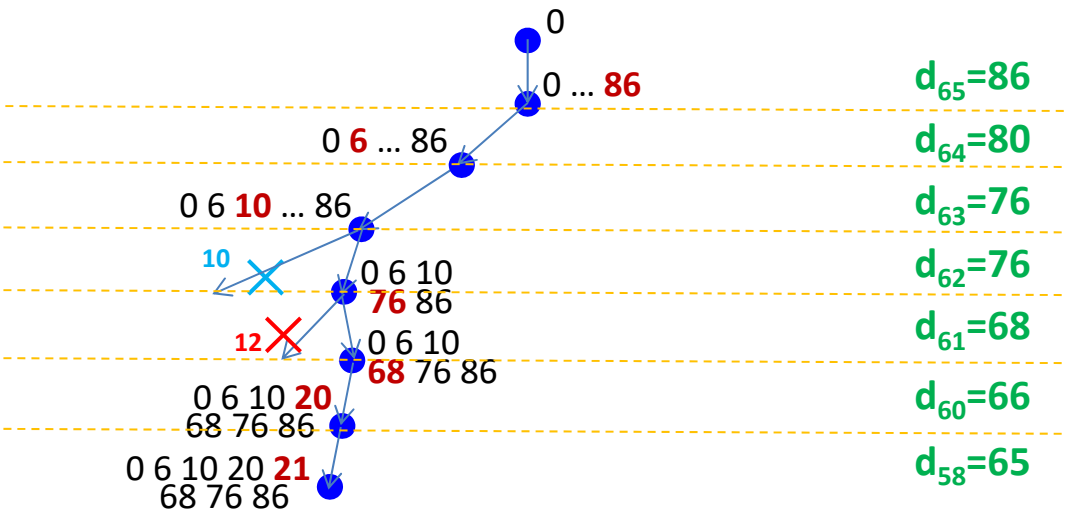
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

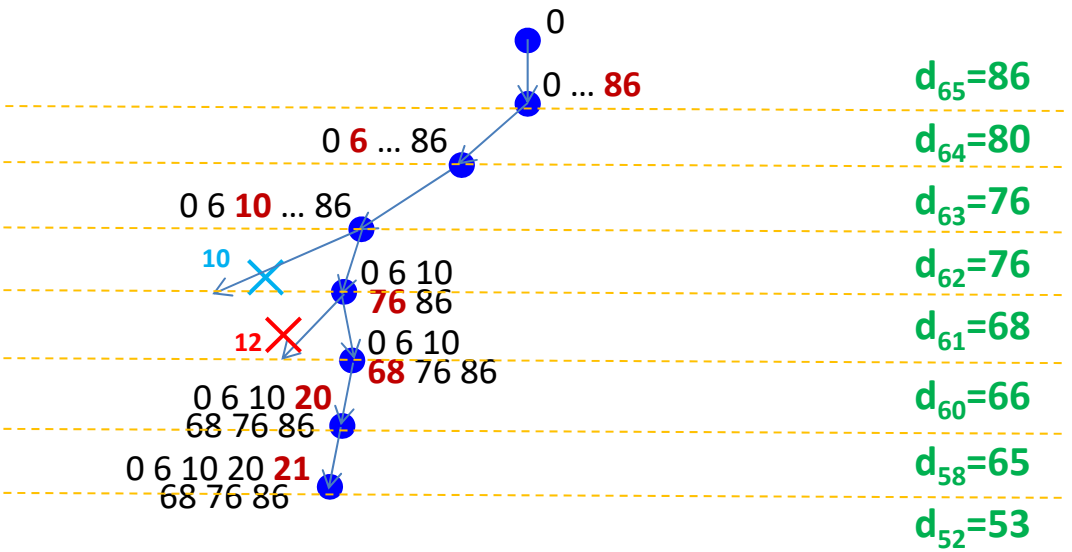
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

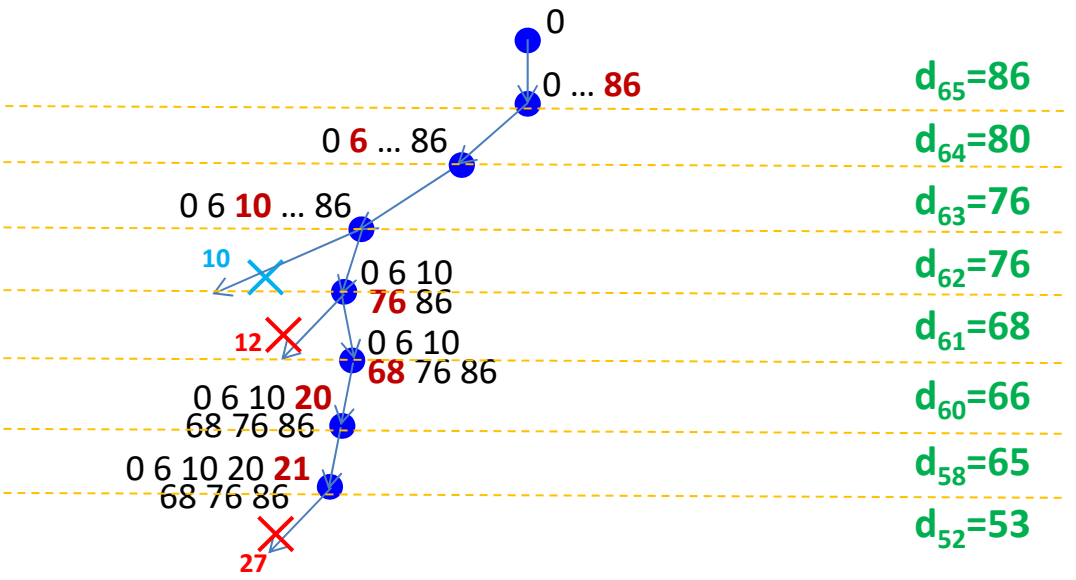
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

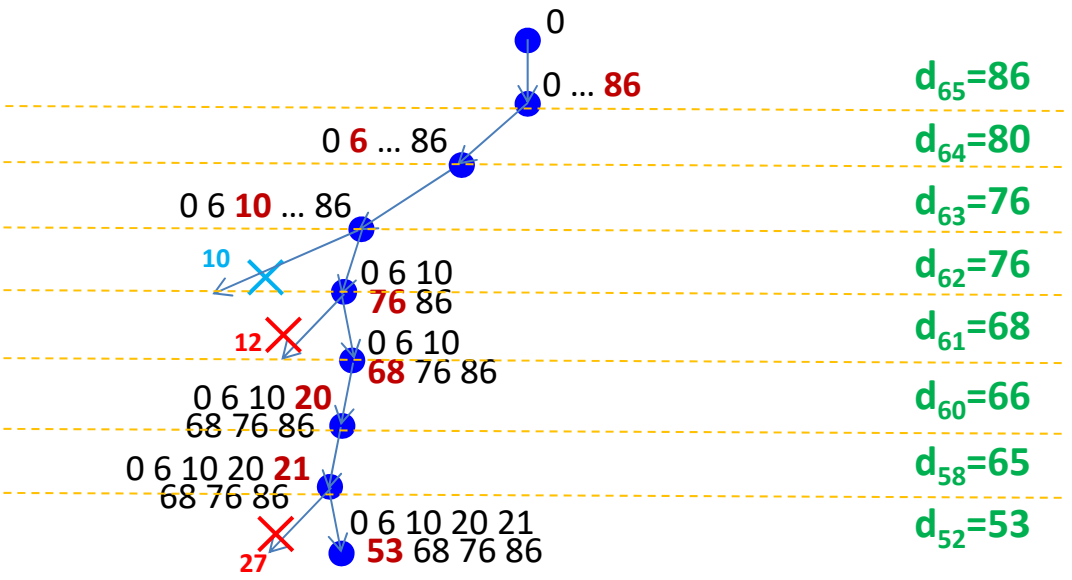
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

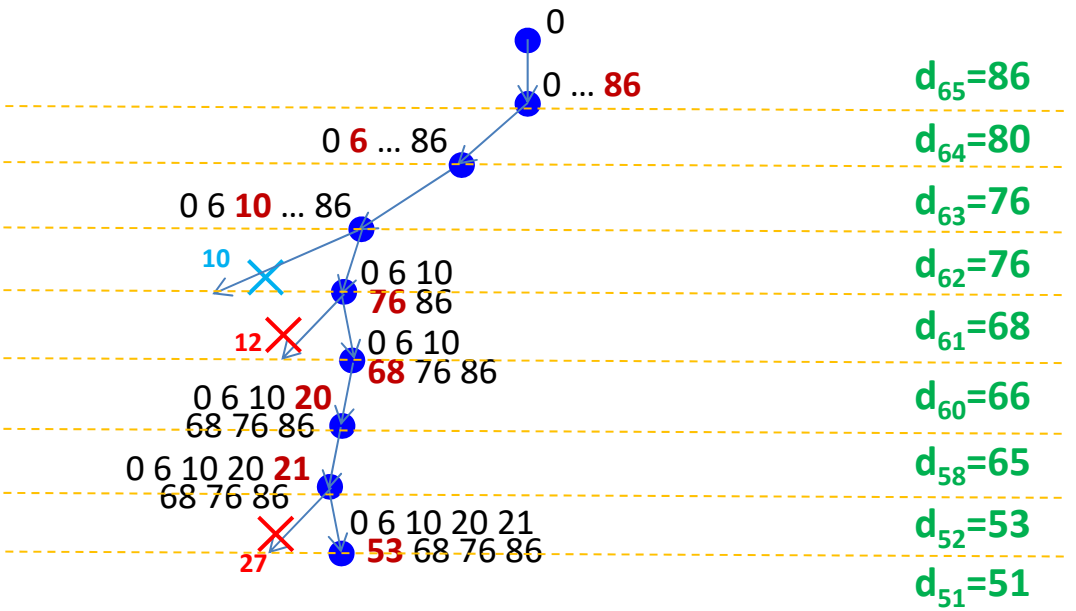
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

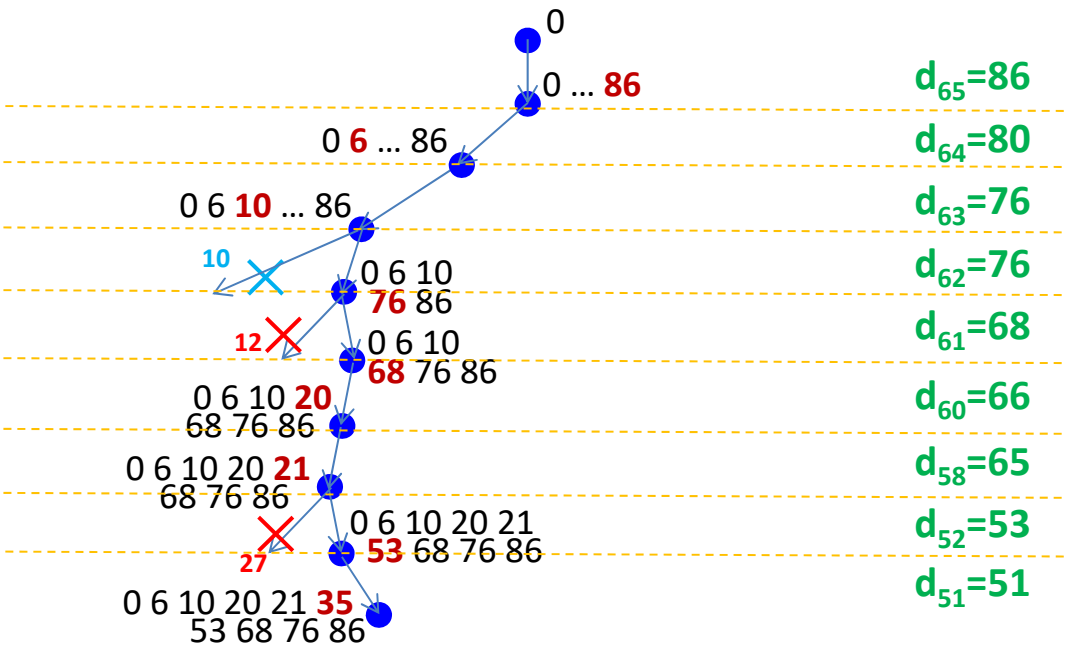
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

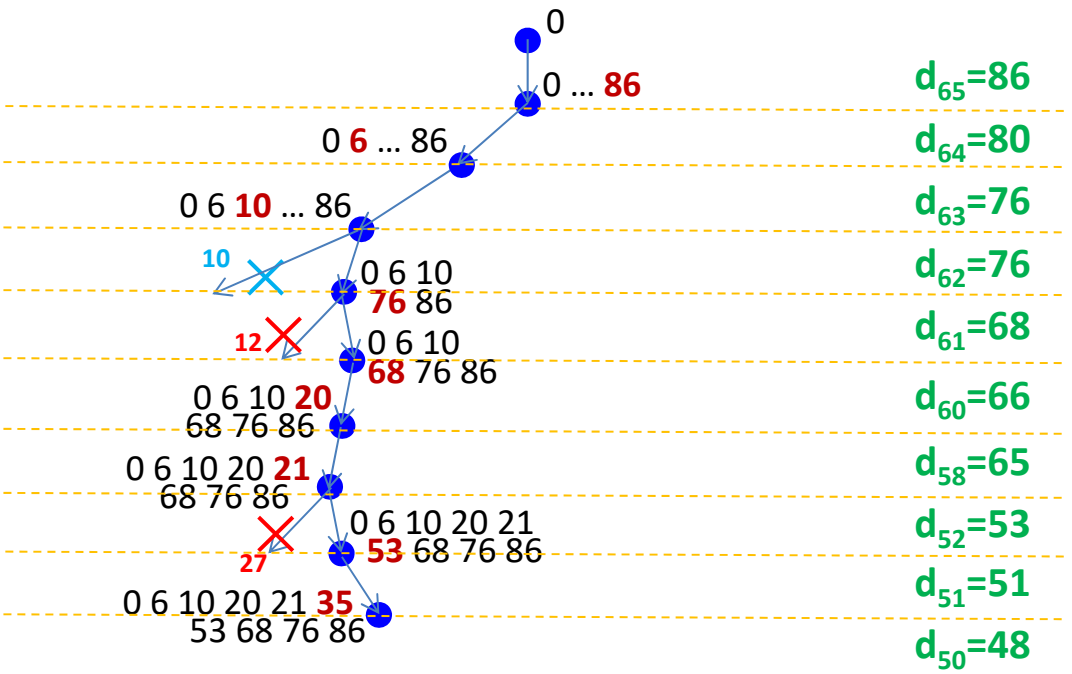
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

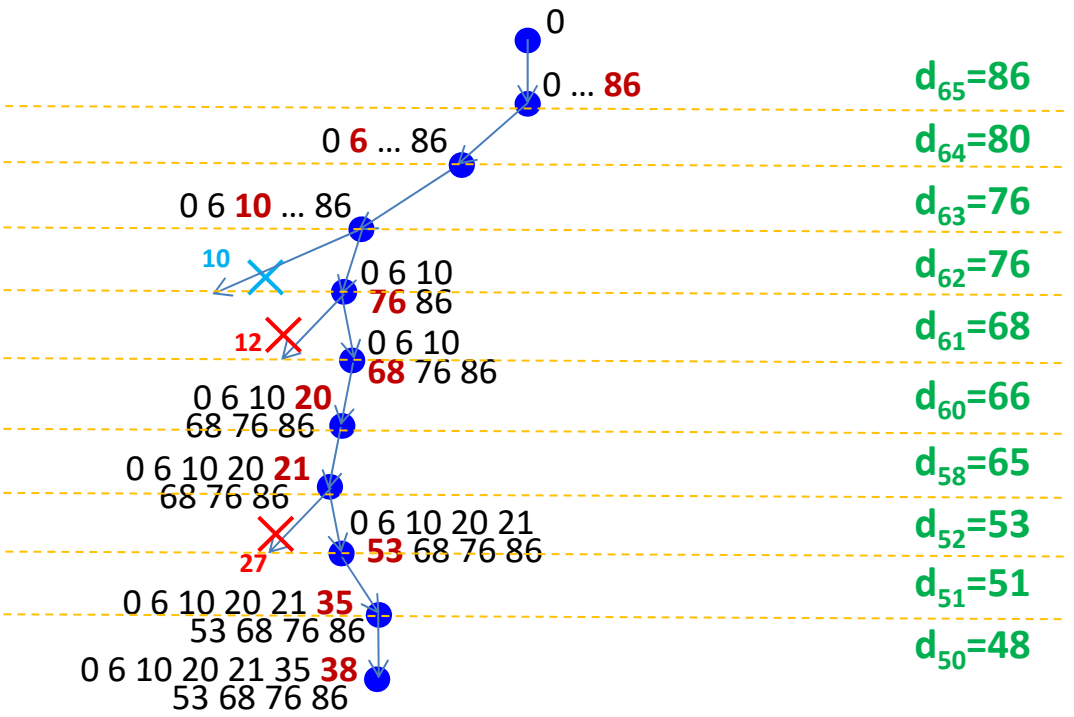
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

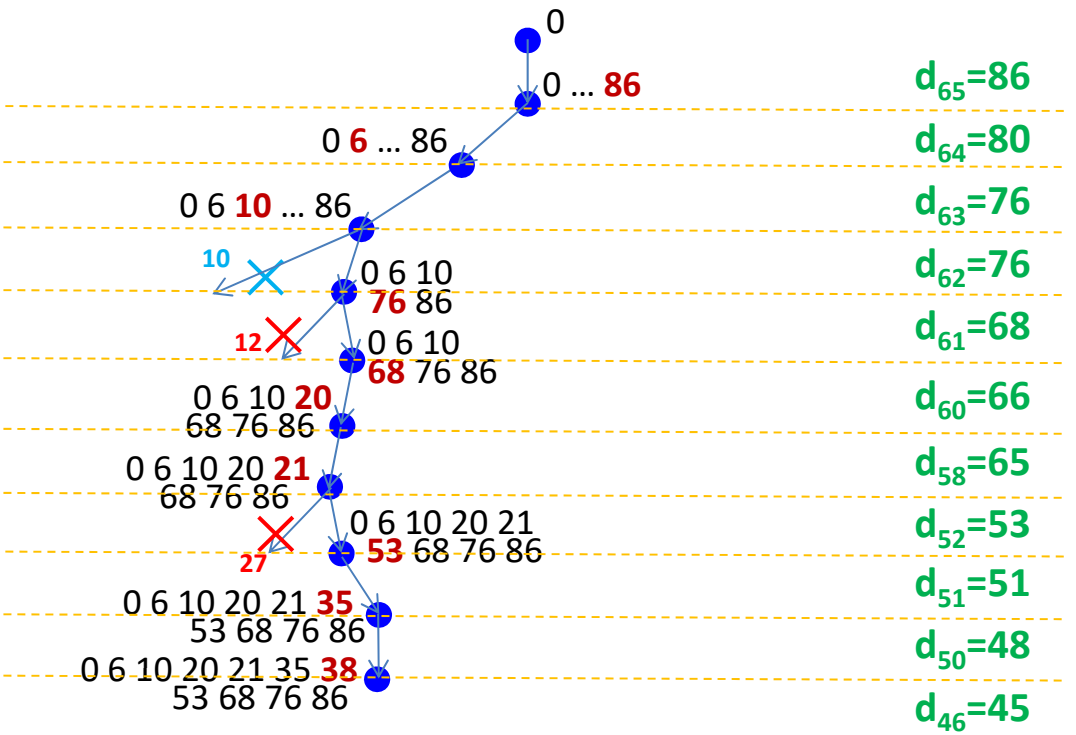
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

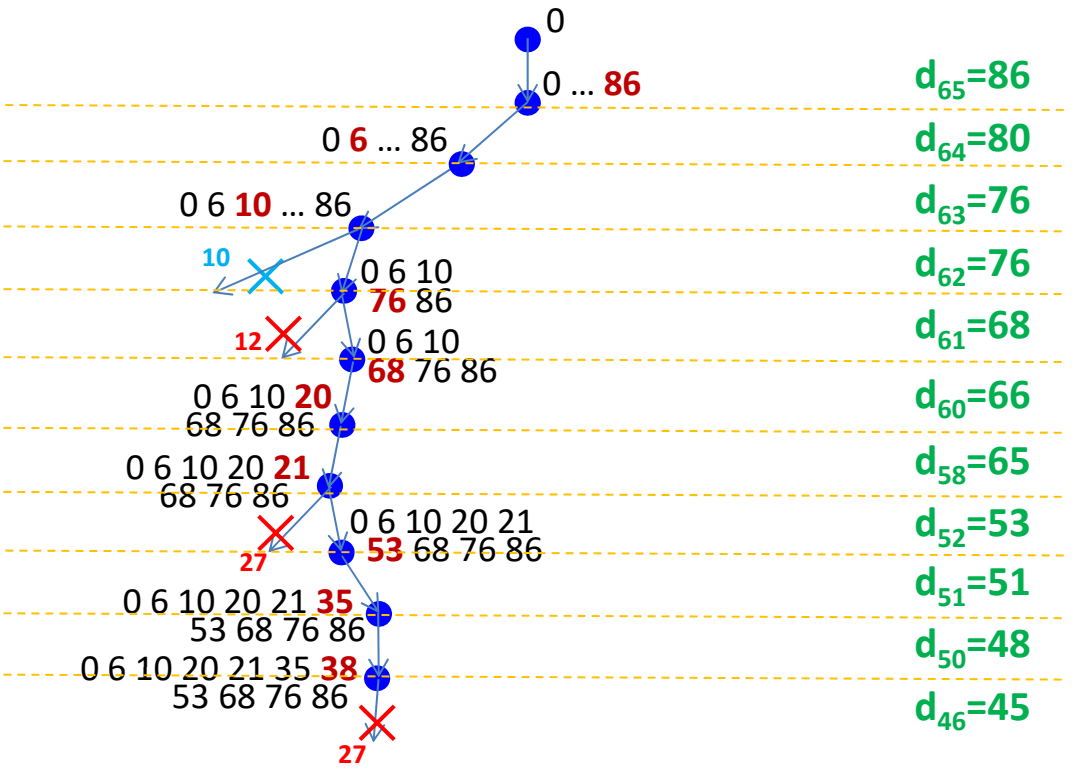
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

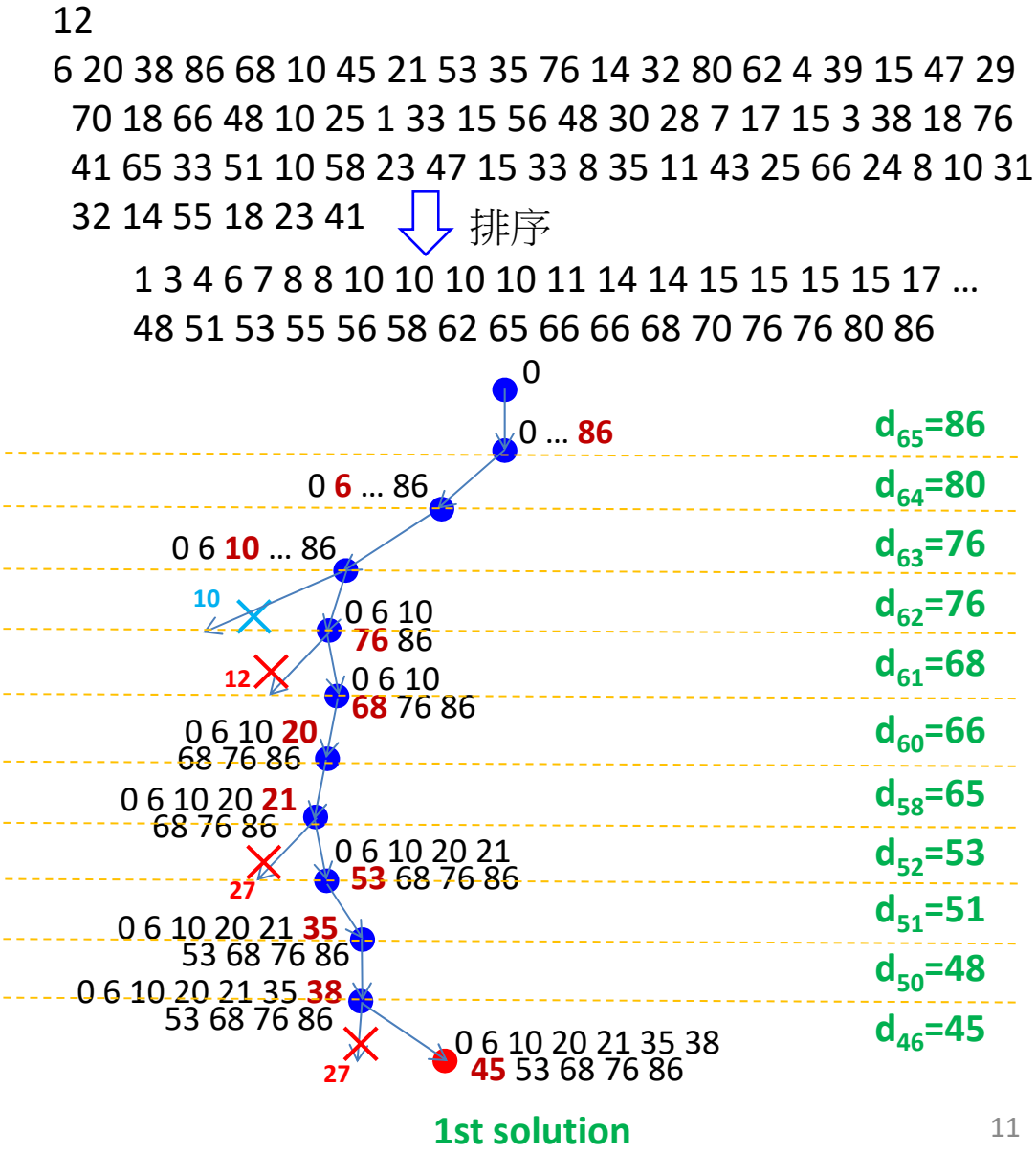
- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

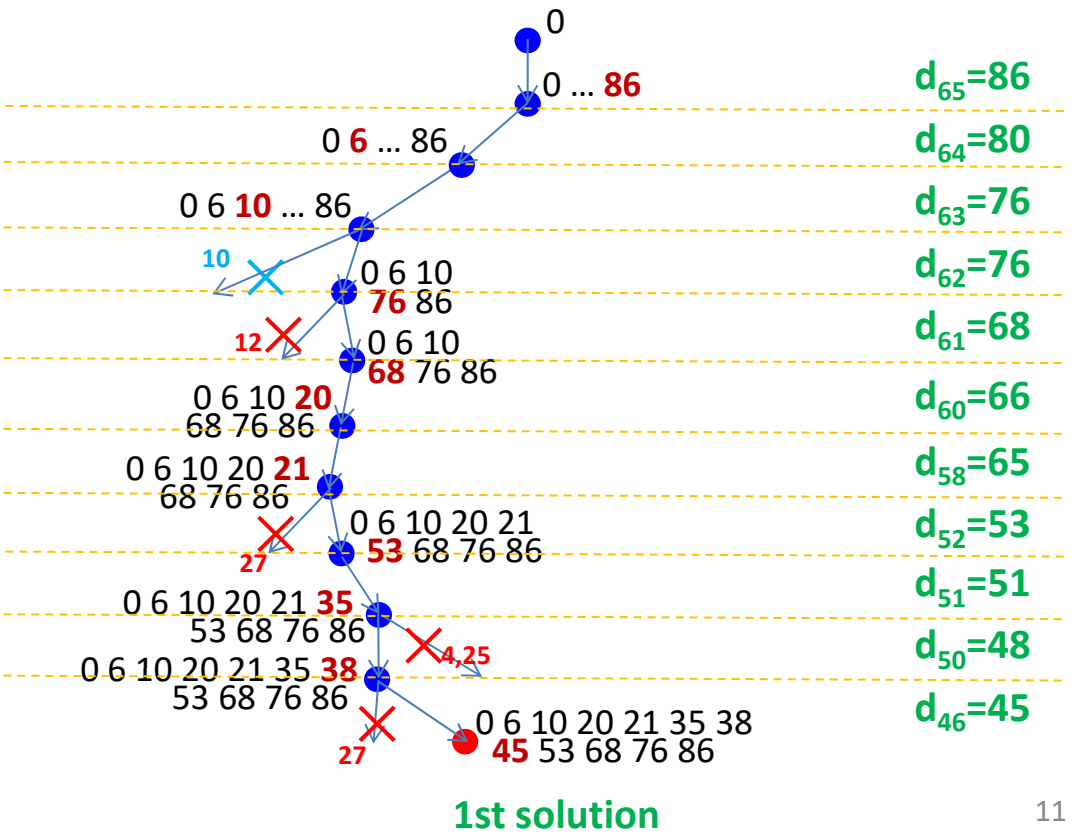
- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

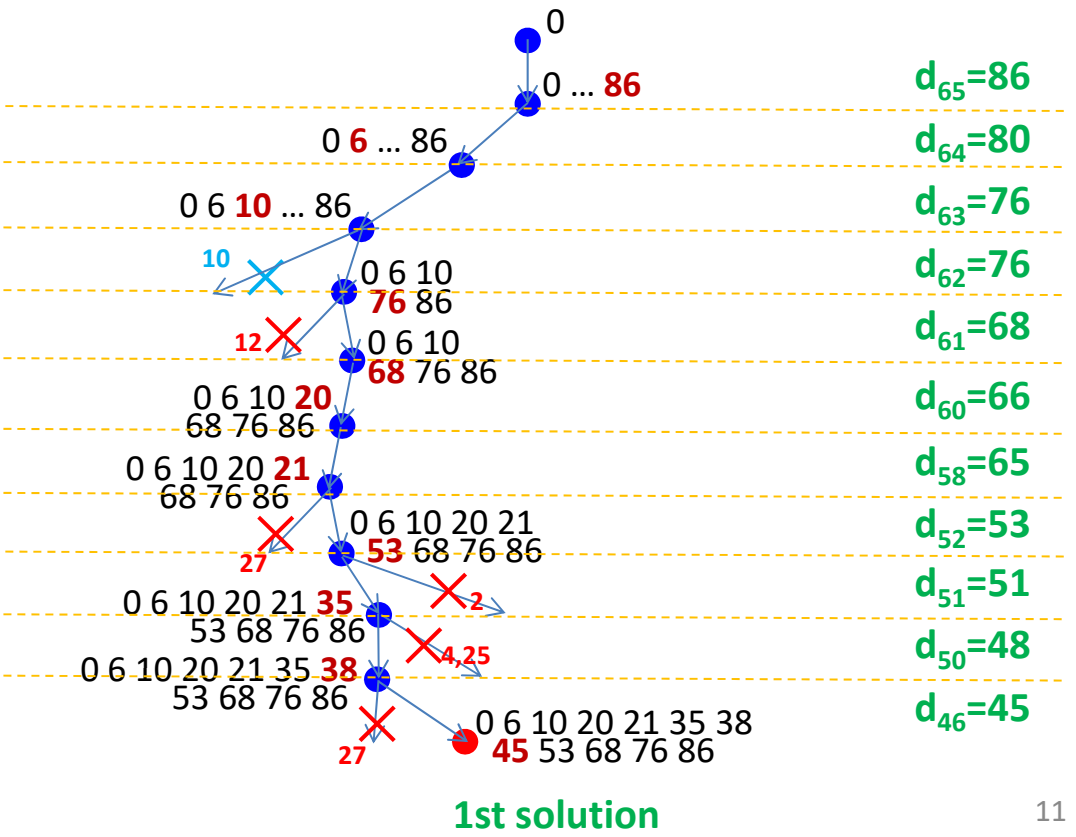
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

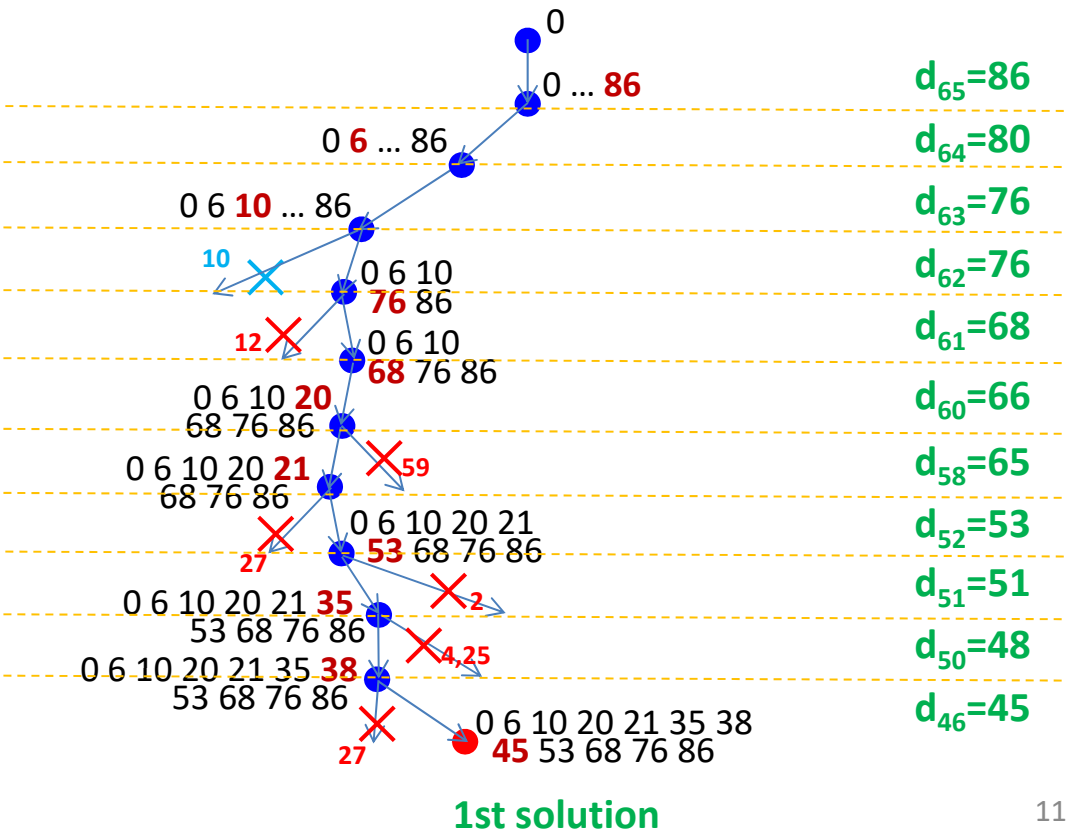
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

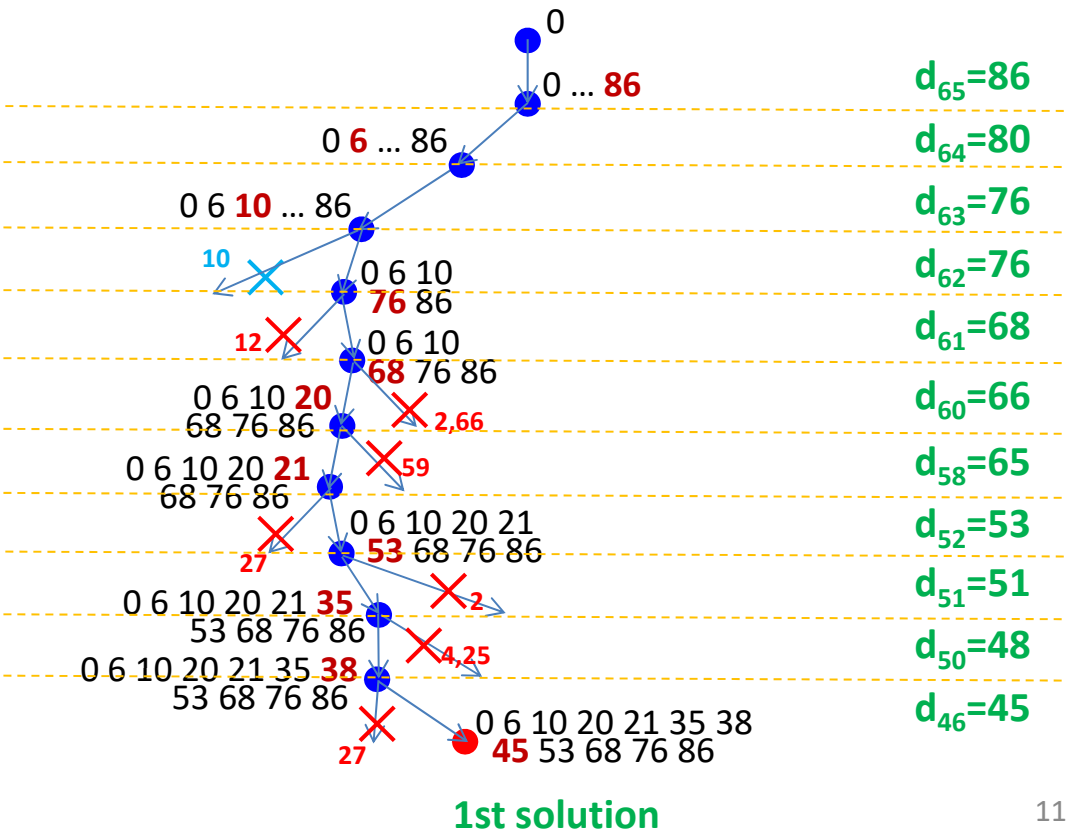
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

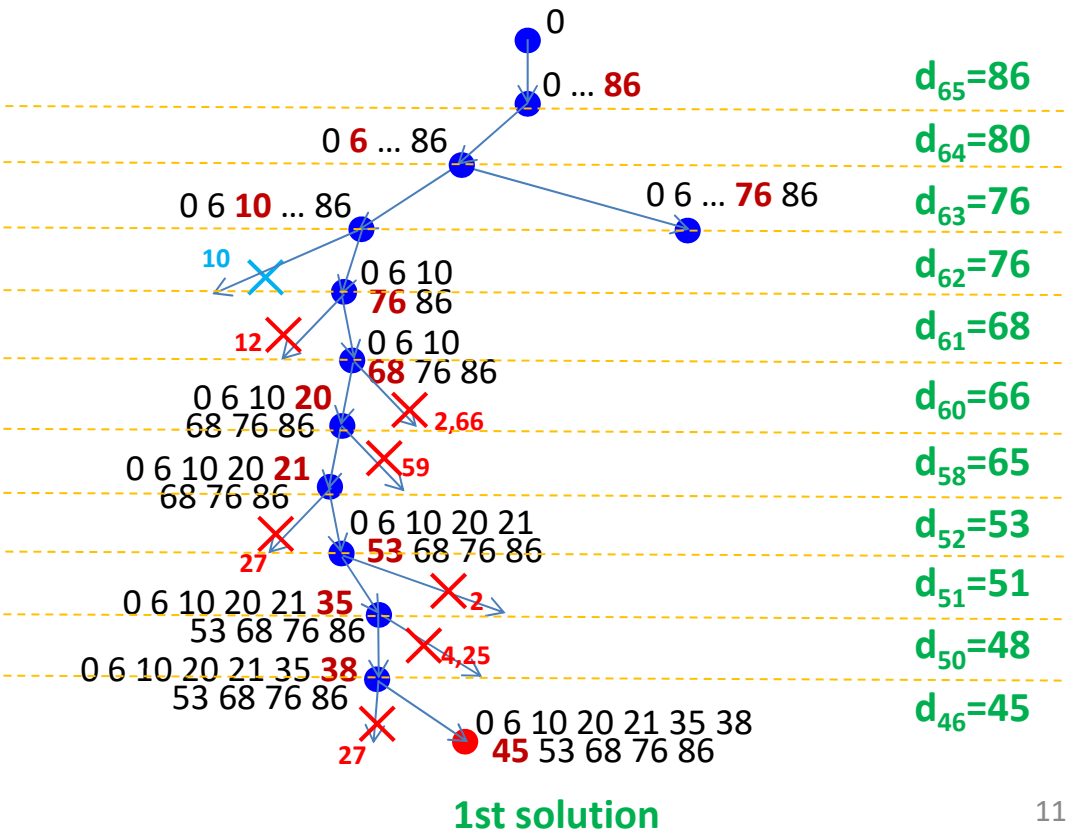
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

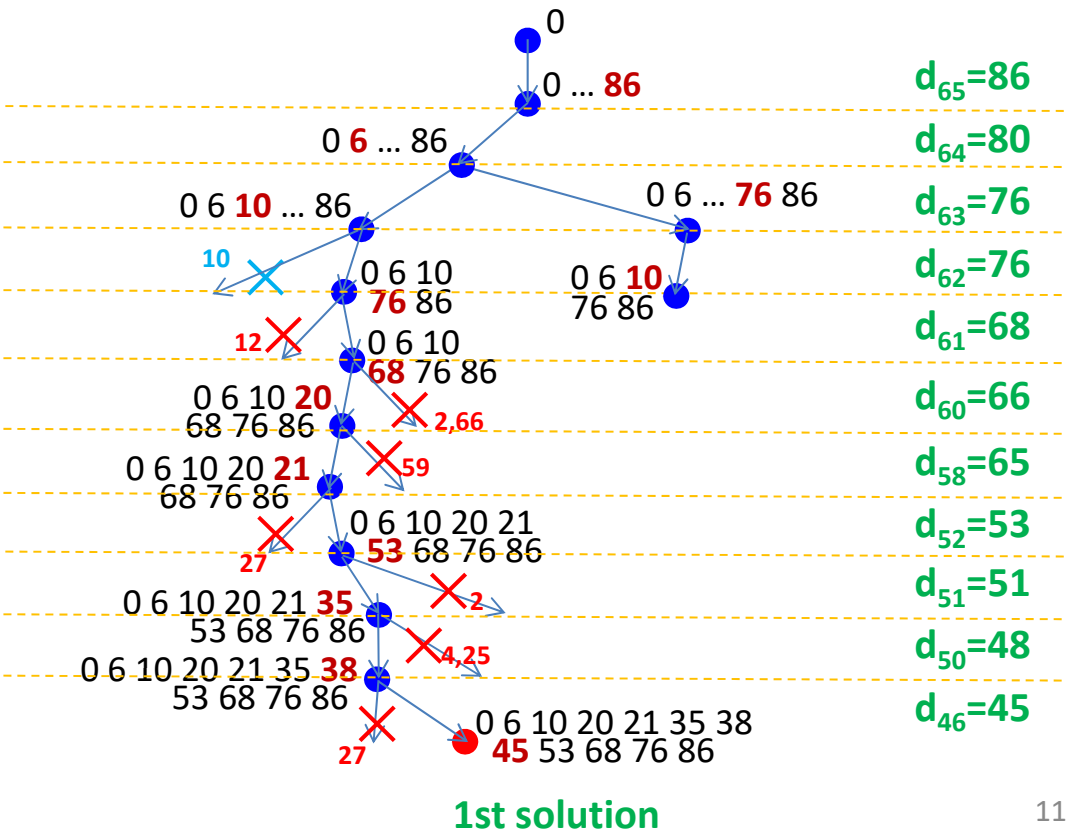
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

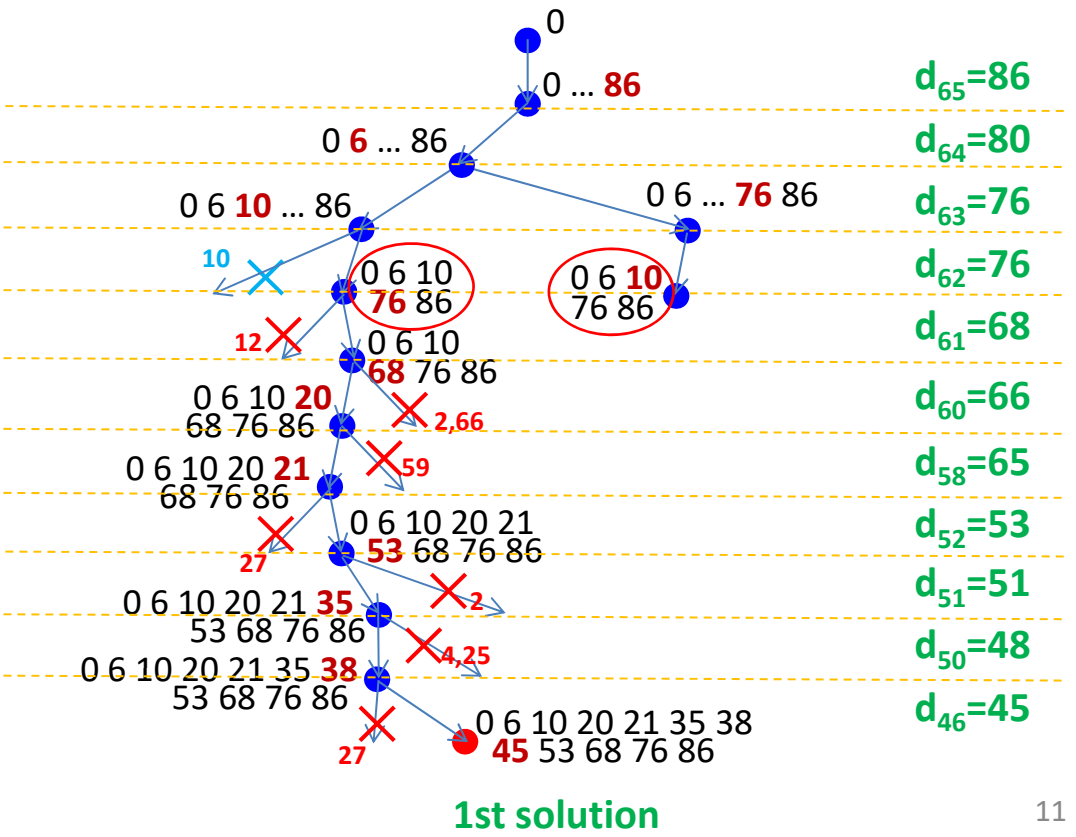
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

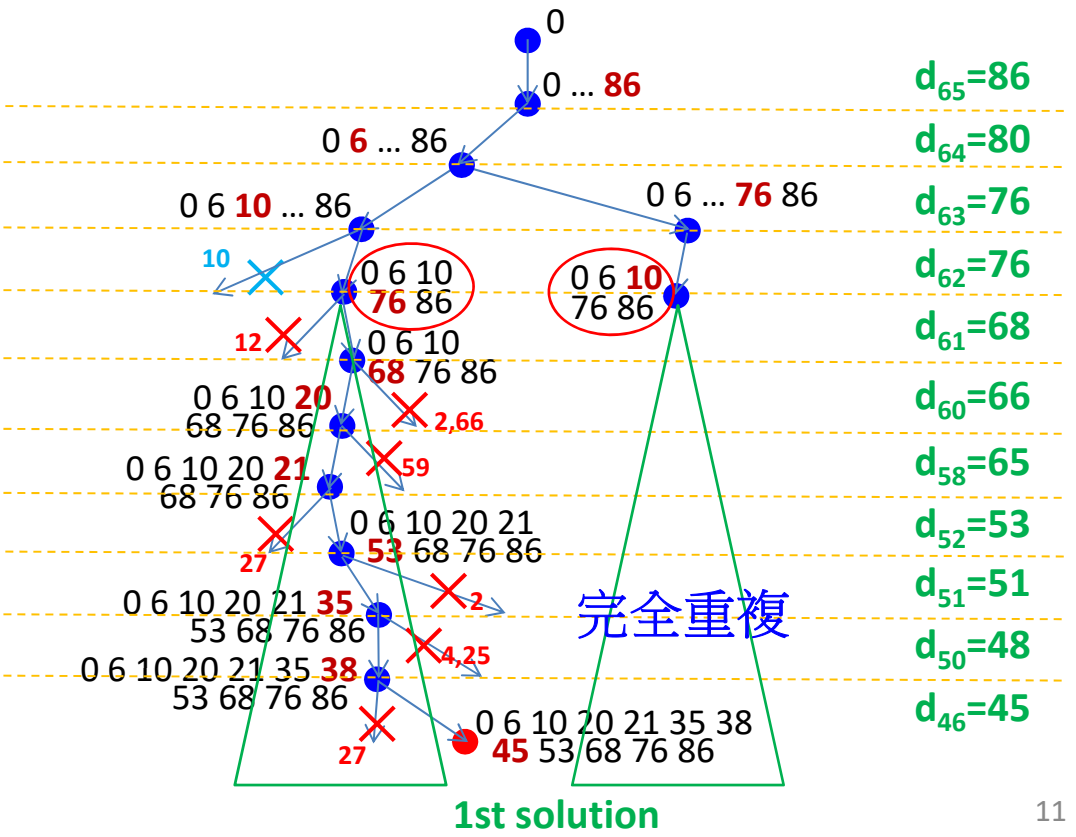
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

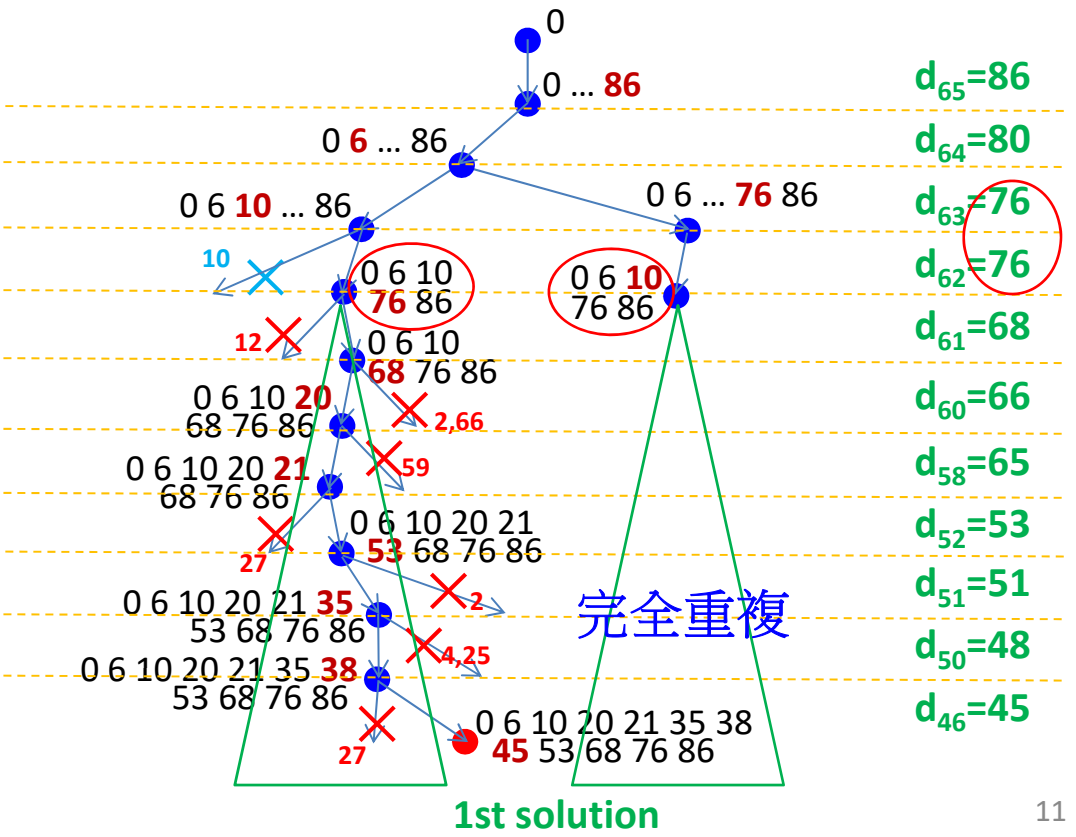
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

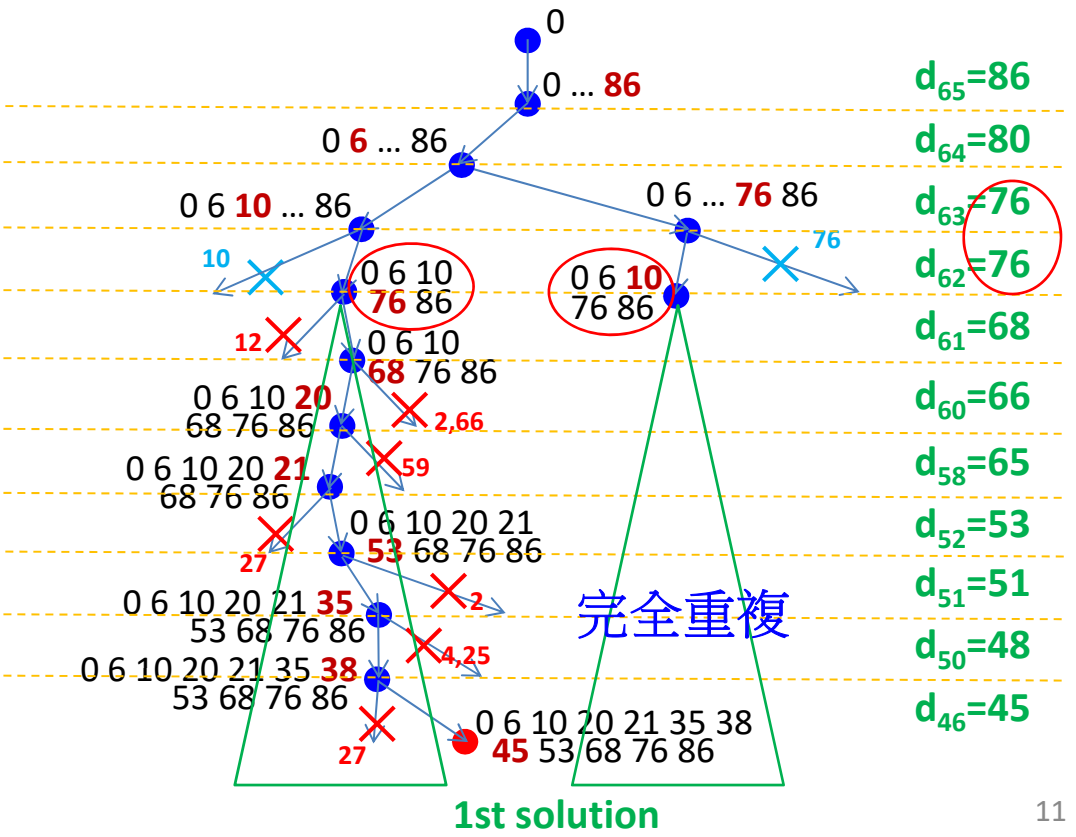
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

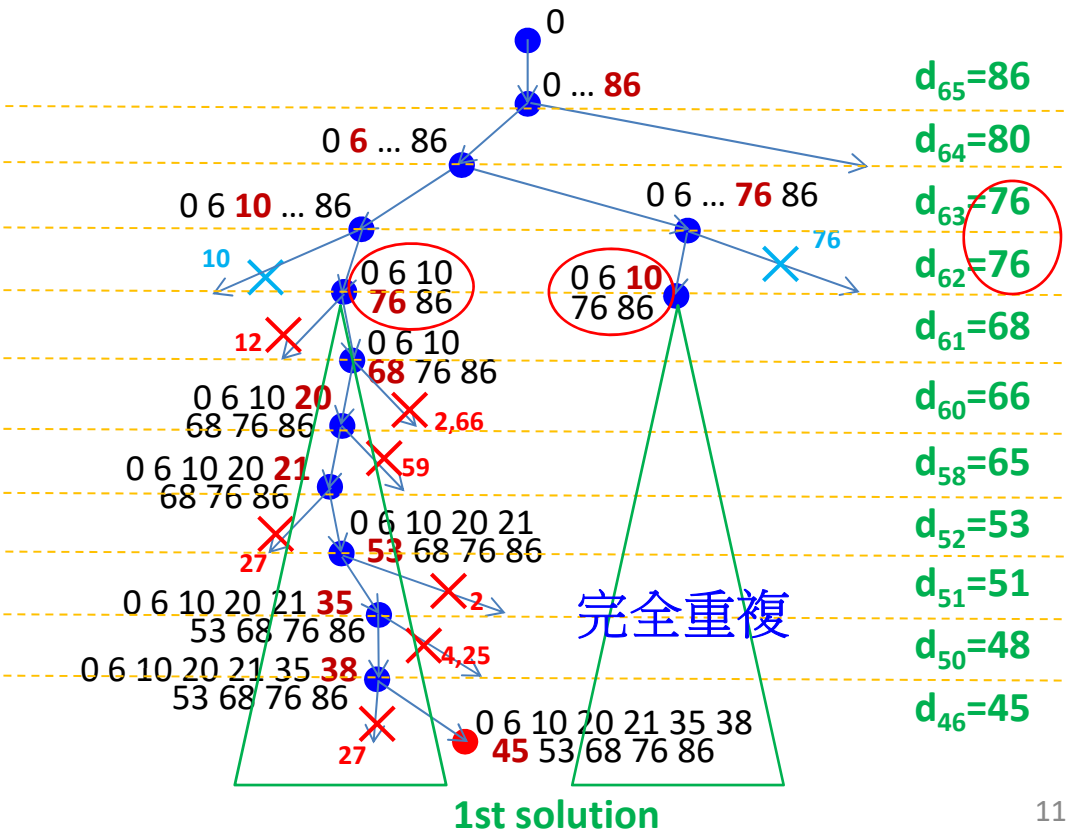
12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



擴充功能 - 找出所有答案

- 前頁程式可以找出字典序最小/最大的數列，和一般 DFS 程式一樣，找到答案的時候 return 1 就以第一個找到的答案結束搜尋，return 0 則代表沒有找到答案，讓遞迴函式退回前一層繼續列舉其它可能性，如此可以找出所有的答案，例如前面的 countUp() 函式
- 修改程式測試以後發現許多組答案會重複出現 ⇒ 前面的 DFS 遞迴函式定義出來的決策樹有重複的分支，會找到一模一樣的答案。

12
6 20 38 86 68 10 45 21 53 35 76 14 32 80 62 4 39 15 47 29
70 18 66 48 10 25 1 33 15 56 48 30 28 7 17 15 3 38 18 76
41 65 33 51 10 58 23 47 15 33 8 35 11 43 25 66 24 8 10 31
32 14 55 18 23 41 ⇩ 排序
1 3 4 6 7 8 8 10 10 10 10 11 14 14 15 15 15 15 17 …
48 51 53 55 56 58 62 65 66 66 68 70 76 76 80 86



Δ 裡面最大元素重複出現

Δ 裡面最大元素重複出現

- 如果 Δ 中最大的兩個數字是 d, d，還有至少兩個數字還沒有決定，所有決策樹的分支如下

Δ 裡面最大元素重複出現

- 如果 Δ 中最大的兩個數字是 d, d，還有至少兩個數字還沒有決定，所有決策樹的分支如下
  1. 左子樹、左子樹：數字 a_n-d 重複

Δ 裡面最大元素重複出現

- 如果 Δ 中最大的兩個數字是 d, d，還有至少兩個數字還沒有決定，所有決策樹的分支如下
  1. 左子樹、左子樹：數字 a_n-d 重複
  2. 左子樹、右子樹：數字 a_n-d, 數字 d

Δ 裡面最大元素重複出現

- 如果 Δ 中最大的兩個數字是 d, d，還有至少兩個數字還沒有決定，所有決策樹的分支如下
  1. 左子樹、左子樹：數字 a_n-d 重複
  2. 左子樹、右子樹：數字 a_n-d, 數字 d
  3. 右子樹、左子樹：數字 d, 數字 a_n-d

Δ 裡面最大元素重複出現

- 如果 Δ 中最大的兩個數字是 d, d，還有至少兩個數字還沒有決定，所有決策樹的分支如下
  1. 左子樹、左子樹：數字 a_n-d 重複
  2. 左子樹、右子樹：數字 a_n-d, 數字 d
  3. 右子樹、左子樹：數字 d, 數字 a_n-d
  4. 右子樹、右子樹：數字 d, 數字 d 重複

Δ 裡面最大元素重複出現

- 如果 Δ 中最大的兩個數字是 d, d，還有至少兩個數字還沒有決定，所有決策樹的分支如下
  1. 左子樹、左子樹：數字 a_n-d 重複
  2. 左子樹、右子樹：數字 a_n-d, 數字 d
  3. 右子樹、左子樹：數字 d, 數字 a_n-d
  4. 右子樹、右子樹：數字 d, 數字 d 重複
- 2 和 3 是等效的，同時數字 a_n-d 和 d 和前面決定的數字之差的絕對值一定在 Δ 裡面，不需要檢查，如果任何一個不在最後的數列中， Δ 裡面就不會有兩個 d，2 和 3 這兩個分支需要合併，才不會像前面的 DFS 找到完全一樣的答案

Δ 裡面最大元素重複出現

- 如果 Δ 中最大的兩個數字是 d, d，還有至少兩個數字還沒有決定，所有決策樹的分支如下
  1. 左子樹、左子樹：數字 a_n-d 重複
  2. 左子樹、右子樹：數字 a_n-d, 數字 d
  3. 右子樹、左子樹：數字 d, 數字 a_n-d
  4. 右子樹、右子樹：數字 d, 數字 d 重複
- 2 和 3 是等效的，同時數字 a_n-d 和 d 和前面決定的數字之差的絕對值一定在 Δ 裡面，不需要檢查，如果任何一個不在最後的數列中， Δ 裡面就不會有兩個 d，2 和 3 這兩個分支需要合併，才不會像前面的 DFS 找到完全一樣的答案
- 如果 Δ 中最大的兩個數字是 d, d，只剩下一個數字還沒有決定，數字 d = a_n-d 是最後的一個數字

超過 2 組解的測資

超過 2 組解的測資

Sample Input:

超過 2 組解的測資

Sample Input:

6

超過 2 組解的測資

Sample Input:

6

1 1 2 2 3 4 5 5 6 6 7 8 9 10 11

超過 2 組解的測資

Sample Input:

6
1 1 2 2 3 4 5 5 6 6 7 8 9 10 11

Sample Output:

1:0 1 2 6 8 11
2:0 1 6 7 9 11
3:0 2 4 5 10 11
4:0 3 5 9 10 11

超過 2 組解的測資

Sample Input:

6

1 1 2 2 3 4 5 5 6 6 7 8 9 10 11

Sample Output:

1:0 1 2 6 8 11

2:0 1 6 7 9 11

3:0 2 4 5 10 11

4:0 3 5 9 10 11

Sample Input:

8

1 1 1 1 1 2 2 3 4 4 4 5 5 5 5 6 6 6 6 7 7 8 9 10 10 11 11 12

超過 2 組解的測資

Sample Input:

6

1 1 2 2 3 4 5 5 6 6 7 8 9 10 11

Sample Output:

1:0 1 2 6 8 11
2:0 1 6 7 9 11
3:0 2 4 5 10 11
4:0 3 5 9 10 11

Sample Input:

8

1 1 1 1 1 2 2 3 4 4 4 5 5 5 5 6 6 6 6 7 7 8 9 10 10 11 11 12

Sample Output:

1:0 1 2 6 7 8 11 12
2:0 1 2 6 7 8 11 12
3:0 1 4 5 6 10 11 12
4:0 1 4 5 6 10 11 12