ZeroJudge d443, UVa 10497 Sweet Child Makes Trouble

丁培毅

113/01/23

題目說明

- 小婷拿東西來玩總是不把東西放回原位，甚至玩過 n 個東西以後，沒有一個東西放回原來位置。

題目說明

- 小婷拿東西來玩總是不把東西放回原位，甚至玩過 n 個東西以後，沒有一個東西放回原來位置。
- 請設計一個程式

題目說明

- 小婷拿東西來玩總是不把東西放回原位，甚至玩過 n 個東西以後，沒有一個東西放回原來位置。
- 請設計一個程式
  - 輸入資料是一個不大於 800 的正整數 n (-1 代表資料結束)

題目說明

- 小婷拿東西來玩總是不把東西放回原位，甚至玩過 n 個東西以後，沒有一個東西放回原來位置。
- 請設計一個程式
  - 輸入資料是一個不大於 800 的正整數 n (-1 代表資料結束)
  - 輸出總共有多少種放置的方法 (所有東西都不在原來位置上)

題目說明

- 小婷拿東西來玩總是不把東西放回原位，甚至玩過 n 個東西以後，沒有一個東西放回原來位置。
- 請設計一個程式
  - 輸入資料是一個不大於 800 的正整數 n (-1 代表資料結束)
  - 輸出總共有多少種放置的方法 (所有東西都不在原來位置上)
- 範例輸入

  1
  2
  3
  4
  -1

題目說明

- 小婷拿東西來玩總是不把東西放回原位，甚至玩過 n 個東西以後，沒有一個東西放回原來位置。
- 請設計一個程式
  - 輸入資料是一個不大於 800 的正整數 n (-1 代表資料結束)
  - 輸出總共有多少種放置的方法 (所有東西都不在原來位置上)

- 範例輸入

1
2
3
4
-1

- 範例輸出

0
1
2
9

方法一: 暴力列舉所有 n! 種排列

- 由於 $2 \le n \le 800$, 暴力列舉在 n=13,14 左右就已經需要 1 秒了

方法一: 暴力列舉所有 n! 種排列

- 由於 $2 \leq n \leq 800$, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one’s foot in the door 的好方法，有機會可以分析出解答的特性

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one’s foot in the door 的好方法，有機會可以分析出解答的特性

n=3

1 2 3
1 3 2
2 1 **3**
2 3 1
3 1 2
3 **2** 1

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3	n=4
1 2 3	**1 2 3 4**
1 3 2	1 2 4 3
2 1 **3**	**1** 3 2 **4**
2 3 1	**1** 3 4 2
3 1 2	**1** 4 2 3
3 **2** 1	**1** 4 **3** 2

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3	n=4	
1 2 3	**1 2 3 4**	2 1 **3 4**
1 3 2	1 2 4 3	2 1 4 3
2 1 **3**	**1** 3 2 **4**	2 3 1 **4**
2 3 1	**1** 3 4 2	2 3 4 1
3 1 2	**1** 4 2 3	2 4 1 3
3 **2** 1	**1** 4 **3** 2	2 4 **3** 1

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3	n=4		
1 2 3	**1 2 3 4**	2 1 **3 4**	3 1 2 **4**
1 3 2	1 2 4 3	2 1 4 3	3 1 4 2
2 1 **3**	**1** 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**
2 3 1	**1** 3 4 2	2 3 4 1	3 **2** 4 1
3 1 2	**1** 4 2 3	2 4 1 3	3 4 1 2
3 **2** 1	**1** 4 **3** 2	2 4 **3** 1	3 4 2 1

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3

1 2 3
1 3 2
2 1 **3**
2 3 1
3 1 2
3 **2** 1

n=4

1 2 3 4	2 1 **3 4**	3 1 2 **4**	4 1 2 3
1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
1 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
1 3 4 2	2 3 4 1	3 **2** 4 1	4 **2 3** 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3	n=4			
1 2 3	**1 2 3 4**	2 1 **3 4**	3 1 2 **4**	4 1 2 3
1 3 2	1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
2 1 **3**	**1** 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
2 3 1	**1** 3 4 2	2 3 4 1	3 **2** 4 1	4 **2 3** 1
3 1 2	**1** 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
3 **2** 1	**1** 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

n
1
2
3
4
5

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3	n=4			
1 2 3	**1 2 3 4**	2 1 **3 4**	3 1 2 **4**	4 1 2 3
1 3 2	1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
2 1 **3**	**1** 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
2 3 1	**1** 3 4 2	2 3 4 1	3 **2** 4 1	4 **2 3** 1
3 1 2	**1** 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
3 **2** 1	**1** 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

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

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3	n=4			
1 2 3	**1 2 3 4**	2 1 **3 4**	3 1 2 **4**	4 1 2 3
1 3 2	1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
2 1 **3**	**1** 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
2 3 1	**1** 3 4 2	2 3 4 1	3 **2** 4 1	4 **2 3** 1
3 1 2	**1** 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
3 **2** 1	**1** 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

n	f(n) 都不在正確位置	n	
1	0	6	
2		7	
3		8	
4		9	
5		10	

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3	n=4			
1 2 3	**1 2 3 4**	2 1 **3 4**	3 1 2 **4**	4 1 2 3
1 3 2	1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
2 1 **3**	**1** 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
2 3 1	**1** 3 4 2	2 3 4 1	3 **2** 4 1	4 **2 3** 1
3 1 2	**1** 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
3 **2** 1	**1** 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

n	f(n) 都不在正確位置	n! 所有排列	n
1	0	1	6
2			7
3			8
4			9
5			10

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3	n=4			
1 2 3	**1 2 3 4**	2 1 **3 4**	3 1 2 **4**	4 1 2 3
1 3 2	1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
2 1 **3**	**1** 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
2 3 1	**1** 3 4 2	2 3 4 1	3 **2** 4 1	4 **2 3** 1
3 1 2	**1** 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
3 **2** 1	**1** 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

n	都不在正確位置 f(n)	所有排列 n!	有部份在正確位置 n!-f(n)	n
1	0	1	1	6
2				7
3				8
4				9
5				10

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3

1 **2** **3**
1 3 2
2 1 **3**
2 3 1
3 1 2
3 **2** 1

n=4

1 **2** **3** **4**	2 1 **3** **4**	3 1 2 **4**	4 1 2 3
1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
1 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
1 3 4 2	2 3 4 1	3 **2** 4 1	4 **2** **3** 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

	都不在正確位置	所有排列	有部份在正確位置	
n	f(n)	n!	n!-f(n)	n
1	0	1	1	6
2	1	2	1	7
3				8
4				9
5				10

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3

1	**2**	**3**
1	3	2
2	1	**3**
2	3	1
3	1	2
3	**2**	1

n=4

1 2 3 4	2 1 **3 4**	3 1 2 **4**	4 1 2 3
1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
1 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
1 3 4 2	2 3 4 1	3 **2** 4 1	4 **2 3** 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

n	都不在正確位置 f(n)	所有排列 n!	有部份在正確位置 n!-f(n)	n
1	0	1	1	6
2	1	2	1	7
3	2	6	4	8
4				9
5				10

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3	n=4			
1 2 3	1 2 3 4	2 1 3 4	3 1 2 4	4 1 2 3
1 3 2	1 2 4 3	2 1 4 3	3 1 4 2	4 1 3 2
2 1 3	1 3 2 4	2 3 1 4	3 2 1 4	4 2 1 3
2 3 1	1 3 4 2	2 3 4 1	3 2 4 1	4 2 3 1
3 1 2	1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
3 2 1	1 4 3 2	2 4 3 1	3 4 2 1	4 3 2 1

n	都不在正確位置 f(n)	所有排列 n!	有部份在正確位置 n!-f(n)	n
1	0	1	1	6
2	1	2	1	7
3	2	6	4	8
4	9	24	15	9
5				10

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3

1 **2** **3**
1 3 2
2 1 **3**
2 3 1
3 1 2
3 **2** 1

n=4

1 **2** **3** **4**	2 1 **3** **4**	3 1 2 **4**	4 1 2 3
1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
1 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
1 3 4 2	2 3 4 1	3 **2** 4 1	4 **2** **3** 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

n	都不在正確位置 f(n)	所有排列 n!	有部份在正確位置 n!-f(n)	n
1	0	1	1	6
2	1	2	1	7
3	2	6	4	8
4	9	24	15	9
5	44	120	76	10

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3

1 **2** **3**
1 3 2
2 1 **3**
2 3 1
3 1 2
3 **2** 1

n=4

1 **2** **3** **4**	2 1 **3** **4**	3 1 2 **4**	4 1 2 3
1 **2** 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
1 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
1 3 4 2	2 3 4 1	3 **2** 4 1	4 **2** **3** 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

n	f(n) 都不在正確位置	n! 所有排列	n!-f(n) 有部份在正確位置
1	0	1	1
2	1	2	1
3	2	6	4
4	9	24	15
5	44	120	76

n	f(n)	n!	n!-f(n)
6	265	720	455
7			
8			
9			
10			

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3

1 2 3
1 3 2
2 1 **3**
2 3 1
3 1 2
3 **2** 1

n=4

1 2 3 4	2 1 **3 4**	3 1 2 **4**	4 1 2 3
1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
1 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
1 3 4 2	2 3 4 1	3 **2** 4 1	4 **2 3** 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

	都不在正確位置	所有排列	有部份在正確位置
n	f(n)	n!	n!-f(n)
1	0	1	1
2	1	2	1
3	2	6	4
4	9	24	15
5	44	120	76

n	f(n)	n!	n!-f(n)
6	265	720	455
7	1854	5040	3186
8			
9			
10			

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3

1 **2** **3**
1 3 2
2 1 **3**
2 3 1
3 1 2
3 **2** 1

n=4

1 **2** **3** **4**	2 1 **3** **4**	3 1 2 **4**	4 1 2 3
1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
1 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
1 3 4 2	2 3 4 1	3 **2** 4 1	4 **2** **3** 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

n	f(n) 都不在正確位置	n! 所有排列	n!-f(n) 有部份在正確位置
1	0	1	1
2	1	2	1
3	2	6	4
4	9	24	15
5	44	120	76

n	f(n)	n!	n!-f(n)
6	265	720	455
7	1854	5040	3186
8	14833	40320	25487
9			
10			

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one's foot in the door 的好方法，有機會可以分析出解答的特性

n=3

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

n=4

1 2 3 4	2 1 3 4	3 1 2 4	4 1 2 3
1 2 4 3	2 1 4 3	3 1 4 2	4 1 3 2
1 3 2 4	2 3 1 4	3 2 1 4	4 2 1 3
1 3 4 2	2 3 4 1	3 2 4 1	4 2 3 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 3 2	2 4 3 1	3 4 2 1	4 3 2 1

n	f(n) 都不在正確位置	n! 所有排列	n!-f(n) 有部份在正確位置
1	0	1	1
2	1	2	1
3	2	6	4
4	9	24	15
5	44	120	76

n	f(n)	n!	n!-f(n)
6	265	720	455
7	1854	5040	3186
8	14833	40320	25487
9	133496	362880	229384
10			

方法一: 暴力列舉所有 n! 種排列

- 由於 2≤n≤800, 暴力列舉在 n=13,14 左右就已經需要 1 秒了
- 不過列舉一些小數字的解是 get one’s foot in the door 的好方法，有機會可以分析出解答的特性

n=3	n=4			
1 2 3	**1 2 3 4**	2 1 **3 4**	3 1 2 **4**	4 1 2 3
1 3 2	1 2 4 3	2 1 4 3	3 1 4 2	4 1 **3** 2
2 1 **3**	**1** 3 2 **4**	2 3 1 **4**	3 **2** 1 **4**	4 **2** 1 3
2 3 1	**1** 3 4 2	2 3 4 1	3 **2** 4 1	4 **2 3** 1
3 1 2	**1** 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
3 **2** 1	**1** 4 **3** 2	2 4 **3** 1	3 4 2 1	4 3 2 1

n	f(n) 都不在正確位置	n! 所有排列	n!-f(n) 有部份在正確位置
1	0	1	1
2	1	2	1
3	2	6	4
4	9	24	15
5	44	120	76

n	f(n)	n!	n!-f(n)
6	265	720	455
7	1854	5040	3186
8	14833	40320	25487
9	133496	362880	229384
10	1334961	3628800	22293839

方法二: 遞迴公式一

- f(n) - 所有 n 個物品都不在原來的位置上的數量

方法二: 遞迴公式一

- f(n) - 所有 n 個物品都不在原來的位置上的數量
- n 個物品所有可能的排列方式有 n! 種

方法二: 遞迴公式一

- f(n) - 所有 n 個物品都不在原來的位置上的數量
- n 個物品所有可能的排列方式有 n! 種
- n 個物品中 i 個在原來位置上, 有 C(n,i) 種選擇方法

方法二: 遞迴公式一

- f(n) - 所有 n 個物品都不在原來的位置上的數量
- n 個物品所有可能的排列方式有 n! 種
- n 個物品中 i 個在原來位置上, 有 C(n,i) 種選擇方法
- 剩餘 n - i 個物品不在原來位置上, 數量有 f(n-i) 種

$$f(n) = n! - \sum_{i=1}^{n-1} C(n,i)\, f(n-i)$$

方法二: 遞迴公式一

- f(n) - 所有 n 個物品都不在原來的位置上的數量
- n 個物品所有可能的排列方式有 n! 種
- n 個物品中 i 個在原來位置上, 有 C(n,i) 種選擇方法
- 剩餘 n - i 個物品不在原來位置上, 數量有 f(n-i) 種

$$f(n) = n! - \sum_{i=1}^{n-1} C(n,i)\, f(n-i)$$

- 不考慮 C(n,i) 的運算量, 根據這個遞迴公式由 f(1), …, f(n-1) 計算 f(n) 只需要 n 次大整數的加法和 n 次大整數的乘法

方法二: 遞迴公式一

- f(n) - 所有 n 個物品都不在原來的位置上的數量
- n 個物品所有可能的排列方式有 n! 種
- n 個物品中 i 個在原來位置上, 有 C(n,i) 種選擇方法
- 剩餘 n - i 個物品不在原來位置上, 數量有 f(n-i) 種

$$f(n) = n! - \sum_{i=1}^{n-1} C(n,i)\, f(n-i)$$

- 不考慮 C(n,i) 的運算量, 根據這個遞迴公式由 f(1), …, f(n-1) 計算 f(n) 只需要 n 次大整數的加法和 n 次大整數的乘法
- 如果 C(m,i), 0≤i≤m, m<n 都記憶起來, 則根據下列公式計算 n-1 個 C(n,i) 需要 n-1 次大整數加法

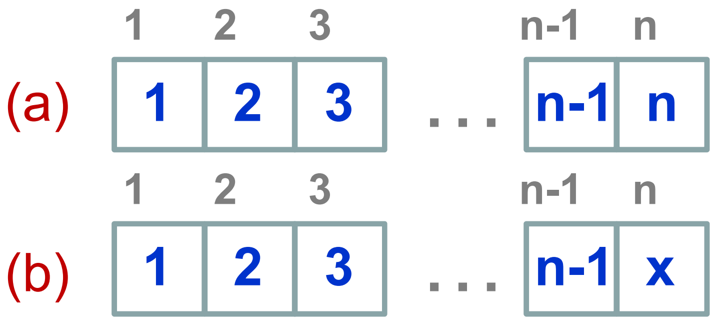
$$C(n,i) = C(n-1,i-1) + C(n-1,i)$$

方法三: 遞迴公式二

	1	2	3		n-1	n
(a)	**1**	**2**	**3**	. . .	**n-1**	**n**

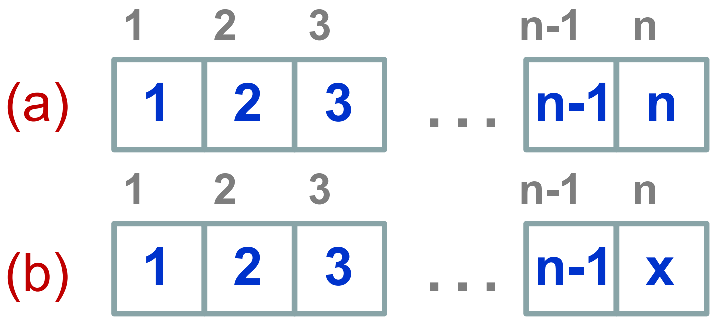
- f(n) - 物品 1,2,…,n 都不在自己位置上的排列方法數量

方法三: 遞迴公式二



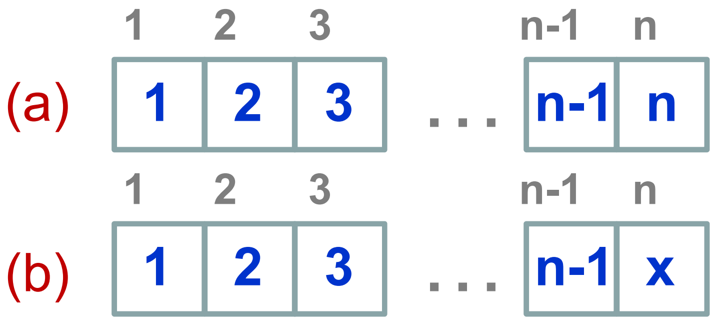
- f(n) - 物品 1,2,…,n 都不在自己位置上的排列方法數量

方法三: 遞迴公式二



- f(n) - 物品 1,2,…,n 都不在自己位置上的排列方法數量
- g(n) - 物品 1,2,…,n-1, x (x>n) 都不在自己位置上的排列方法數量

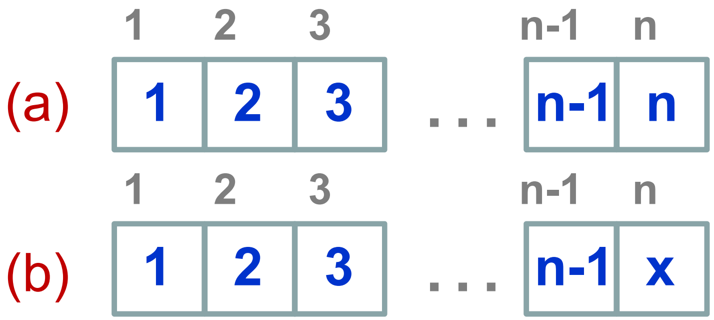
方法三: 遞迴公式二



- f(n) - 物品 1,2,…,n 都不在自己位置上的排列方法數量
- g(n) - 物品 1,2,…,n-1, x (x>n) 都不在自己位置上的排列方法數量

- 物品 x 在位置 1~n 上都不在自己位置上

方法三: 遞迴公式二



- f(n) - 物品 1,2,…,n 都不在自己位置上的排列方法數量
- g(n) - 物品 1,2,…,n-1, x (x>n) 都不在自己位置上的排列方法數量

- 物品 x 在位置 1~n 上都不在自己位置上

- 物品 x 取代 1~n 中任一位置上物件應該有相同的結果

方法三: 遞迴公式二

(a)

1	2	3		n-1	n
1	2	3	. . .	n-1	n

(b)

1	2	3		n-1	n
1	2	3	. . .	n-1	x

- f(n) - 物品 1,2,…,n 都不在自己位置上的排列方法數量
- g(n) - 物品 1,2,…,n-1, x (x>n) 都不在自己位置上的排列方法數量

- 物品 x 在位置 1~n 上都不在自己位置上

- 物品 x 取代 1~n 中任一位置上物件應該有相同的結果

- (b) 中前 n-1 個物品不在自己位置上的排列方法有 f(n-1) 種，加上 x 是一種 n 個物品都不在自己位置上的排列；另外每一種也可以挑選任意第 i 個位置上的物品和 x 交換，此時 n 個物品都不在自己位置上，共有 f(n-1) + (n-1) f(n-1) = **n f(n-1)** 種 (x在位置 i , i 不在位置 n)

方法三: 遞迴公式二

(a)

1	2	3		n-1	n
1	**2**	**3**	. . .	**n-1**	**n**

(b)

1	2	3		n-1	n
1	**2**	**3**	. . .	**n-1**	**x**

- f(n) - 物品 1,2,…,n 都不在自己位置上的排列方法數量
- g(n) - 物品 1,2,…,n-1, x (x>n) 都不在自己位置上的排列方法數量

- 物品 x 在位置 1~n 上都不在自己位置上

- 物品 x 取代 1~n 中任一位置上物件應該有相同的結果

- (b) 中前 n-1 個物品不在自己位置上的排列方法有 f(n-1) 種，加上 x 是一種 n 個物品都不在自己位置上的排列；另外每一種也可以挑選任意第 i 個位置上的物品和 x 交換，此時 n 個物品都不在自己位置上，共有 f(n-1) + (n-1) f(n-1) = **n f(n-1)** 種 (x在位置 i , i 不在位置 n)
- (b) 中物品 x 可以先和前 n-1 個物品中任選的第 i 個物品交換，此時物品 i 和物品 x 都不在自己位置上，剩下 n-2 個物品總共有 f(n-2) 種不在自己位置上的排列方法，共有 **(n-1) f(n-2)** 種排列 (x 在位置 i , i 在位置 n)

方法三: 遞迴公式二

	1	2	3		n-1	n
(a)	**1**	**2**	**3**	. . .	**n-1**	**n**

	1	2	3		n-1	n
(b)	**1**	**2**	**3**	. . .	**n-1**	**x**

- f(n) - 物品 1,2,…,n 都不在自己位置上的排列方法數量
- g(n) - 物品 1,2,…,n-1, x (x>n) 都不在自己位置上的排列方法數量

- 物品 x 在位置 1~n 上都不在自己位置上
- 物品 x 取代 1~n 中任一位置上物件應該有相同的結果

- (b) 中前 n-1 個物品不在自己位置上的排列方法有 f(n-1) 種，加上 x 是一種 n 個物品都不在自己位置上的排列；另外每一種也可以挑選任意第 i 個位置上的物品和 x 交換，此時 n 個物品都不在自己位置上，共有 f(n-1) + (n-1) f(n-1) = **n f(n-1)** 種 (x在位置 i , i 不在位置 n)
- (b) 中物品 x 可以先和前 n-1 個物品中任選的第 i 個物品交換，此時物品 i 和物品 x 都不在自己位置上，剩下 n-2 個物品總共有 f(n-2) 種不在自己位置上的排列方法，共有 **(n-1) f(n-2)** 種排列 (x 在位置 i , i 在位置 n)

$$g(n) = n\,f(n-1) + (n-1)\,f(n-2)$$

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 g(n-1) 且 **f(n) = (n-1) g(n-1)**

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 $g(n-1)$ 且 $f(n) = (n-1)\, g(n-1)$
- $g(n) = n\, f(n-1) + (n-1)\, f(n-2)$ 與 $f(n) = (n-1)\, g(n-1)$ 兩式可以合併為

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 g(n-1) 且 $f(n) = (n-1)\ g(n-1)$

- $g(n) = n\ f(n-1) + (n-1)\ f(n-2)$ 與 $f(n) = (n-1)\ g(n-1)$ 兩式可以合併為

$$f(n) = (n-1)\ ((n-1)\ f(n-2) + (n-2)\ f(n-3)),\ f(1)=0,\ f(2)=1,\ f(3)=2$$

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 g(n-1) 且 **$f(n) = (n-1)\, g(n-1)$**

- **$g(n) = n\, f(n-1) + (n-1)\, f(n-2)$** 與 **$f(n) = (n-1)\, g(n-1)$** 兩式可以合併為

  $f(n) = (n-1)\, ((n-1)\, f(n-2) + (n-2)\, f(n-3)),\ f(1)=0,\ f(2)=1,\ f(3)=2$

例如：

$f(1) = 0$

$f(2) = 1$

$f(3) = 2$

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 g(n-1) 且 **f(n) = (n-1) g(n-1)**

- **g(n) = n f(n-1) + (n-1) f(n-2)** 與 **f(n) = (n-1) g(n-1)** 兩式可以合併為

f(n) = (n-1) ((n-1) f(n-2) + (n-2) f(n-3)), f(1)=0, f(2)=1, f(3)=2

例如：

f(1) = 0

f(2) = 1

f(3) = 2

f(4) = 3(3*1+2*0) = 9

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 g(n-1) 且 $f(n) = (n-1)\ g(n-1)$

- $g(n) = n\ f(n-1) + (n-1)\ f(n-2)$ 與 $f(n) = (n-1)\ g(n-1)$ 兩式可以合併為

$$f(n) = (n-1)\ ((n-1)\ f(n-2) + (n-2)\ f(n-3)),\ f(1)=0,\ f(2)=1,\ f(3)=2$$

例如：

$f(1) = 0$

$f(2) = 1$

$f(3) = 2$

$f(4) = 3(3*1+2*0) = 9$

$f(5) = 4(4*2+3*1) = 44$

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 g(n-1) 且 **$f(n) = (n-1)\ g(n-1)$**

- **$g(n) = n\ f(n-1) + (n-1)\ f(n-2)$** 與 **$f(n) = (n-1)\ g(n-1)$** 兩式可以合併為

$f(n) = (n-1)\ ((n-1)\ f(n-2) + (n-2)\ f(n-3)),\ f(1)=0,\ f(2)=1,\ f(3)=2$

例如：

$f(1) = 0$

$f(2) = 1$

$f(3) = 2$

$f(4) = 3(3*1+2*0) = 9$

$f(5) = 4(4*2+3*1) = 44$

$f(6) = 5(5*9+4*2) = 265$

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 g(n-1) 且 **$f(n) = (n-1)\, g(n-1)$**

- **$g(n) = n\, f(n-1) + (n-1)\, f(n-2)$** 與 **$f(n) = (n-1)\, g(n-1)$** 兩式可以合併為

$$f(n) = (n-1)\,((n-1)\, f(n-2) + (n-2)\, f(n-3)),\ f(1)=0,\ f(2)=1,\ f(3)=2$$

例如：

$f(1) = 0$

$f(2) = 1$

$f(3) = 2$

$f(4) = 3(3*1+2*0) = 9$

$f(5) = 4(4*2+3*1) = 44$

$f(6) = 5(5*9+4*2) = 265$

$f(7) = 6(6*44+5*9) = 6*309 = 1854$

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 g(n-1) 且 **f(n) = (n-1) g(n-1)**

- **g(n) = n f(n-1) + (n-1) f(n-2)** 與 **f(n) = (n-1) g(n-1)** 兩式可以合併為

f(n) = (n-1) ((n-1) f(n-2) + (n-2) f(n-3)), f(1)=0, f(2)=1, f(3)=2

例如：

f(1) = 0
f(2) = 1
f(3) = 2
f(4) = 3(3*1+2*0) = 9
f(5) = 4(4*2+3*1) = 44
f(6) = 5(5*9+4*2) = 265
f(7) = 6(6*44+5*9) = 6*309 = 1854
f(8) = 7(7*265+6*44) = 7(1855+264) =14833

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 g(n-1) 且 **f(n) = (n-1) g(n-1)**

- **g(n) = n f(n-1) + (n-1) f(n-2)** 與 **f(n) = (n-1) g(n-1)** 兩式可以合併為

f(n) = (n-1) ((n-1) f(n-2) + (n-2) f(n-3)), f(1)=0, f(2)=1, f(3)=2

例如：

f(1) = 0
f(2) = 1
f(3) = 2
f(4) = 3(3*1+2*0) = 9
f(5) = 4(4*2+3*1) = 44
f(6) = 5(5*9+4*2) = 265
f(7) = 6(6*44+5*9) = 6*309 = 1854
f(8) = 7(7*265+6*44) = 7(1855+264) =14833
f(9) = 8(8*1854+7*265) = 8(14832+1855)
= 133496

方法三: 遞迴公式二 (續)

- (a) 中前 n-1 個物品任意挑第 i 個和第 n 個物品交換，則前面 n-1 個物品中有一個是物品 n，此時前 n-1 個位置上物品不放在自己位置上的數量為 g(n-1) 且 **$f(n) = (n-1)\ g(n-1)$**

- **$g(n) = n\ f(n-1) + (n-1)\ f(n-2)$** 與 **$f(n) = (n-1)\ g(n-1)$** 兩式可以合併為

  $f(n) = (n-1)\ ((n-1)\ f(n-2) + (n-2)\ f(n-3)),\ f(1)=0,\ f(2)=1,\ f(3)=2$

例如：

$f(1) = 0$

$f(2) = 1$

$f(3) = 2$

$f(4) = 3(3*1+2*0) = 9$

$f(5) = 4(4*2+3*1) = 44$

$f(6) = 5(5*9+4*2) = 265$

$f(7) = 6(6*44+5*9) = 6*309 = 1854$

$f(8) = 7(7*265+6*44) = 7(1855+264) =14833$

$f(9) = 8(8*1854+7*265) = 8(14832+1855)$
$= 133496$

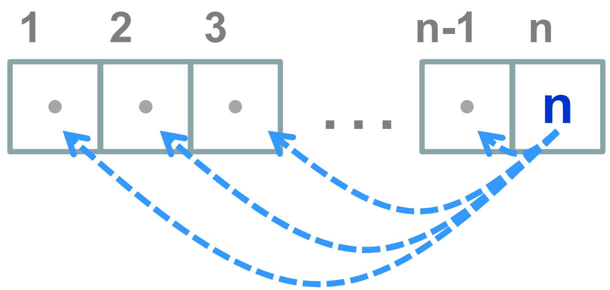
$f(10) = 9(9*14833+8*1854)$
$= 9\ (133497+14832) = 1334961$

⋮

方法四: 遞迴公式三

- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

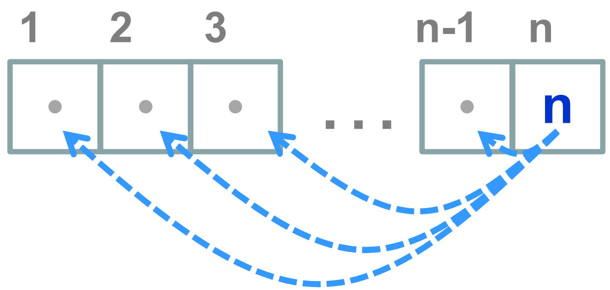
方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**,

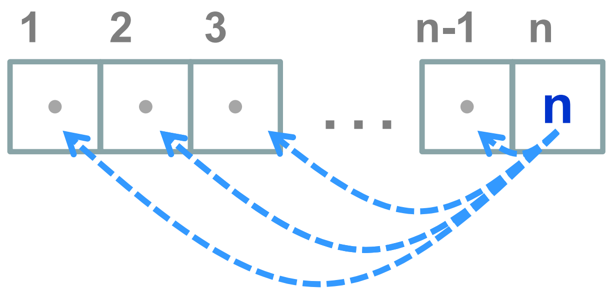
方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有兩種可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

方法四: 遞迴公式三

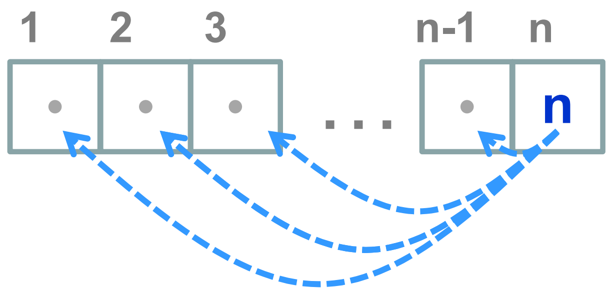


- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有**兩種**可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種

方法四: 遞迴公式三



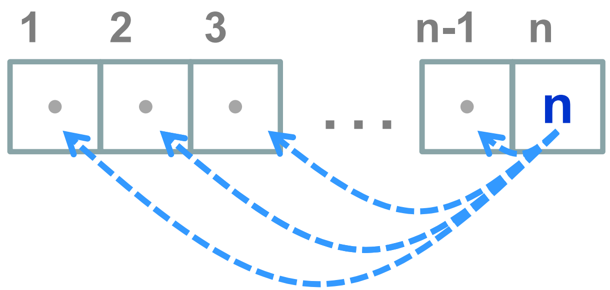
- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有**兩種**可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種

2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 **f(n-1) (n-1)** 種

方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

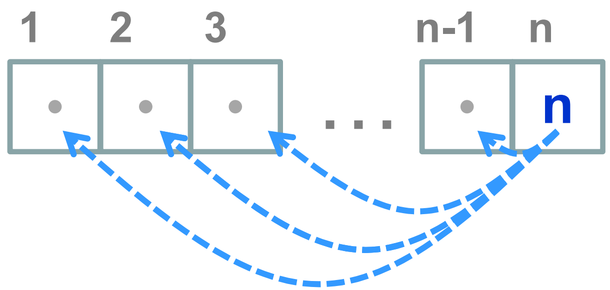
第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有**兩種**可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種

2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 **f(n-1) (n-1)** 種

$$f(n) = (n-1)\,(f(n-1) + f(n-2))$$

方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有**兩種**可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種

2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 **f(n-1) (n-1)** 種

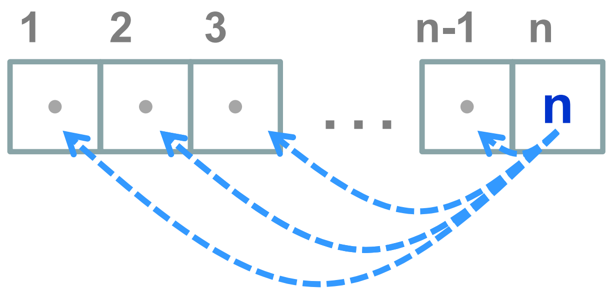
例如：

$f(1) = 0$

$f(2) = 1$

$$f(n) = (n-1)\,(f(n-1) + f(n-2))$$

方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有**兩種**可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種

2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 **f(n-1) (n-1)** 種

例如：

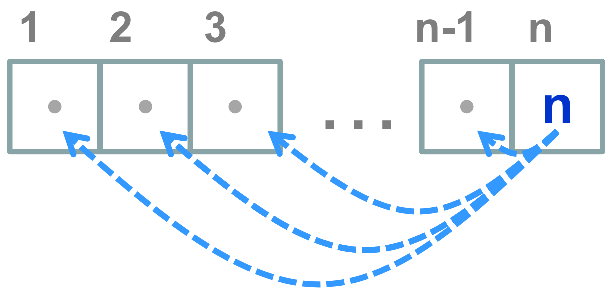
f(1) = 0

f(2) = 1

f(3) = 2 = 2 (1+0)

f(n) = (n-1) (f(n-1) + f(n-2))

方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的第 i 個位置, 第 n 個位置放的物品則有兩種可能: 1.第 i 個物品或是 2.其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 (n-1) f(n-2) 種
2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 f(n-1) (n-1) 種

例如：

$f(n) = (n-1)\,(f(n-1) + f(n-2))$

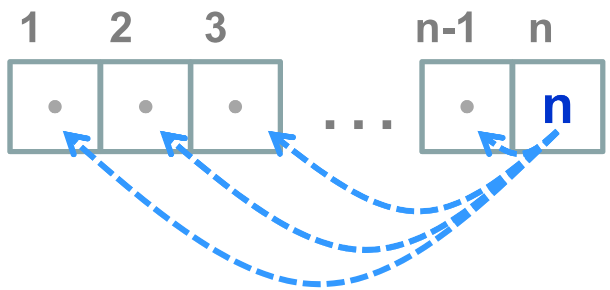
$f(1) = 0$

$f(2) = 1$

$f(3) = 2 = 2\,(1+0)$

$f(4) = 9 = 3\,(2+1)$

方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有兩種可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種

2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 **f(n-1) (n-1)** 種

例如：

f(1) = 0

f(2) = 1

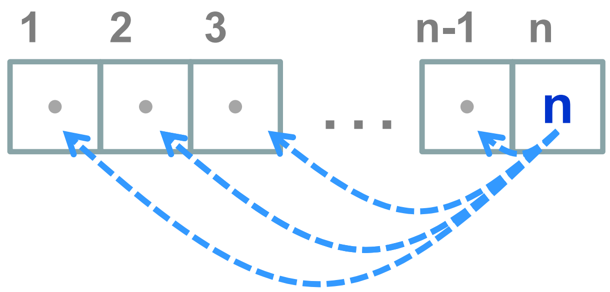
f(3) = 2 = 2 (1+0)

f(4) = 9 = 3 (2+1)

f(5) = 44 = 4 (9+2)

f(n) = (n-1) (f(n-1) + f(n-2))

方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有**兩種**可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種
2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 **f(n-1) (n-1)** 種

例如：

f(n) = (n-1) (f(n-1) + f(n-2))

f(1) = 0

f(2) = 1

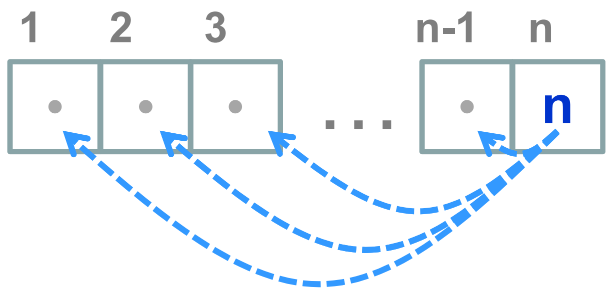
f(3) = 2 = 2 (1+0)

f(4) = 9 = 3 (2+1)

f(5) = 44 = 4 (9+2)

f(6) = 265 = 5 (44+9)

方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有**兩種**可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種
2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 **f(n-1) (n-1)** 種

例如：

$$f(n) = (n-1)\,(f(n-1) + f(n-2))$$

f(1) = 0

f(2) = 1

f(3) = 2 = 2 (1+0)

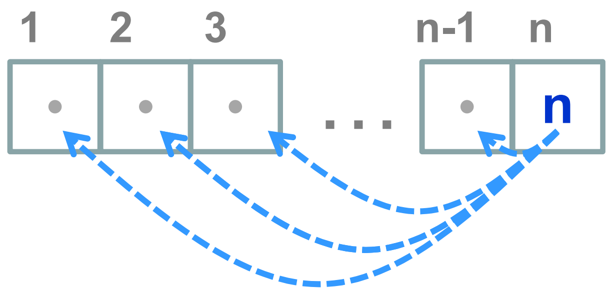
f(4) = 9 = 3 (2+1)

f(5) = 44 = 4 (9+2)

f(6) = 265 = 5 (44+9)

f(7) = 1854 = 6 (265+44)

方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有**兩種**可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種
2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 **f(n-1) (n-1)** 種

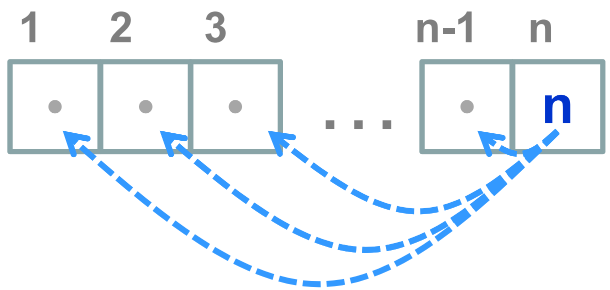
例如：

f(n) = (n-1) (f(n-1) + f(n-2))

f(1) = 0
f(2) = 1
f(3) = 2 = 2 (1+0)
f(4) = 9 = 3 (2+1)
f(5) = 44 = 4 (9+2)
f(6) = 265 = 5 (44+9)

f(7) = 1854 = 6 (265+44)
f(8) = 14833 = 7 (1854+265)

方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有**兩種**可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種

2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 **f(n-1) (n-1)** 種

例如：

f(n) = (n-1) (f(n-1) + f(n-2))

f(1) = 0

f(2) = 1

f(3) = 2 = 2 (1+0)

f(4) = 9 = 3 (2+1)

f(5) = 44 = 4 (9+2)

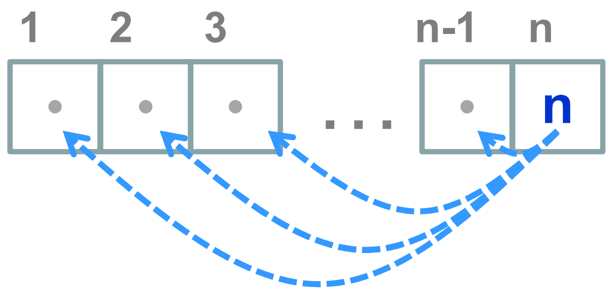
f(6) = 265 = 5 (44+9)

f(7) = 1854 = 6 (265+44)

f(8) = 14833 = 7 (1854+265)

f(9) = 133496 = 8 (14833+1854)

方法四: 遞迴公式三



- 假設我們已經知道 f(n-1) 及 f(n-2),考慮第 n 個物品/位置如下:

第 n 個物品不能放在第 n 個位置, 一定要放在前 n-1 個位置中的**第 i 個位置**, **第 n 個位置**放的物品則有**兩種**可能: **1.**第 i 個物品或是 **2.**其它 n-2 個物品之一

1. 第 n 個物品先和第 i 個物品交換位置, 剩下 n-2 個物品則排出都不在自己位置上的一種排法, 共 **(n-1) f(n-2)** 種

2. 前 n-1 個物品先做出一種不在自己位置上的排法, 再把第 i 個位置上的物品(一定不是第 i 個物品)和第 n 個物品交換位置, 共 **f(n-1) (n-1)** 種

$$f(n) = (n-1)\,(f(n-1) + f(n-2))$$

例如：

f(1) = 0
f(2) = 1
f(3) = 2 = 2 (1+0)
f(4) = 9 = 3 (2+1)
f(5) = 44 = 4 (9+2)
f(6) = 265 = 5 (44+9)

f(7) = 1854 = 6 (265+44)
f(8) = 14833 = 7 (1854+265)
f(9) = 133496 = 8 (14833+1854)
f(10) = 1334961 = 9 (133496+14833)
⋮

簡易大整數相加與相乘實作

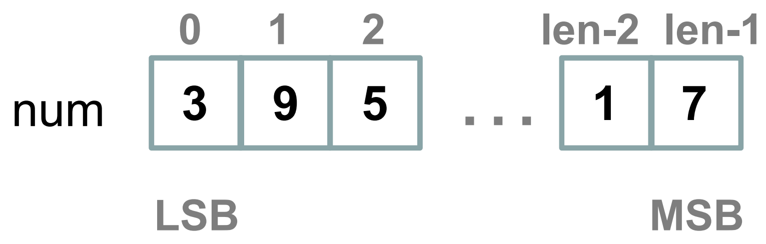
- 結構與字元陣列表示

```
#include <stdio.h>
#include <string.h>
#define MAX 1001
struct BigInt {
    int len;
    char num[MAX];
};
```

簡易大整數相加與相乘實作

- 結構與字元陣列表示

```c
#include <stdio.h>
#include <string.h>
#define MAX 1001
struct BigInt {
    int len;
    char num[MAX];
};
```

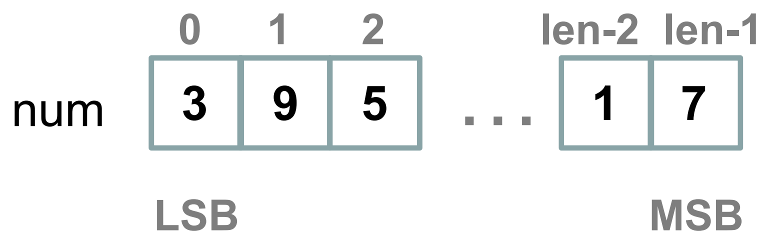


表示十進位數字 71…593

簡易大整數相加與相乘實作

- 結構與字元陣列表示

```
#include <stdio.h>
#include <string.h>
#define MAX 1001
struct BigInt {
    int len;
    char num[MAX];
};
```



表示十進位數字 71…593

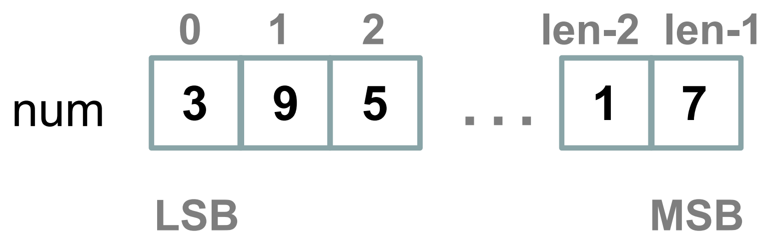
- 整數轉換為大整數

```
void assign(struct BigInt *bn, int x) {
    bn->len = 0;
    while (x)
        bn->num[bn->len++]=x%10, x/=10;
    if (bn->len==0) bn->num[bn->len++]=0;
}
```

簡易大整數相加與相乘實作

- 結構與字元陣列表示

```
#include <stdio.h>
#include <string.h>
#define MAX 1001
struct BigInt {
    int len;
    char num[MAX];
};
```



表示十進位數字 71…593

- 整數轉換為大整數

```
void assign(struct BigInt *bn, int x) {
    bn->len = 0;
    while (x)
        bn->num[bn->len++]=x%10, x/=10;
    if (bn->len==0) bn->num[bn->len++]=0;
}
```

- 列印大整數

```
void print(struct BigInt *bn) {
    int i;
    printf("[");
    for (i=bn->len-1; i>=0; i--)
      printf("%d", bn->num[i]);
    printf("]");
}
```

簡易大整數實作 (cont'd)

- 兩個大整數相加

```
struct BigInt add(const struct BigInt *x,
                  const struct BigInt *y) {
    int carry=0, i;
    struct BigInt r;
    memset(&r, 0, sizeof(r));
    for (i=0; i<x->len || i<y->len; i++) {
        if (i < x->len) carry += x->num[i];
        if (i < y->len) carry += y->num[i];
        r.num[r.len++] = carry%10,
        carry /= 10;
    }
    while (carry)
        r.num[r.len++] = carry%10,
        carry /= 10;
    return r;
}
```

簡易大整數實作 (cont'd)

- 兩個大整數相加

```
struct BigInt add(const struct BigInt *x,
                  const struct BigInt *y) {
    int carry=0, i;
    struct BigInt r;
    memset(&r, 0, sizeof(r));
    for (i=0; i<x->len || i<y->len; i++) {
        if (i < x->len) carry += x->num[i];
        if (i < y->len) carry += y->num[i];
        r.num[r.len++] = carry%10,
        carry /= 10;
    }
    while (carry)
        r.num[r.len++] = carry%10,
        carry /= 10;
    return r;
}
```

- 大整數與整數相乘

```
struct BigInt mult(const struct BigInt *x,
                   const int y) {
    int i, carry=0;
    struct BigInt ans;
    memset(&ans, 0, sizeof(ans));
    for (i=0; i<x->len; i++) {
        carry += y * x->num[i];
        ans.num[ans.len++] = carry%10;
        carry /= 10;
    }
    while (carry)
        ans.num[ans.len++] = carry%10,
        carry /= 10;
    return ans;
}
```

簡易大整數實作 (cont'd)

費氏數列 $f(n) = f(n-1) + f(n-2)$實作

簡易大整數實作 (cont'd)

費氏數列 f(n) = f(n-1) + f(n-2)實作

```
int main() {
    int i;
    struct BigInt fib[401]; memset(fib, 0, sizeof(fib));
    printf("size of struct BigInt[0:400]=%d\n", sizeof(fib));
    assign(&fib[0],0); assign(&fib[1],1);
    for (i=2; i<401; i++)
        fib[i] = add(&fib[i-1], &fib[i-2]), print(&fib[i]);
    for (i=198; i<201; i++)
        print(&fib[i]), printf("\n");
    print(&fib[400]), printf("\n");
    return 0;
}
```

簡易大整數實作 (cont'd)

費氏數列 f(n) = f(n-1) + f(n-2)實作

```
int main() {
    int i;
    struct BigInt fib[401]; memset(fib, 0, sizeof(fib));
    printf("size of struct BigInt[0:400]=%d\n", sizeof(fib));
    assign(&fib[0],0); assign(&fib[1],1);
    for (i=2; i<401; i++)
        fib[i] = add(&fib[i-1], &fib[i-2]), print(&fib[i]);
    for (i=198; i<201; i++)
        print(&fib[i]), printf("\n");
    print(&fib[400]), printf("\n");
    return 0;
}
```

fib[198] = 107168651819712326877926895128666735145224
fib[199] = 173402521172797813159685037284371942044301
fib[200] = 280571172992510140037611932413038677189525
fib[400] = 176023680645013966468226945392411250770384383304492191886
725992896575345044216019675

常見延伸題目

- 如果只要求算出所有可能放置方法的「數量除以 1000000007 的餘數」

常見延伸題目

- 如果只要求算出所有可能放置方法的「數量除以 1000000007 的餘數」
  - 則物品種類 n 的數值還可以再增加

常見延伸題目

- 如果只要求算出所有可能放置方法的「數量除以 1000000007 的餘數」
  - 則物品種類 n 的數值還可以再增加
  - 計算過程和結果不需要使用大整數，只需要使用 long long 即可

常見延伸題目

- 如果只要求算出所有可能放置方法的「數量除以 1000000007 的餘數」
  - 則物品種類 n 的數值還可以再增加
  - 計算過程和結果不需要使用大整數，只需要使用 long long 即可
  - 每次加法和乘法計算的結果都以其除以 1000000007 的餘數取代