

枚舉

temmie

1. 枚舉介紹
2. 全部枚舉
3. 迴圈枚舉
4. 全排列枚舉
5. 位元枚舉

枚舉介紹

枚舉的定義

字 詞	枚舉
注 音	méi jǔ
漢語拼音	méi jǔ
相 似 詞	列舉
釋 義	一一列舉。如：「隨著科技進步，全世界每日新發明的產品，真是不勝枚舉。」金·元好問《故物譜》：「住在鄉里，常侍諸父及兩兄燕談，每及家所有書，則必枚舉而問之。」元·陶宗儀《南村輟耕錄·卷一九·闕駕上書》：「歌曰：『官吏黑漆皮燈籠，奉使來時添一重。』如此怨謠，未能枚舉，皆萬姓不平之氣，鬱結于懷，而發諸聲者然也。」

- 只要把所有需要的情況列出來，然後找哪個是答案

枚舉的用處

- 我們之前有提過高中的比賽都會有部份分數
- 在大部份的比賽裡都會有關於枚舉的子任務
- 通常這些題目不會特別卡枚舉，撈分可說是非常好用
- **我認為這是所有單元裡面 CP 值最高的**

全部枚舉

枚舉的方式

- 把題目給予的所有情況都列出一遍，即可得到答案
- 通常會有三種方式：迴圈枚舉、全排列枚舉、位元枚舉、遞迴枚舉

迴圈枚舉

例題

迴圈枚舉

題目連結

給你一個正整數 N ，請判斷它是不是兩個 $1 \sim 9$ 的數的積

保證 $1 \leq N \leq 100$

迴圈枚舉

- 如果一個題目的參數較少，則可以用 for 迴圈枚舉所有參數
- 另外要注意時間複雜度的限制

全排列枚舉

例題

全排列枚舉

題目連結

給你一個字串 S ，請從從小到大的字典序輸出所有不同排列

保證 $1 \leq |S| \leq 8$

全排列枚舉

- 這時候需要用到一個特殊的語法 `next_permutation(L, R)`
- 他會把 $[L, R)$ 範圍內的替換成下一個字典序的排列
- 並且會回傳是否排列成功

全排列枚舉

- 這時候需要用到一個特殊的語法 `next_permutation(L, R)`
- 他會把 `[L, R)` 範圍內的替換成下一個字典序的排列
- 並且會回傳是否排列成功
- 要做到「全排列」這件事情，其實只需要從最小字典序的排列不斷執行直到 `next_permutation` 不能排列為止
- 時間複雜度為 $O(n!)$ ，大概可以接受到 $n \leq 10$

全排列枚舉模板

```
sort(v.begin(), v.end()); // 用 sort 變成最小字典序
do{
    for (auto x : v){
        cout << x << " ";
    }    cout << endl;
}while(next_permutation(v.begin(), v.end()));
```

- 這裡出現了一個新的語法 do...while
- 和 while 的概念一模一樣，不過是等到裡面的東西執行完再判斷
- 這裡需要使用的原因是因為我們也要判斷第一個排列是否為答案

例題

全排列枚舉

題目連結

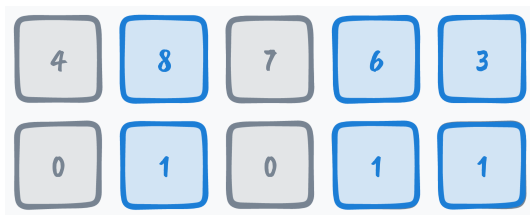
給你 n 個蘋果，每個重量為 p_i ，請問該怎麼分成兩組讓它們重量差最小
保證 $1 \leq n \leq 20$ 且 $1 \leq p_i \leq 10^9$

位元枚舉

- 位元枚舉通常用在「分成兩組」、「拿或不拿」的問題
- 這時候不能使用全排列，有什麼是可以判斷要不要用某個參數呢？

位元枚舉

- 位元枚舉通常用在「分成兩組」、「拿或不拿」的問題
- 這時候不能使用全排列，有什麼是可以判斷要不要用某個參數呢？
- 二進位！我們可以使用二進位來分組



4	8	7	6	3
0	1	0	1	1

位元枚舉

位元枚舉

- 由於要枚舉每一個二進位的可能，其實就是枚舉 $0 \sim 2^n - 1$!
- 原因是因為 n bit 可以組成 $0 \sim 2^n - 1$ 的所有數字

```
// (1<<n) 就是 2^n，不使用 pow() 是因為會有誤差
for (int i=0 ; i<(1<<n) ; i++){
    // 判斷是否為答案
}
```

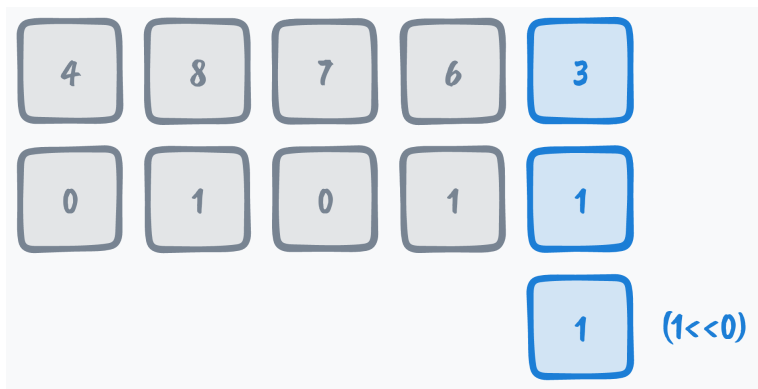
位元枚舉

- 枚舉完一種排列後，就要分組並且計算答案

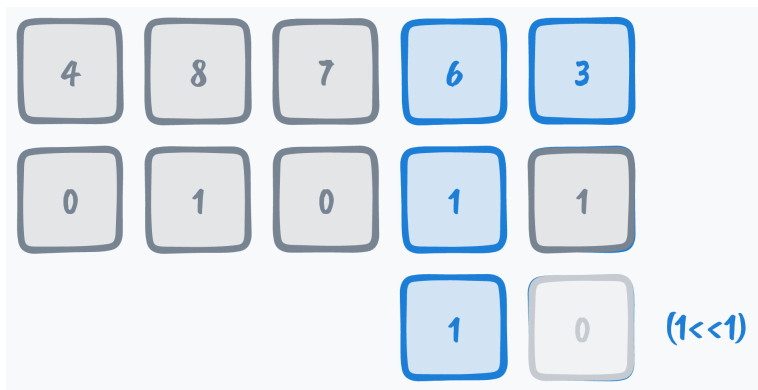
位元枚舉

- 枚舉完一種排列後，就要檢查是否為答案
- 我們可以將 1 不斷的去「掃描」每個位元，並且做 & 運算
- 只要結果不是 0 的話就代表這個為 true，也就是同一組的意思

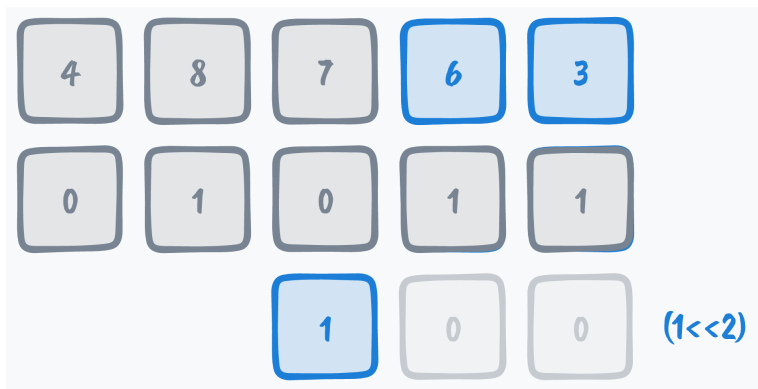
位元枚舉



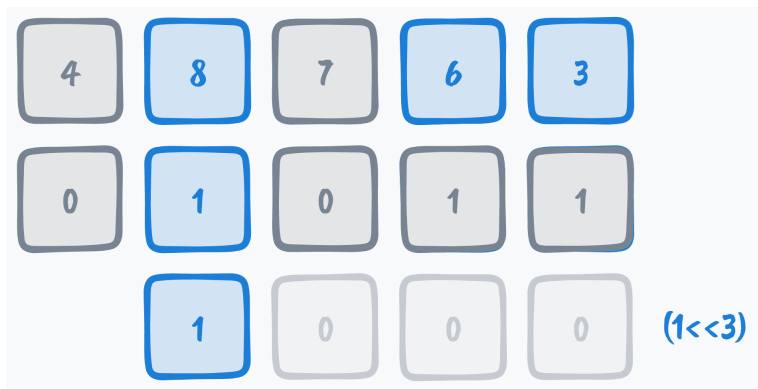
位元枚舉



位元枚舉



位元枚舉



位元枚舉

•

4	8	7	6	3	
0	1	0	1	1	
1	0	0	0	0	$(1 < 4)$

位元枚舉

