

# December 31, 2024 week 1

Presenter : 芝岑

A – Apple Division

There are  $n$  apples with known weights. Your task is to divide the apples into two groups so that the difference between the weights of the groups is minimal.

## Input

The first input line has an integer  $n$ : the number of apples.

The next line has  $n$  integers  $p_1, p_2, \dots, p_n$ : the weight of each apple.

## Output

Print one integer: the minimum difference between the weights of the groups.

## Constraints

- $1 \leq n \leq 20$
- $1 \leq p_i \leq 10^9$

## Example

Input	copy	Output
5 3 2 7 4 1		1

Explanation: Group 1 has weights 2, 3 and 4 (total weight 9), and group 2 has weights 1 and 7 (total weight 8).

範例：

- 假設蘋果重量分別是  $[1, 2, 3, 4, 7]$ ，那麼一種分法是：
  - 第1組： $[2, 3, 4]$ ，重量之和  $= 9$
  - 第2組： $[1, 7]$ ，重量之和  $= 8$
  - 兩組重量差為  $|9 - 8| = 1$ 。
- 確認所有可能的分法後，可以知道最小差為 1，所以答案就是 1。

題目同時給了以下限制：

1.  $1 \leq n \leq 20$
2.  $1 \leq p_i \leq 10^9$

由於  $n$  最大只有 20，但每個蘋果的重量可高達  $10^9$ ，在思考演算法時，要同時考量到「蘋果個數少，但每顆重量可能很大」的情況。

要把全部蘋果分成兩組，等同於「找一個子集合(Subset)」，令這個子集合的權重和盡可能接近「總重量的一半」。

- 若能精準地分成兩組重量相同，差就會是0(完美分割)。
- 如果無法分成兩組一樣重，就找差最小的分法。

因為  $n$  只有 20，直接用「枚舉(bitmask enumeration)」所有子集合的方法是可行的。

**任何一顆蘋果，要嘛放在第一組，要嘛放在第二組**

⇒ 共有  $2^n$  種分組方式。

# Algorithmic Steps

```
vector<int> weights(n);

for (int i = 0; i < n; i++) {
    cin >> weights[i];
}

long long totalSum = 0;
for (int i = 0; i < n; i++) {
    totalSum += weights[i];
}

long long minDifference = LLONG_MAX;

for (int mask = 0; mask < (1 << n); mask++) {
    long long groupSum = 0;
    for (int i = 0; i < n; i++) {
        if (mask & (1 << i)) {
            groupSum += weights[i];
        }
    }

    long long difference = abs(totalSum - 2 * groupSum);
    minDifference = min(minDifference, difference);
}

cout << minDifference << endl;
```



# B – Chessboard and Queens



Your task is to place eight queens on a chessboard so that no two queens are attacking each other. As an additional challenge, each square is either free or reserved, and you can only place queens on the free squares. However, the reserved squares do not prevent queens from attacking each other.

How many possible ways are there to place the queens?

## Input

The input has eight lines, and each of them has eight characters. Each square is either free (  ) or reserved (  ).

## Output

Print one integer: the number of ways you can place the queens.

## Example

Input	Output
<div data-bbox="1049 355 1132 412">copy</div> <pre data-bbox="434 456 566 743">..... ..... ..*..... ..... ..... .....** ..*..... .....</pre>	65

## 不重複放置多個皇后在同一行

- 每次呼叫 `solve(row, board)` 只負責在「第 row 列」放置一個皇后，並在迴圈中遍歷所有欄位。
- 因此保證了「同一行不會出現多個皇后」的條件。

## 對角線index的計算

- 主對角線(像是 \ 方向):  $row - col$  取值會落在  $[-7, 7]$  之間，因此加上 7 讓index變為  $[0, 14]$ 。
- 反對角線(則是 / 方向):  $row + col$  取值落在  $[0, 14]$ ，可直接做為index。

# Algorithmic Steps

```
int countSolutions = 0;
vector<bool> columnUsed(8, false), diag1Used(15, false), diag2Used(15, false);

void solve(int row, vector<string> &board) {
    if (row == 8) {
        countSolutions++;
        return;
    }

    for (int col = 0; col < 8; col++) {
        if (board[row][col] == '*' || columnUsed[col] || diag1Used[row - col + 7] || diag2Used[row + col])
            continue;

        columnUsed[col] = diag1Used[row - col + 7] = diag2Used[row + col] = true;
        solve(row + 1, board);
        columnUsed[col] = diag1Used[row - col + 7] = diag2Used[row + col] = false;
    }
}
```

# C – Coin Piles



You have two coin piles containing  $a$  and  $b$  coins. On each move, you can either remove one coin from the left pile and two coins from the right pile, or two coins from the left pile and one coin from the right pile.

Your task is to efficiently find out if you can empty both the piles.

## Input

The first input line has an integer  $t$ : the number of tests.

After this, there are  $t$  lines, each of which has two integers  $a$  and  $b$ : the numbers of coins in the piles.

## Output

For each test, print "YES" if you can empty the piles and "NO" otherwise.

## Constraints

- $1 \leq t \leq 10^5$
- $0 \leq a, b \leq 10^9$

## Example


Input	copy	Output	copy
3		YES	
2 1		NO	
2 2		YES	
3 3			

1.  $a = 2, b = 1$  : 總和  $2 + 1 = 3$  , 是 3 的倍數 ; 且  $\min(2, 1) \times 2 \geq \max(2, 1)$  。輸出 "YES" 。
2.  $a = 2, b = 2$  : 總和  $2 + 2 = 4$  , 不是 3 的倍數 。輸出 "NO" 。
3.  $a = 3, b = 3$  : 總和  $3 + 3 = 6$  , 是 3 的倍數 ; 且  $\min(3, 3) \times 2 \geq \max(3, 3)$  。輸出 "YES" 。

# Algorithmic Steps

總硬幣數符合 3 的倍數, 若否, 無法恰好把所有硬幣在「3 枚 3 枚」的操作下清空。

```
bool canEmptyPiles(long long a, long long b) {  
    return (a + b) % 3 == 0 && 2 * min(a, b) >= max(a, b);  
}
```



用來檢查兩堆硬幣之間的數量落差不能過大, 確保「小堆」硬幣能夠一直搭配「大堆」的硬幣來做 1 枚與 2 枚的配合移除。



# D - Palindrome Reorder



# D - Palindrome Reorder

Given a string, your task is to reorder its letters in such a way that it becomes a palindrome (i.e., it reads the same forwards and backwards).

## Input

The only input line has a string of length  $n$  consisting of characters A–Z.

## Output

Print a palindrome consisting of the characters of the original string. You may print any valid solution. If there are no solutions, print "NO SOLUTION".

## Constraints

- $1 \leq n \leq 10^6$

# Example

**Input**

copy

AAAACACBA

**Output**

copy

AACABACAA

# Algorithmic Steps

```
string input;  
cin >> input;  
  
vector<long long> freq(26, 0);
```

We use a `vector<long long>` of size 26 to store the frequency of each letter ('A' through 'Z').

`input[index] - 'A'` gives us the appropriate index (0 to 25).

Increment `freq[...]` to record how many times each letter appears

```
index = 0;  
while (index < 26) {  
    if (freq[index] % 2 != 0) {  
        countOdd++;  
        oddIndex = index;  
  
        if (countOdd > 1) {  
            cout << "NO SOLUTION\n";  
            return 0;  
        }  
    }  
    index++;  
}
```

```
int index = 0;  
while (index < (int)input.size()) {  
    freq[input[index] - 'A']++;  
    index++;  
}
```

```
string half;  
half.reserve(input.size() / 2);  
  
index = 0;  
while (index < 26) {  
    half.append(freq[index] / 2, static_cast<char>('A' + index));  
    index++;  
}  
  
string middle = "";  
if (countOdd == 1) {  
    middle.push_back(static_cast<char>('A' + oddIndex));  
}  
string reversedHalf = half;  
reverse(reversedHalf.begin(), reversedHalf.end());  
cout << half << middle << reversedHalf << "\n";  
  
return 0;
```

Thank you

Presenter: 芝岑