PD2017 LAB EXAM I

讓我們突破 100 分！！！

Problem 1

Problem 1

- 給一個「由小排到大」的數列。
- 如果數列裡面某一項是數列中「另一項」的平方，即輸出此數。
- 由小到大輸出。

Problem 1

- 1 3 5 9 25

Problem 1

- | 3 5 9 25

Problem I

- I 3 5 9 25
- I 是特殊狀況，不輸出

Problem 1

- 1 3 5 9 25
- 1 是特殊狀況，不輸出
- 其餘數字可以直接判斷

Problem 1

```
for(int i=0; i<n; i++){
    if(arr[i] == 1) continue;
    bool flag = false;
    for(int j=0; j<n; j++){
        if(arr[i]==arr[j]*arr[j]){
            flag = true;
            break;
        }
    }
    if(flag){
        // 符合條件
    }
}
```

Problem 1

- [1 3 5] 9 25
- 除了 1 以外，所有的數字只需要考慮比它小的即可

Problem 1

```
for(int i=0; i<n; i++){
    bool flag = false;
    for(int j=0; j<i; j++){
        if(arr[i] == arr[j]*arr[j]){
            flag = true;
            break;
        }
    }
    if(flag){
        //符合條件
    }
}
```

Problem I

- 到目前為止的演算法都是 $O(N^2)$

Problem I

- 到目前為止的演算法都是 $O(N^2)$
- 來一個更精妙的做法

Problem 1

- 1 3 5 9 25

Problem 1

- 1 3 5 9 25
- 我們考慮數字只會越考慮越大

Problem 1

- 1 3 5 9 25
- 我們考慮數字只會越考慮越大
- 我們可以忽略某些一定比我們還小的數

Problem 1

- 1 3 5 9 25
- 因為 1 3 不比 9 大

Problem 1

- 1 3 5 9 25
- 因為 1 3 不比 9 大
- 9 比 25 小

Problem 1

- 1 3 5 9 25
- 因為 1 3 不比 9 大
- 9 比 25 小
- 所以對 25 來說，1 3 皆不用列入考慮

Problem 1

```
int le = 0;
for(int i = 0; i < n; i++) {
    while(le < i && arr[le] * arr[le] < arr[i]) {
        le++;
    }
    if(le != i && arr[le] * arr[le] == arr[i]) {
        // 符合條件
    }
}
```

Problem I 總結

- 加入一個左指針，使迴圈有雙指針
- 時間複雜度 O(N)
- 整個做法必須在「數列已排序」情況下成立

Problem 2

Problem 2

- 給一個二維矩陣。
- 問「方形」子矩陣和為 0 的個數有幾個？

Problem 2

- 一個想法：如何唯一的表達一個矩陣？

Problem 2

- 一個想法：如何唯一的表達一個矩陣？
- 左上角座標、長（直）、寬（橫）

Problem 2

- 一個想法：如何唯一的表達一個矩陣？
- 左上角座標、長（直）、寬（橫）
- 方形子矩陣只需要座標、邊長

Problem 2

- 如何枚舉所有方形子矩陣？

Problem 2

- 如何枚舉所有方形子矩陣？
- 二維迴圈枚舉左上角座標

Problem 2

- 如何枚舉所有方形子矩陣？
- 二維迴圈枚舉左上角座標
- 一維迴圈枚舉可能的邊長

Problem 2

```
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        for(int k=1; k<=n; k++){
            //座標(i,j)、邊長 k
        }
    }
}
```

Problem 2

```
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        for(int k=1; k<=n; k++){
            //座標(i,j)、邊長 k
        }
    }
}
```

- 邊長超出陣列怎麼辦？

Problem 2

```
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        for(int k=1; i+k<=n && j+k<=n; k++){
            //座標(i,j)、邊長 k
        }
    }
}
```

Problem 2

```
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        for(int k=1; i+k<=n && j+k<=n; k++){
            //座標(i,j)、邊長 k
        }
    }
}
```

```
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        for(int k=1; k<=n; k++){
            if(i+k<=n && j+k<=n){
                //座標(i,j)、邊長 k
            }
        }
    }
}
```

Problem 2

- 三維迴圈枚舉方形矩陣

Problem 2

- 三維迴圈枚舉方形矩陣
- 二維迴圈加總矩陣！

Problem 2

```
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        for(int k=1; i+k<=n && j+k<=n; k++){
            int sum = 0;
            for(int p=0; p<k; p++){
                for(int q=0; q<k; q++){
                    sum += arr[i+p][j+q];
                }
            }
            // sum -> 子矩陣和
        }
    }
}
```

Problem 2

- 五層迴圈！ $O(N^5)$！

Problem 2

- 五層迴圈！ $O(N^5)$！
- 來一個更精妙的做法！

Problem 2

- 五層迴圈！ $O(N^5)$！
- 來一個更精妙的做法！
- 排容原理！

Problem 2

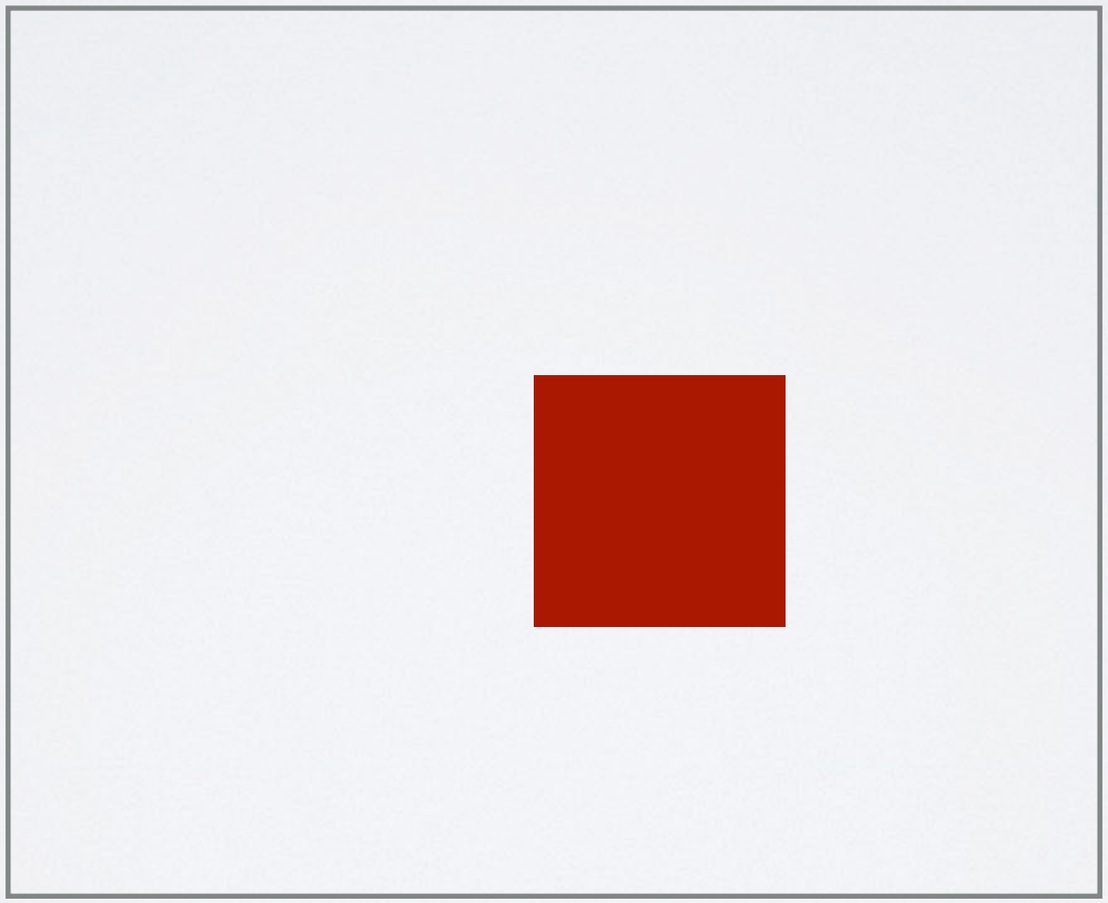
- 求矩陣和？

Problem 2

- 求矩陣和？

Problem 2

- 求矩陣和？



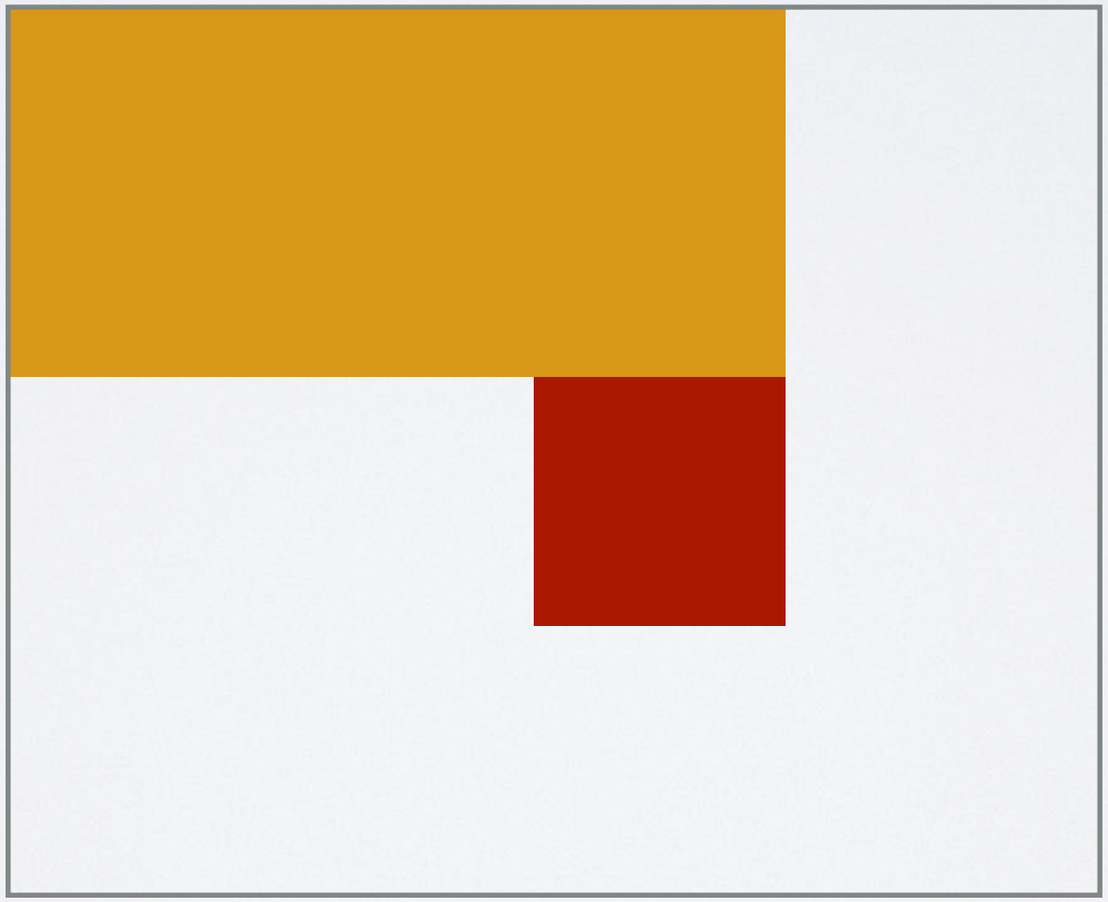
Problem 2

- 求矩陣和？

目標

Problem 2

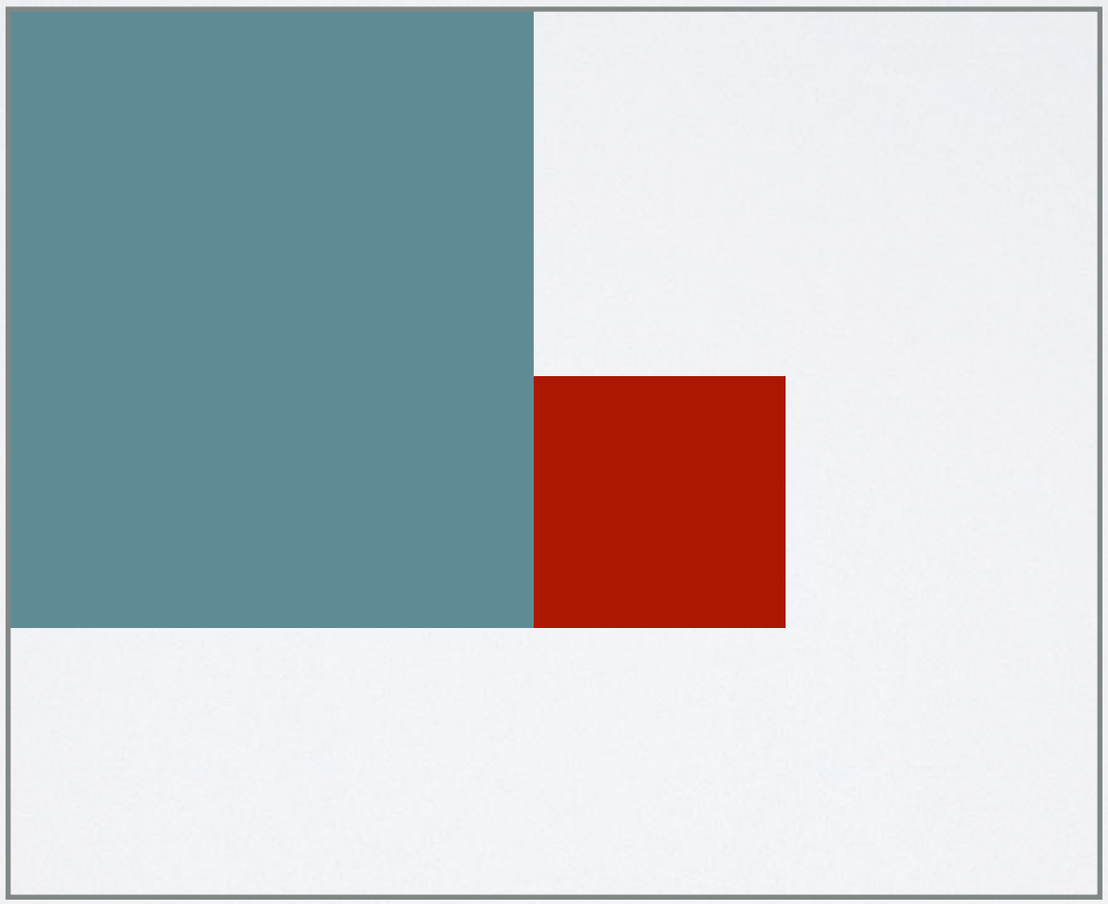
- 求矩陣和？



目標

Problem 2

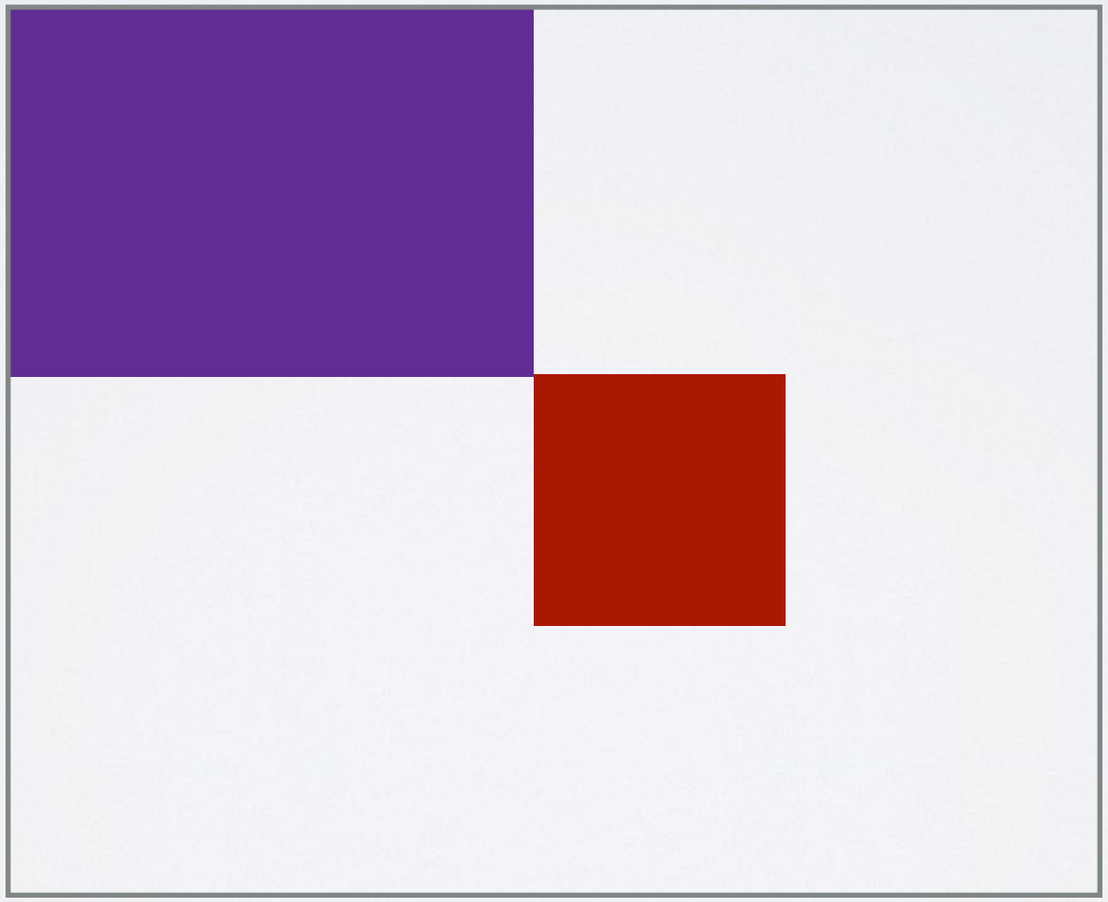
- 求矩陣和？



目標

Problem 2

- 求矩陣和？



Problem 2

目標

加

減

減

加

Problem 2

- 我們只要能求出每個座標的左上角和，就能一瞬間算出任意一個矩形的總和。

Problem 2

- 我們只要能求出每個座標的左上角和，就能一瞬間算出任意一個矩形的總和。
- 怎麼求左上角和？

Problem 2

- 我們只要能求出每個座標的左上角和，就能一瞬間算出任意一個矩形的總和。

- 怎麼求左上角和？

- 把排容反過來做！

Problem 2

- 如果我們要求目標和

Problem 2

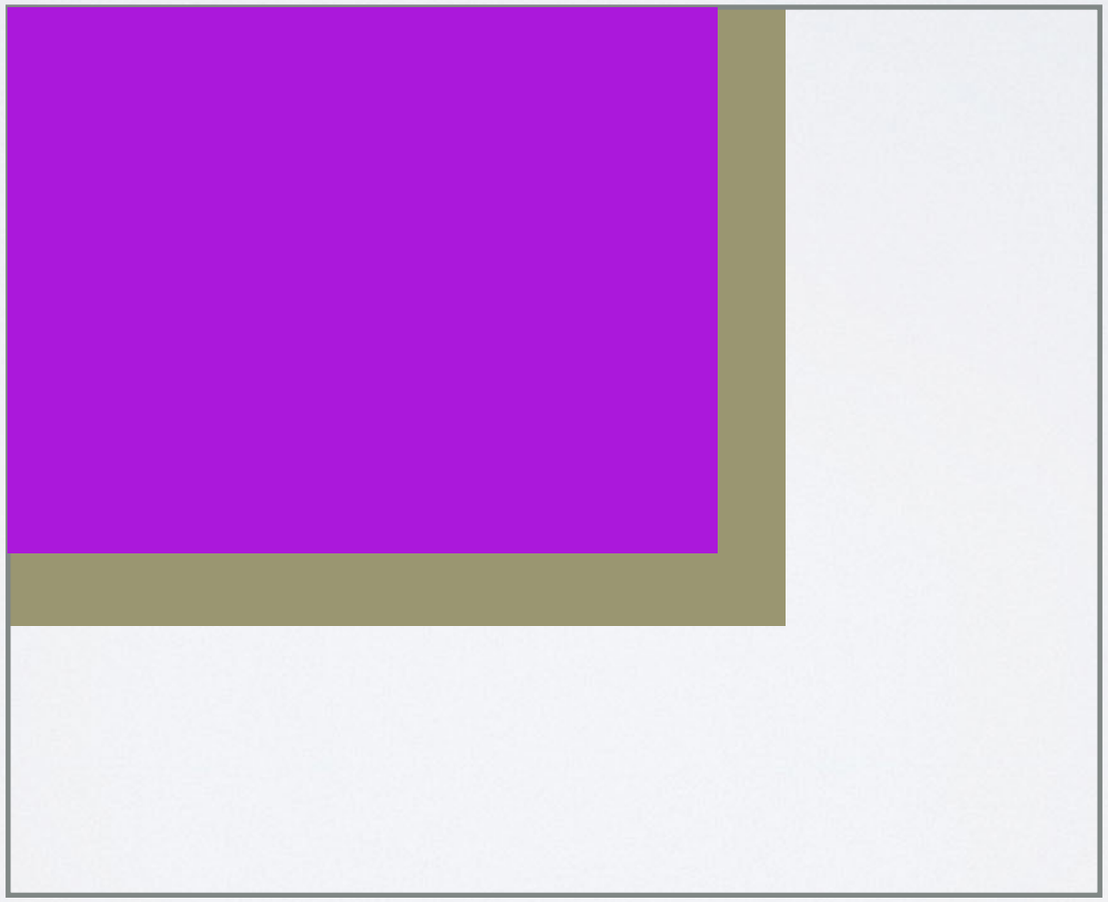
- 如果我們要求目標和

Problem 2

- 如果我們要求目標和

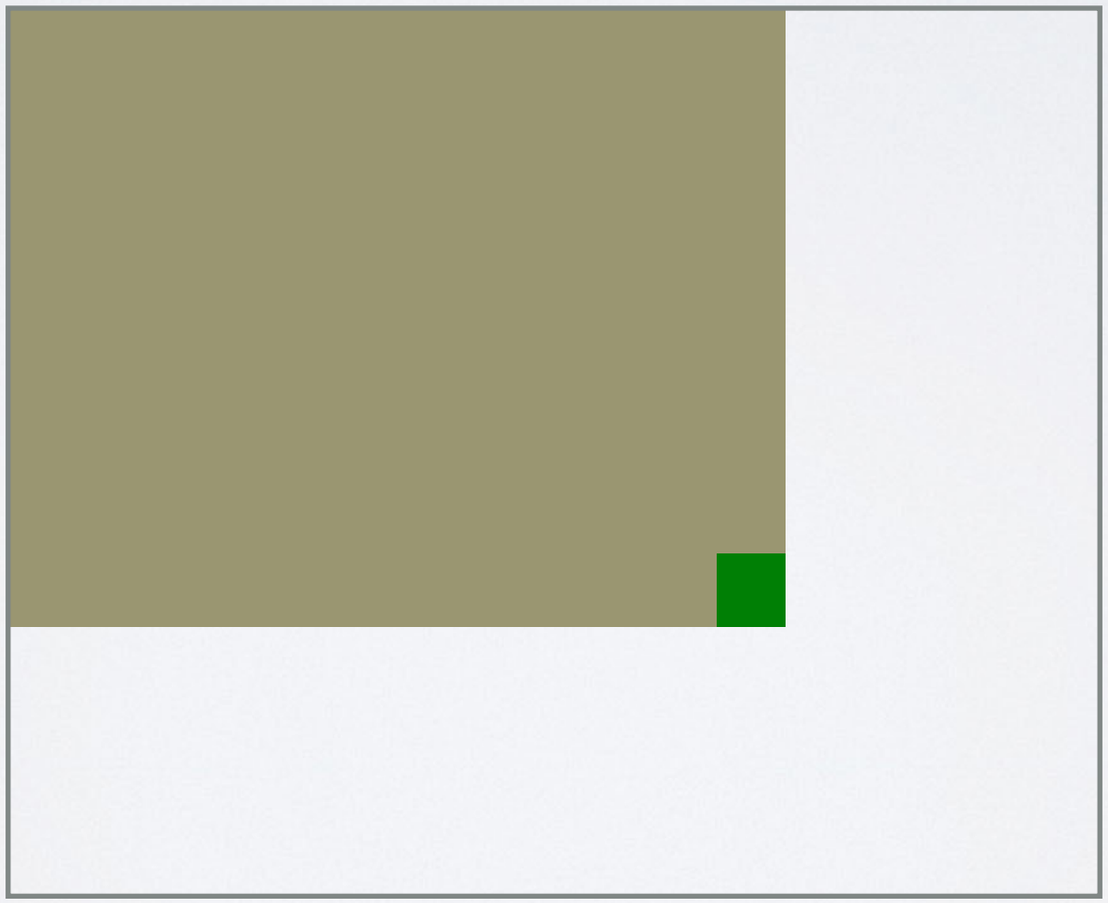
Problem 2

- 如果我們要求目標和



Problem 2

- 如果我們要求目標和

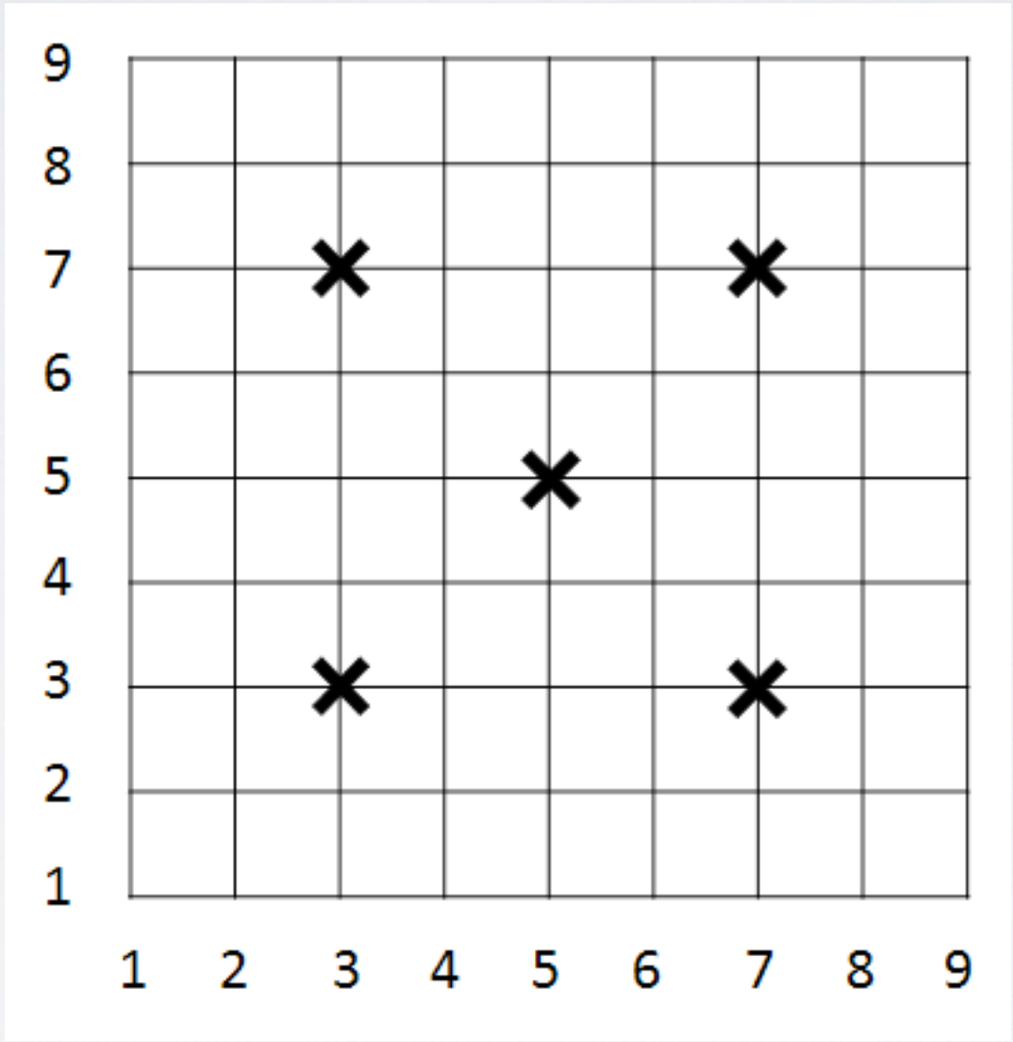


Problem 2 總結

- 剩下讓有興趣的各位回去想想！
- 可以將複雜度壓到 $O(N^3)$
- 考試中有人使用這個做法，相當厲害！

Problem 3

Problem 3



Problem 3

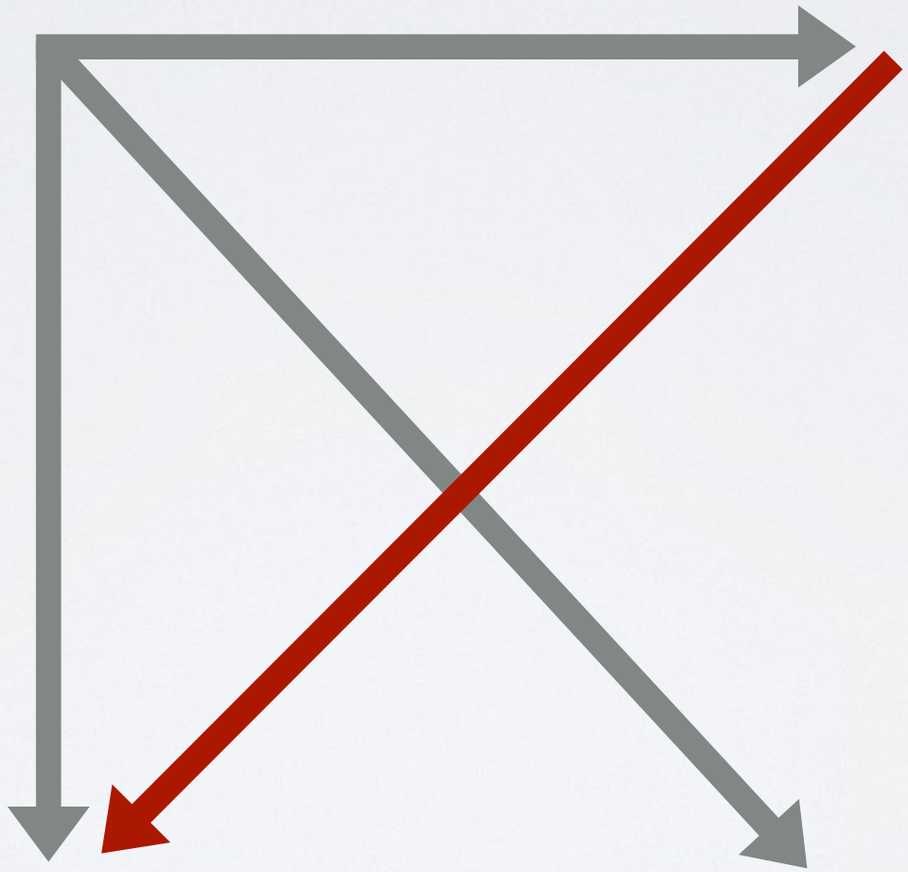
- 連續超過五個棋子怎麼辦？

Problem 3

- 連續超過五個棋子怎麼辦？
- 只要不重複算到，當成獨立的狀況處理即可

Problem 3

- 四種方向五連珠



Problem 3

```
// 0 for X, 1 for Black, 2 for White
for(int i=1; i<=9; i++){
    for(int j=1; j<=5; j++){
        bool black_line = true;
        for(int k=0; k<5; k++){
            if(arr[i][j+k] == 2){
                black_line = false;
                break;
            }
        }
        bool white_line = true;
        for(int k=0; k<5; k++){
            if(arr[i][j+k] == 1){
                white_line = false;
                break;
            }
        }
    }
}
```

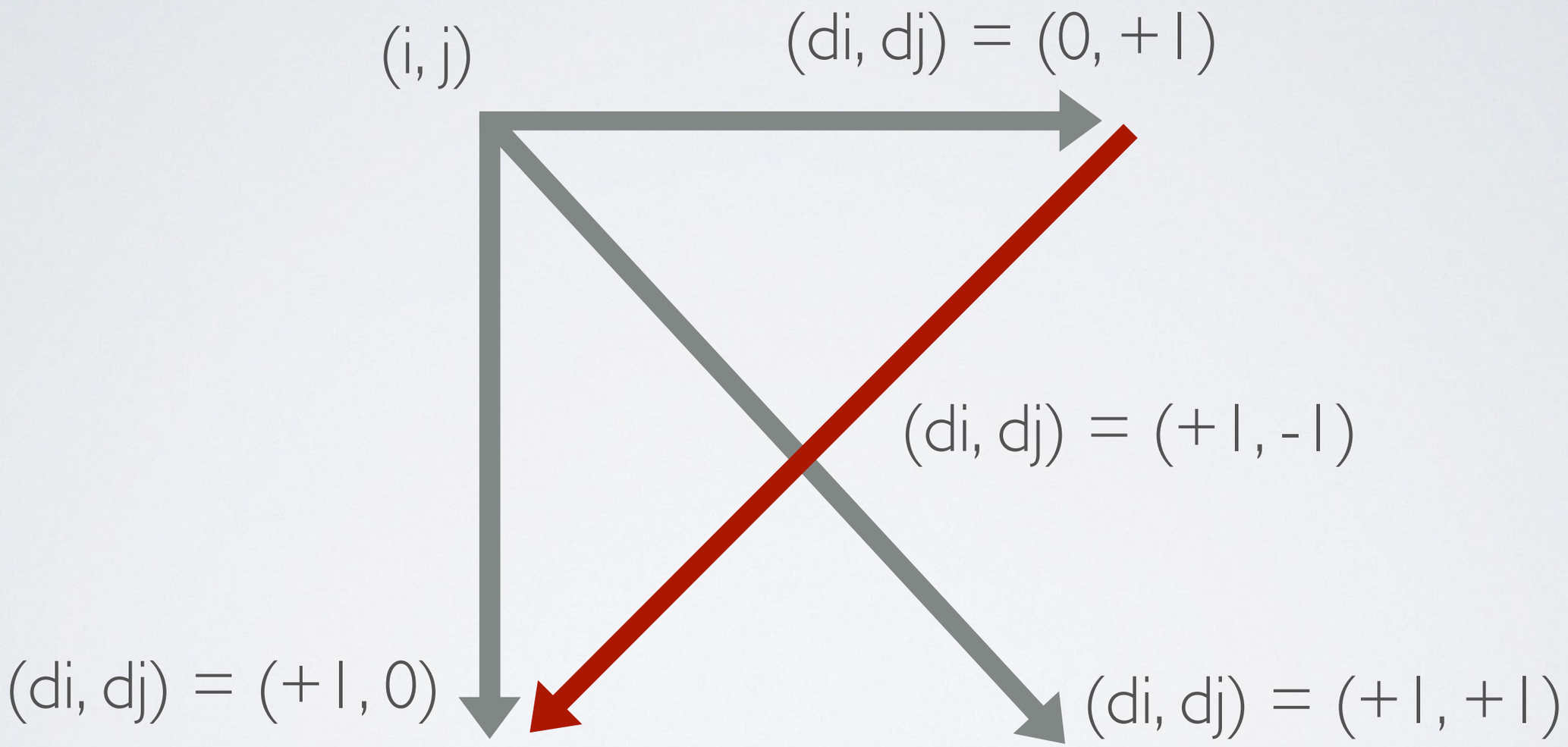
Problem 3

- 照著剛剛的程式碼，分別實作四種狀況即可

Problem 3

- 照著剛剛的程式碼，分別實作四種狀況即可
- 有沒有更漂亮的寫法？

Problem 3



Problem 3

```
// 0 for X, 1 for Black, 2 for White
int di[4] = {1, 1, 1, 0};
int dj[4] = {0, 1,-1, 1};
for(int i=1; i<=9; i++){
    for(int j=1; j<=9; j++){
        for(int c=1; c<=2; c++) { //color
            for(int dir=0; dir<4; dir++) { //direction
                for(int step=0; step<5; step++){
                    // 考慮 i+di[dir]*step, j+dj[dir]*step
                }
            }
        }
    }
}
```

- 請自己加入邊界判斷

Problem 3 總結

- 考試中有人寫了兩百多行程式碼
- 主要運算的程式碼其實不用二十行就夠了！

Problem 4

Problem 4

- N 個物流中心、M 個零售店
- 物流中心流量無限，要滿足零售店需求，並最大化銷貨毛利

Problem 4

- 最大化銷貨毛利 = 找最近的物流中心補貨

Problem 4

- 最大化銷貨毛利 = 找最近的物流中心補貨
- 對於每個零售店，找最近的物流中心

Problem 4

- 最大化銷貨毛利 = 找最近的物流中心補貨
- 對於每個零售店，找最近的物流中心
- 如果 $p\text{-}cd_{ij} >= 0$，則零售店被滿足
- 如果 $p\text{-}cd_{ij} < 0$，則零售店未被滿足

Problem 4 總結

- 照著題目敘述算，並且代入公式就可以了
- 可以分別把「找最近點」跟「算銷貨毛利」寫成兩個函式來做

Q & A