







# SOLUTION CODEWAR 2019 BY TEAM ONLYBOYS

		A (100)	B (100)	C (120)	TOTAL	D (100)	E (120)	F (150)	TOTAL	G (120)	H (200)	I (200)	TOTAL	J (160)	K (120)	L (200)	TOTAL	
		100	100	120	320	100	120	150	370	120	200	200	520	160	120	200	480	1690
		00:09:07 0.47 s 1 try	00:08:46 0.49 s 1 try	00:22:40 0.56 s 1 try	00:40:33 1.52 s 3 tries	00:03:20 0.24 s 1 try	00:47:51 0.62 s 1 try	00:40:31 0.86 s 1 try	01:31:42 1.72 s 3 tries	01:54:29 0.43 s 1 try	01:04:22 0.46 s 1 try	02:05:35 0.43 s 2 tries	05:04:26 1.32 s 4 tries	01:41:29 0.97 s 3 tries	03:49:00 0.3 s 8 tries	04:53:39 0.59 s 2 tries	10:24:08 1.86 s 13 tries	17:40:49 6.42 s 23 tries
		100	100	120	320	100	120	150	370	120	200	200	520	160	120	200	480	1690
		00:02:49 0.52 s 1 try	00:01:12 0.52 s 1 try	00:07:55 0.59 s 1 try	00:11:56 1.63 s 3 tries	00:01:19 0.21 s 1 try	00:03:06 0.5 s 1 try	00:08:09 1.23 s 1 try	00:12:34 1.94 s 3 tries	03:27:42 0.43 s 1 try	00:47:05 0.59 s 1 try	02:02:32 0.55 s 1 try	06:17:19 1.57 s 3 tries	00:28:45 0.36 s 1 try	02:55:19 0.33 s 5 tries	12:01:12 1.04 s 5 tries	15:25:16 1.73 s 11 tries	22:07:05 6.87 s 20 tries
		100	100	120	320	100	120	150	370	120	200	200	520	160	120	200	480	1690
		00:01:54 0.49 s 1 try	00:04:27 0.53 s 1 try	00:19:44 0.55 s 1 try	00:26:05 1.57 s 3 tries	00:11:02 0.2 s 1 try	00:48:41 5.36 s 1 try	00:43:44 0.83 s 1 try	01:43:27 6.39 s 3 tries	05:49:19 0.45 s 9 tries	00:52:51 0.67 s 1 try	27:28:31 0.38 s 8 tries	34:10:41 1.50 s 19 tries	00:27:07 0.39 s 1 try	01:54:23 0.29 s 5 tries	06:57:28 0.47 s 4 tries	09:18:58 1.15 s 10 tries	45:39:11 10.61 s 35 tries

## Thành viên team:

- Trương Công Thành: K18
- Trần Thanh Bình: K18
- Trần Đức Duy: K19

Mọi chi tiết về thắc mắc xin hãy liên hệ với Page

<https://www.facebook.com/groups/BachKhoaAlgorithmsClub/>

Hay mình:

<https://www.facebook.com/congthanh.truong.775>

## Bài A: codewar2019\_ascArrayTransformSteps:

<https://ideone.com/xg6BAO>

### Kiến thức: Tham lam

Vì các phần tử mình chỉ được thêm chứ không bớt nên xét x là giá trị hiện tại.

- Nếu  $arr[i] \geq x + 1$  thì mình gán  $x = arr[i]$ ;
- Nếu  $arr[i] < x + 1$  Tăng kết quả lên 1 và gán  $x = x + 1$  (cho  $arr[i] = x + 1$ )

## Bài B: codewar2019\_nonOverlappingArea:

<https://ideone.com/Jlg9mK>

### Kiến thức: Toán hình học

$$S(A \text{ xor } B) = S(A) + S(B) - 2 * S(AB)$$

Nên đầu tiên mình tính diện tích riêng 2 hình chữ nhật. Sau đó tính diện tích phần chung của 2 hình bằng cách.

- Gọi x là độ dài của đoạn giao  $[A.xL, A.xR]$  với  $[B.xL, B.xR]$

- Gọi  $y$  là độ dài của đoạn giao  $[A.yL, A.yR]$  với  $[B.yL, B.yR]$
- Diện tích phần chung là tích  $x * y$

## **Bài C: codewar2019\_cinemasBuildingCost:**

<https://ideone.com/v7stXZ>

### **Kiến thức: BFS & Sort & Kỹ thuật lập trình**

Với mỗi `citiesList[i]` thì mình cần tách name và trọng số. Đầu tiên với mỗi name thì mình cần lưu trọng số của name đó. Dùng STL map `c++`: `map<string, int>` để quản lý một cách dễ dàng.

Sort các name lại sao cho trọng số của name đó tăng dần. Với mỗi name  $A$  chưa được đánh dấu thì mình tăng kết quả theo trọng số của name  $A$  và đánh dấu tất cả các name khác cùng nằm trong một miền liên thông với  $A$ .

## **Bài D: codewar2019\_encodeString: <https://ideone.com/RHQfaY>**

### **Kiến thức: Duyệt**

Với mỗi kí tự  $c$  trong message thì nếu có tồn tại cặp  $(c, d)$  trong `encodeRules` thì đổi  $c$  thành  $d$  còn không thì giữ nguyên.

## **Bài E: codewar2019\_findTheLargestPrime:**

<https://ideone.com/7VEwIf>

### **Kiến thức: Sinh hoán vị & Sort & Sàng & Kiểm tra nguyên tố**

Đầu tiên sort vector `Digits` giảm dần. Sinh tất cả các hoán vị có thể của `Digits`, dùng hàm `prev_permutation` hay `next_permutation` có sẵn trong C++.

Đầu tiên nếu toàn bộ dãy số là một số nguyên tố return ngay luôn (vì các số sẽ được hoán vị từ số lớn nhất về số nhỏ nhất). Trong quá trình tạo số 1, 12, 123 thì có thể dùng sàng nguyên tố để kiểm tra trong  $O(1)$ .

## **Bài F: codewar2019\_queriesOnRectangle:**

<https://ideone.com/u74Lbj>

### **Kiến thức: Đạo hàm & Prefix Sum 2D**

Mình sẽ nói một bài toán cơ bản của bài này: "Cho một dãy số  $A$  có  $N$  phần tử ban đầu toàn 0, cho  $Q$  truy vấn mỗi truy vấn tăng 1 đoạn  $[L, R]$  lên  $x$  đơn vị"

Cách giải quyết: Gọi  $B(x) = A(x) - A(x - 1)$ . Mảng  $B$  gọi là mảng đạo hàm của  $A$ . Truy vấn  $L, R, x$  thực ra là

- $B(L) += x$
- $B(R + 1) -= x$ ;

Sau đó  $A(0) = B(0)$ , ta tính được  $A(x) = A(x - 1) + B(x)$ , thực chất ta có thể xem như B chính là A luôn.

Áp dụng vào bài toán này, truy vấn update  $x_L$   $y_L$   $x_R$   $y_R$  ta làm như sau:

- Nếu  $x_L = x_R$  thì tăng  $A(x_L, y_L) += 1$ , giảm  $A(x_L, y_R + 1) -= 1$ ;
- Nếu  $x_L \neq x_R$  thì:
  - tăng  $A(x_L, y_L) += 1$
  - tăng  $A(x, 0) += 1$  với  $x_L < x < x_R$
  - tăng  $A(x_R, 0) += 1$ , giảm  $A(x_R, y_R + 1) -= 1$ ;

Rồi  $A(x, y) = A(x, y - 1) + A(x, y)$  là ta đã xây dựng được bảng A trong bài.

Gọi  $F(x, y)$  là tổng trong hình chữ nhật với góc trái trên là  $(0, 0)$ , góc phải dưới là  $(x, y)$ :  $F(x, y) = F(x, y - 1) + F(x - 1, y) - F(x - 1, y - 1) + A(x, y)$ . Dùng  $F(x, y)$  để thực hiện Query loại 2.

## Bài G: isOneMoveCheckMate: <https://ideone.com/gTAm0A>

### Kiến thức: BFS – Loang trên bảng

Đầu tiên ta phải hiểu định nghĩa của đề là các quân trắng hiện tại (sau khi 1 quân trắng di chuyển) phải chiếu được ô Vua đen và Vua đen không thể đi đến ô an toàn nào khác thì mình mới tính là true còn không là false

Đầu tiên mình cho mỗi quân trắng **lần lượt** di chuyển đến một ô nào đấy (lưu ý xe, tượng, hậu không được đi xuyên ô trắng khác). Cứ 1 lần di chuyển của 1 quân trắng ta sẽ có một trạng thái bảng  $8 * 8$  gồm các quân cờ trắng và 1 ô vua Đen. Để việc di chuyển 1 quân cờ dễ dàng mình nên có ma trận như thế này: với các quân lần lượt của mình là K, k, q, r, b, n.

```
const int dx[6][8] = {
    {-1, -1, 0, 1, 1, 1, 0, -1},
    {-1, -1, 0, 1, 1, 1, 0, -1},
    {-1, -1, 0, 1, 1, 1, 0, -1},
    {-1, 0, 1, 0, 0, 0, 0, 0},
    {-1, 1, 1, -1, 0, 0, 0, 0},
    {-2, -1, 1, 2, 2, 1, -1, -2}
```

```
};
```

```
const int dy[6][8] = {
    {0, 1, 1, 1, 0, -1, -1, -1},
    {0, 1, 1, 1, 0, -1, -1, -1},
    {0, 1, 1, 1, 0, -1, -1, -1},
```

```
{0, 1, 0, -1, 0, 0, 0, 0},  
{1, 1, -1, -1, 0, 0, 0, 0},  
{1, 2, 2, 1, -1, -2, -2, -1}  
};
```

Sau khi đã có một bảng trạng thái thì ta làm như sau:

- Với mỗi quân trắng ta sẽ đánh dấu tất cả những ô mà quân trắng có thể di chuyển tới được (Chú ý xe, hậu, tượng không được đi xuyên qua ô chứa quân trắng)
- Kiểm tra ô Vua đen và 8 ô xung quanh vua đen (bảng 3 x 3 mà ô Vua đen là trung tâm) xem nếu tồn tại một ô mà không được đánh dấu thì bảng hiện tại không thể chiếu được Vua đen còn không là chiếu được ra return true ngay liền.

## **Bài H: codewar2019\_treasurePath:** <https://ideone.com/RyTigi>

**Kiến thức: BFS – Loang trên bảng & Chặt nhị phân & Cặp ghép**

Đầu tiên mình BFS từ ô 0 để tính đường đi từ 0 đến tất cả các ô 3. Nếu tồn tại ô 3 không thể đi tới được thì return -1 luôn.

Với mỗi ô 3 mình sẽ BFS để đi đến tất cả các ô 4, với mỗi ô 4 mình phải lưu lại trọng số trên đường đi. Trọng số trên đường đi từ 0 -> 3 -> 4 của mình là trọng số đường đi từ 0 đến 3 + trọng số đường đi từ 3 -> 4.

Chặt nhị phân đáp án: giả sử là H rồi kiểm tra xem H có thỏa không:

- Xây dựng đồ thị 2 phía với tập bên trái là các ô 3, tập bên phải là các ô 4. Nối một cung từ ô 3 sang ô 4 nếu trọng số đường đi 0 -> 3 -> 4 <= H.
- Chạy bộ cặp ghép cực đại, nếu trọng số bộ cặp ghép = số lượng số ô 3 (= số lượng số ô 4) thì H thỏa

## **Bài I: Codewar2019\_findLargestRectangle:**

<https://ideone.com/9ZS8vx>

**Kiến thức: Đệ quy đặt cận – quay lui**

“Theo mình đánh giá đây là bài khó nhất trong 12 bài. Đòi hỏi các bạn phải cực kỳ thành thạo đệ quy đặt cận - quay lui, có rất nhiều bước xử lý buộc bạn phải cực kỳ cẩn thận”

Đầu tiên mình sẽ xét tất cả các bảng  $N * M$  có thể của bảng. Lưu ý:  $N * M$  phải chia hết cho 4 vì số ô của một block luôn là 4 nên khi xếp K thanh thì nó cũng là số chia hết cho 4. Mình ưu tiên  $N * M$  theo thứ tự giảm dần của  $N *$

$M, N * M$  lớn nhất là tổng các `blockNumber[i]` \* 4. Sau khi đã có bảng  $N * M$  thì mình sẽ kiểm tra bảng  $N * M$  có thể được tạo thành từ các thanh không?

Mình dùng mảng đánh dấu  $DD(x, y) = \text{true}$  nếu ô  $DD(x, y)$  đã được đặt và ngược lại. Mình xét cả 7 block, mỗi block sẽ xét tất cả các trạng thái xoay của nó. Với mỗi block cùng với trạng thái xoay của nó mình sẽ tìm ô  $(x, y)$  **đầu tiên** còn trống:  $DD(x, y) = \text{false}$  và xét:

- Nếu có thể đặt vào thì mình đệ quy tiếp. Sau khi quay lui mình phải set lại trạng thái quân  $(x, y)$  và các ô khi mình đặt vào là false.
- Nếu không thì xét các trạng thái xoay khác của block hay xét các block khác
- **Lưu ý:** Khi tìm được ô  $(x, y)$  **đầu tiên** trống thì nếu có thể đặt hay không thể đặt thì mình cũng ngắt vòng lặp để không cần tìm ô  $(x, y)$  tiếp theo.

Nếu như tồn tại cách đặt thỏa bảng  $N * M$  thì mình sẽ return true và không cần phải đệ quy thêm 1 lần nào nữa.

## Bài J: Codewar2019\_connectingPipesCost:

<https://ideone.com/hE4Wj3>

### Kiến thức: Đường đi ngắn nhất (Dijkstra or BFS)

Đầu tiên mình định nghĩa 0 là hướng lên, 1 là bên phải, 2 là xuống, 3 là bên trái. Với 11 ống mình phải có bảng  $KT(x, h1, h2) = \text{true}$  nếu với ống loại  $x$  thì hướng  $h1$  có liên thông được hướng  $h2$  không. Ví dụ  $KT(0, 1, 3) = KT(0, 3, 1) = \text{true}$

Mình sẽ đánh số bảng theo kiểu hàng là từ 1 đến  $N$  và cột là từ 1 đến  $M$ .

Gọi  $D(x, y, h)$  là đường đi ngắn nhất tới ô  $(x, y)$  với đầu vào là từ hướng  $h$ . Ví dụ  $D(2, 2, 0)$  là đường đi ngắn nhất đến ô  $(2, 2)$  với đầu vào là từ hướng lên, tức là từ ô  $(1, 2)$  đi tới ô  $(2, 2)$ . Khởi tạo tất cả các  $D(x, y, h)$  là  $\infty$ , còn riêng  $D(1, 1, 3) = 0$ .

Chạy thuật toán Dijkstra khi lấy 3 giá trị  $(x, y, h)$  ra thì mình duyệt 4 hướng  $h'$  để đến được ô  $(u, v)$  từ  $(x, y)$  rồi duyệt 11 ống, giả sử đang xét ống id lưu ý  $KT(id, h, h')$  phải = true thì cập nhật  $D(u, v, (h' - 2 + 4) \% 4)$  với  $(h' - 2 + 4)$  là hướng vào của ô  $(u, v)$  còn  $h'$  là hướng tới  $(u, v)$  từ ô  $(x, y)$ .

- $D(u, v, (h' - 2 + 4) \% 4) = D(x, y, h) + 1$  nếu  $id \neq a[x][y]$  (Khác loại nên phải xoay lại thành ống id)
- $D(u, v, (h' - 2 + 4) \% 4) = D(x, y, h)$  nếu  $id == a[x][y]$  (Cùng loại)

Kết quả là  $D(N, M + 1, 3)$

## Bài K: Codewar2019\_maximumEarning:

<https://ideone.com/5TUjCT>

### **Kiến thức: Sort & Dynamic Programming & Time**

"Đây là một bài khá lằng nhằng vì cần phải biết kĩ thuật xử lý ngày tháng năm, C++ có cung cấp một hàm cho biết khoảng cách số giây từ một ngày tháng hiện tại đến thời gian chuẩn: 0:00:00 ngày 1/1/1970 và bài này dùng cái này rất nhiều"

<https://stackoverflow.com/questions/37306895/c-converting-a-string-to-a-time-t-variable>

Đầu tiên mình sẽ loại bỏ những công việc không hợp lệ (có làm vào thời gian nghỉ)

- Đầu tiên nếu thời gian làm (tính theo giây)  $\geq 86400$  (thời gian 1 ngày theo giây) thì chắc chắn loại
- Rồi sau đó kiểm tra nếu nó rơi vào những khoảng time trong ngày không hợp lệ thì loại (Lưu ý trường hợp kết thúc vào lúc 0:00:00 ngày tiếp theo)

Với mỗi công việc hợp lệ thì mình sẽ lưu 3 giá trị (L, R, val) với

- L: khoảng cách số giây từ thời gian bắt đầu làm đến ngày chuẩn
- R: Khoảng cách số giây từ thời gian kết thúc đến ngày chuẩn
- Val: là hệ số công việc

Sort các công việc theo R tăng dần, nếu R bằng nhau thì sort L tăng dần.

Gọi  $F(x)$  là tiền công lớn nhất nếu như làm công việc x.

- $F(x) = \max: F(y) + \text{tiền công làm công việc } x$  (điều kiện là R của y  $\leq$  L của x)
- Kết quả của bài toán trả về là  $\max: F(x)$ .

## Bài L: codewar2019\_countStrikeRounds:

<https://ideone.com/gRKkcX>

### **Kiến thức: Toán – Giải hệ phương trình**

Đầu tiên mình phải viết các trạng thái play game có thể của 2 bên xem với trạng thái đó thì Kills và Death thay đổi như thế nào. Cho một trạng thái Kills & Death ban đầu, tìm cách biến đổi để về dạng (0, 0, 0) & (0, 0, 0)

	Kill	Death
V-N	1, 0, 0	0, 0, 1
A-N	0, 1, 0	0, 0, 1
N-A, V-N	1, 0, 1	0, 1, 1
NV, A-N	0, 1, 1	1, 0, 1
N-V, N-A	0, 0, 2	1, 1, 0
V-A, V-N	0, 0, 0	0, 1, 1
V-A, N-N	<del>1, 0, 1</del>	1, 1, 0
A-V, A-N	0, 0, 0	1, 0, 1
A-V, N-A	0, -1, 1	1, 1, 0
Game	$K_1 \ K_2 \ K_3$	$D_1 \ D_2 \ D_3$
$x_1 + x_3 - x_7 = K_1$ $x_2 + x_4 - x_8 = K_2$ $x_3 + x_4 + x_5 + x_7 + x_9 = K_3$ $x_4 + x_5 + x_7 + x_8 + x_9 = D_1$ $x_3 + x_5 + x_6 + x_7 + x_9 = D_2$ $x_1 + x_2 + x_3 + x_4 + x_6 + x_8 = D_3$ $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 = n$		

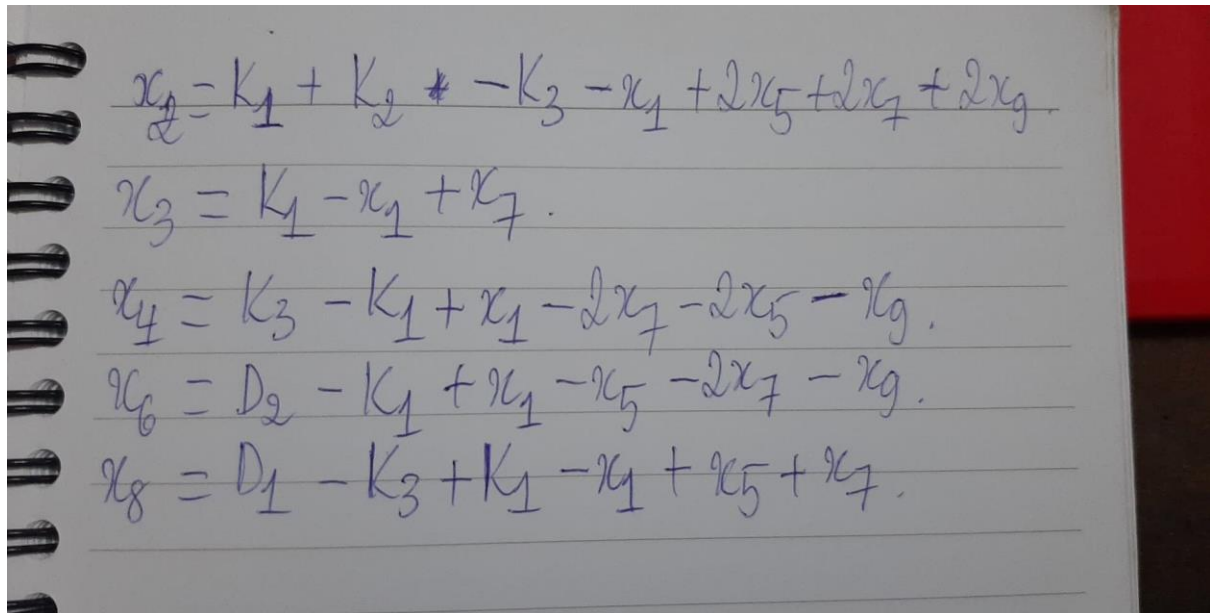
Gọi  $x_1, x_2, \dots, x_9$  là số lần mình play Game theo trạng thái I;  $n$  là số lượt chơi  
 $\Rightarrow x_1 + x_2 + \dots + x_9 = n$ . Bài toán yêu cầu tìm bộ nghiệm  $(x_1, x_2, \dots, x_9)$   
 sao cho  $n$  min và  $n$  max.

Nhìn hình thì mình tưởng đây là một hệ 6 phương trình 9 ẩn nhưng thực chất nó chỉ là hệ 5 phương trình 9 ẩn vì có 1 phương trình phụ thuộc  $K_1 + K_2 - K_3 + D_1 + D_2 - D_3 = 0$  nên càng làm bài toán trở nên khó khăn hơn.

Mình xét bộ 4  $(x_1, x_5, x_7, x_9)$  rồi biến đổi  $(x_2, x_3, x_4, x_6, x_8)$  theo  $(x_1, x_5, x_7, x_9)$  để thành hệ phương trình tuyến tính.



Chú ý:  $n = D_3 + x_5 + x_7 + x_9$  nên  $n$  min hay max phụ thuộc vào  $x_5 + x_7 + x_9$



Handwritten equations on a spiral notebook page:

$$\begin{aligned}x_2 &= K_1 + K_2 - K_3 - x_1 + 2x_5 + 2x_7 + 2x_9 \\x_3 &= K_1 - x_1 + x_7 \\x_4 &= K_3 - K_1 + x_1 - 2x_7 - 2x_5 - x_9 \\x_6 &= D_2 - K_1 + x_1 - x_5 - 2x_7 - x_9 \\x_8 &= D_1 - K_3 + K_1 - x_1 + x_5 + x_7\end{aligned}$$

Thuật toán **Vét**: For hết tất cả bộ 3 ( $x_5, x_7, x_9$ ) rồi với từng phương trình  $x_2, x_3, x_4, x_6, x_8$  mình sẽ tìm điều kiện ràng buộc  $x_1$  để tất cả ( $x_2, x_3, x_4, x_6, x_8$ )  $\geq 0$ . Chỉ cần  $x_1 \geq 0$  là bộ 3 ( $x_5, x_7, x_9$ ) thỏa.

Thuật toán **Full**: For hết tất cả bộ 2 ( $x_5, x_7$ ):

- Từ  $x_3$  và  $x_8 \Rightarrow$  ta biết  $x_1$  sẽ thuộc đoạn  $[0, \max\_x1]$  để  $x_3, x_8 \geq 0$
- Từ  $x_2, x_4, x_6$  &  $x_1$  thuộc đoạn  $[0, \max\_x1]$  mình cần tìm  $x_9$  thuộc đoạn  $[L, R]$  để  $x_2, x_4, x_6 \geq 0$
- Vậy  $x_9 \min = L, x_9 \max = R$ .

---

Cảm ơn các bạn đã theo dõi!