

---

ESCOM - IPN

# Problemas 1: Divide & Conquer

ANÁLISIS DE ALGORITMOS 3CM3



Oscar Andrés Rosas Hernandez

Mayo 2018

# Índice

<b>1. Divide-and-conquer-1: Maximum Subarray Sum</b>	<b>2</b>
1.1. Código de Refencia . . . . .	2
1.2. Accepted . . . . .	3
1.3. Idea del Algoritmo . . . . .	4
<b>2. Amigos y Regalos</b>	<b>5</b>
2.1. Código de Refencia . . . . .	5
2.2. Accepted . . . . .	6
2.3. Idea del Algoritmo . . . . .	7
<b>3. Cúmulo - Closest Pair of Points</b>	<b>8</b>
3.1. Código de Refencia . . . . .	8
3.2. Accepted . . . . .	10
3.3. Idea del Algoritmo . . . . .	11
<b>4. Inversion Count</b>	<b>12</b>
4.1. Código de Refencia . . . . .	12
4.2. Accepted . . . . .	14
4.3. Idea del Algoritmo . . . . .	15

# 1. Divide-and-conquer-1: Maximum Subarray Sum

## 1.1. Código de Refencia

```
1  /*
2   ===== LARGEST SUM CONTIGOUS SUBARRAY =====
3   =====
4   #include <iostream>                                // Include Libraries
5   #include <vector>                                 // Include Libraries
6   using namespace std;                             // Bad practice
7
8
9  /*
10  ===== MAX SUBARRAY SUM =====
11  =====
12  template <class T>                                // Templates for all
13  long long int MaxSubArraySum(vector<T> &Data) {    // Function
14
15      T MaximumContiguosSum = Data[0];                // Local vars
16      T CurrentMaxSum = Data[0];                      // Local vars
17
18      for (int i = 1; i < Data.size(); ++i) {          // For each element FROM 1
19          if (Data[i] > CurrentMaxSum + Data[i])        // What is better
20              CurrentMaxSum = Data[i];                  // just take current
21          else CurrentMaxSum += Data[i];                // or add it to subarray
22
23          if (CurrentMaxSum > MaximumContiguosSum)       // If we fin new maximum
24              MaximumContiguosSum = CurrentMaxSum;        // Add it !
25      }
26
27      return MaximumContiguosSum;                      // Return it !
28 }
```

## 1.2. Accepted

Nota que con el algoritmo actual llegamos al mejor tiempo de todos

The screenshot shows the omegaUp platform interface for the "Divide and conquer 1" problem. The problem details include:

- Points:** 16.37
- Memory limit:** 32MB
- Time limit (case):** 1s
- Time limit (total):** 60s

**Description:** Edgardo se puso un poco intenso este semestre y puso a trabajar a sus alumnos con problemas de mayor dificultad. La tarea es simple, dado un arreglo de números enteros debes imprimir cual es la suma máxima en cualquier subarreglo contiguo. Por ejemplo si el arreglo dado es {-2, -5, 6, -2, -3, 1, 5, -6}, entonces la suma máxima en un subarreglo contiguo es 7.

**Entrada:** La primera linea contendrá un numero. En la siguiente linea enteros representando el arreglo.

**Salida:** La suma máxima en cualquier subarreglo contiguo.

**Ejemplo:**

Entrada	Salida
<pre>8 -2 -5 6 -2 -3 1 5 -6 . .</pre>	7
<pre>5 1 2 3 4 5 . .</pre>	15

**Límites:**

- 

Source: Filiberto Fuentes  
Uploaded by: galloska  
Report inappropriate content in this problem.  
Rate problem

**Submissions:**

Submitted	GUID	Status	Percentage	Language	Memory	Runtime	Details
2016-05-12 17:10:31	a8133127d	Accepted	100.00%	cpp11	2.10 MB	0.02 s	<a href="#">View</a>
2016-05-12 17:09:18	cdf450a3e	Accepted	100.00%	cpp11	2.11 MB	0.02 s	<a href="#">View</a>
2016-05-12 17:07:43	10b5f5b71	Accepted	100.00%	cpp11	2.11 MB	0.02 s	<a href="#">View</a>
2016-05-12 17:04:53	52208d8de	Accepted	100.00%	cpp11	2.12 MB	0.02 s	<a href="#">View</a>
2016-05-12 17:03:35	1a1c59b6	Accepted	100.00%	cpp11	2.16 MB	0.02 s	<a href="#">View</a>
2016-05-12 16:40:04	ab7e712ae	Accepted	100.00%	cpp11	2.11 MB	0.01 s	<a href="#">View</a>
2016-05-12 16:38:36	0b4c0229	Compilation error	0.00%	—	—	—	<a href="#">View</a>
2016-05-10 22:26:48	e7e6f71f	Accepted	100.00%	cpp11	3.55 MB	0.08 s	<a href="#">View</a>
2016-05-10 22:23:48	170e6eb8	Compilation error	0.00%	—	—	—	<a href="#">View</a>
2016-05-10 22:21:15	340cf686	Accepted	100.00%	cpp11	3.88 MB	0.07 s	<a href="#">View</a>
2016-05-10 22:19:29	4945298e	Partially accepted	58.33%	cpp11	3.55 MB	0.06 s	<a href="#">View</a>
2016-05-10 21:54:53	b3d636d7	Accepted	100.00%	cpp11	3.51 MB	0.07 s	<a href="#">View</a>
2016-05-10 21:53:40	1e38ac88	Accepted	100.00%	cpp11	3.94 MB	0.08 s	<a href="#">View</a>
2016-05-10 21:51:52	b4455d47	Accepted	100.00%	cpp11	3.93 MB	0.07 s	<a href="#">View</a>
2016-05-10 21:49:02	c8a837db	Accepted	100.00%	cpp11	3.52 MB	0.06 s	<a href="#">View</a>
2016-05-10 21:47:11	4e0303b14	Accepted	100.00%	cpp11	3.58 MB	0.06 s	<a href="#">View</a>
2016-05-10 21:44:48	a36eccc50	Partially accepted	41.67%	cpp11	3.61 MB	0.06 s	<a href="#">View</a>
2016-05-10 21:43:34	d41f5c5fb	Partially accepted	41.67%	cpp11	3.56 MB	0.06 s	<a href="#">View</a>
2016-05-10 21:41:48	d5cb1a48	Partially accepted	50.00%	cpp11	3.52 MB	0.06 s	<a href="#">View</a>
2016-05-10 21:36:20	07937e99	Compilation error	0.00%	—	—	—	<a href="#">View</a>
2016-05-10 21:35:01	4fd23c5f8	Partially accepted	50.00%	cpp11	3.58 MB	0.31 s	<a href="#">View</a>

New submission

**Best accepted solvers:**

User	Language	Memory	Runtime	Submitted
SoyOscarRH	cpp11	2.11	0.01	2016-05-12 16:40:05
BetoSCL	cpp11	3.91	0.02	2016-05-12 23:06:07
acostas413	cpp	3.05	0.03	2017-05-29 00:23:51
Marcos	cpp11	4.35	0.06	2017-05-29 00:42:46
ianoroso	c	3.24	0.06	2017-10-23 09:25:59
cruzjorgesalazar	cpp11	3.17	0.06	2018-05-02 10:02:02
bogv	cpp	3.53	0.07	2017-05-29 22:25:21
Diego_Briares	cpp11	4.35	0.07	2017-08-28 01:00:24
JFCowboy	cpp11	3.61	0.08	2017-05-28 00:44:24
josemanuelaraf	cpp	3.63	0.08	2017-05-26 14:51:35

### 1.3. Idea del Algoritmo

#### ■ Forma Estúpida:

El método estupido es ejecutar dos bucles.

El bucle externo selecciona el elemento inicial, el bucle interno encuentra la suma máxima posible con el primer elemento elegido por el bucle externo y compara este máximo con el máximo general.

Finalmente devuelve el máximo general. Esta cosa corre en  $O(n^2)$

#### ■ Forma Divide and Conquer:

Usando el enfoque Divide and Conquer, podemos encontrar la suma máxima en  $O(n \log n)$ .

La idea sería:

1. Divida el arreglo dada en dos mitades
2. Devuelve el máximo de los siguientes tres
  - Suma del subarreglo máxima en la mitad izquierda (Hacer una llamada recursiva)
  - Suma del subarreglo máxima en la mitad derecha (Hacer una llamada recursiva)
  - Máxima suma del subarreglo tal que el subarreglo cruza el punto medio

#### ■ Forma Greedy

Lo que tenemos que hacer es darnos cuenta que a cada paso de recorrer el arreglo tenemos 2 opciones: ¿Qué es mejor (en cuanto a la mayor suma)? El arreglo que ya llevó más el elemento actual o solo el elemento actual.

Resulta trivial demostrar que este algoritmo funciona.

Y lo mejor, corre en  $O(n)$

## 2. Amigos y Regalos

### 2.1. Código de Refencia

```
1  /*
2   =====
3   AMIGOS Y REGALOS
4   =====
5   https://omegaup.com/arena/problem/Amigos-y-Regalos#problems */
6
7 #include <iostream>                                //Include Libraries
8 #include <vector>                                 //Include Libraries
9 using namespace std;                             //Bad practice
10
11 typedef long long lli;                           //Just a so long name, sorry
12
13 lli NumberOfIntsFromFriend1, NumberOfIntsFromFriend2; //Global Vars
14 lli BadPrimesFor1, BadPrimesFor2;                //Global Vars
15
16 bool Valid(lli n) {                                //Is n a valid sequence?
17     lli NotValid = n / (BadPrimesFor1 * BadPrimesFor2); //Not valid intergers
18     lli Remove1 = (n / BadPrimesFor1) - NotValid;      //Remove places
19     lli Remove2 = (n / BadPrimesFor2) - NotValid;      //Remove places
20
21     lli Missing1 = max(NumberOfIntsFromFriend1 - Remove2, (lli) 0); //Missing numbers
22     lli Missing2 = max(NumberOfIntsFromFriend2 - Remove1, (lli) 0); //Missing numbers
23
24     return (n - Remove1 - Remove2 - NotValid >= Missing1 + Missing2); //Is n enough?
25 }
26
27 lli BinarySearch (lli Initial, lli Final) {        //Binary Search
28     lli Middle = Initial + (Final - Initial) / 2; //Find middle
29
30     if (Valid(Middle)) {                            //If we found the one
31         if (not Valid(Middle - 1)) return Middle; //Return it!
32         return BinarySearch(Initial, Middle - 1); //If not, find previous one
33     }
34     else return BinarySearch(Middle + 1, Final); //Find binary search
35 }
36
37
38 int main() {                                       //Main
39
40     cin >> NumberOfIntsFromFriend1 >> NumberOfIntsFromFriend2; //Read
41     cin >> BadPrimesFor1 >> BadPrimesFor2;                  //Read
42
43     cout << BinarySearch(
44         NumberOfIntsFromFriend1 + NumberOfIntsFromFriend2,
45         9223372036854775790) << "\n";
46
47     return 0;
48 }
```

## 2.2. Accepted

**omegaUp** Arena Contests Problems Rank Schools Blog Questions  SoyOscarRH+

### Amigos y Regalos

Points	24.47	Memory limit	32MB
Time limit (case)	1s	Time limit (total)	60s

**Descripción**

Tienes dos amigos. A ambos quieren regalarte varios números enteros como obsequio. A tu primer amigo quieres regalarle  $C_1$  enteros y a tu segundo amigo quieres regalarle  $C_2$  enteros. No satisfecho con eso, también quires que todos los regalos sean únicos, lo cual implica que no podrás regalar el mismo entero a ambos de tus amigos.

Además de eso, a tu primer amigo no le gustan los enteros que son divisibles por el número primo  $X$ . A tu segundo amigo no le gustan los enteros que son divisibles por el número primo  $Y$ . Por supuesto, tú no le regalarias a tus amigos números que no les gusten.

Tu objetivo es encontrar el mínimo número  $V$ , de tal modo que puedas dar los regalos a tus amigos utilizando únicamente enteros del conjunto  $1, 2, 3, \dots, V$ . Por supuesto, tú podrías decidir no regalar algunos enteros de ese conjunto.

Un número entero positivo mayor a 1 es llamado primo si no tiene divisores enteros positivos además del 1 y el mismo.

**Entrada**

Una linea que contiene cuatro enteros positivos  $C_1, C_2, X, Y$ . Se garantiza que  $X$  y  $Y$  son números primos.

**Salida**

Una linea. Un entero que representa la respuesta al problema.

**Ejemplo**

Entrada	Salida	Descripción
3 1 2 3 5	5	Teniendo el conjunto de números: {1, 2, 3, 4, 5}, podemos regalarle los enteros {1, 3, 5} al amigo 1, y los enteros {2, 4} al amigo 2. Éste es el conjunto más pequeño que cumple con la solución.
1 1 2 3 2	2	Teniendo el conjunto de números: {1, 2}, podemos regalarle el entero {1} al amigo 1, y el entero {2} al amigo 2. Éste es el conjunto más pequeño que cumple con la solución.

**Límites**

- $1 \leq C_1, C_2 < 10^9$
- $2 \leq X < Y \leq 3 * 10^4$

**Subtareas**

- Para un subconjunto de casos con el valor del 30% de los puntos,  $C_1 + C_2 \leq 10^3$
- Para un subconjunto de casos con el valor del 30% de los puntos,  $C_1 + C_2 \leq 10^6$
- Para el resto de los casos,  $C_1 + C_2 \leq 10^9$

Source: Luis Martín Jiménez Rodríguez (CIOmPs)  
Uploaded by: Luis Martín Jiménez Rodríguez  
Report inappropriate content in this problem.

**Submissions**

Submitted	GUID	Status	Percentage	Language	Memory	Runtime	Details
2018-05-14 22:08:21	5d5f5e8b3	Accepted	100.00%	cpp11	3.12 MB	<0.1 s	
2018-05-14 22:00:43	426e590ed	Accepted	100.00%	cpp11	3.12 MB	0.01 s	
2018-05-14 21:59:14	98c5c523b	Partially accepted	70.00%	cpp11	3.06 MB	0.00 s	
2018-05-14 21:58:13	0f05c5514	Compilation error	—	—	—	—	
2018-05-14 21:52:34	2e2bb252a	Accepted	100.00%	cpp11	3.13 MB	0.00 s	
2018-05-14 21:50:34	f2faea1a	Accepted	100.00%	cpp11	3.13 MB	<0.1 s	
2018-05-14 21:47:03	f3f55578	Wrong Answer	0.00%	cpp11	3.16 MB	>0.0 s	
2018-05-14 21:18:40	b718368e	Accepted	100.00%	cpp11	3.12 MB	0.00 s	
2018-05-14 20:45:05	j8929c91	Time limit exceeded	0.00%	cpp11	3.10 MB	>6.95 s	
2018-05-14 19:52:33	7c2138e4d	Time limit exceeded	0.00%	cpp11	3.16 MB	>1.01 s	
2018-05-14 19:42:54	838ee096	Partially accepted	10.00%	cpp11	3.16 MB	0.00 s	
2018-05-12 16:03:34	c4b9c0cf	Compilation error	—	—	—	—	
2018-05-11 09:10:26	ae05c64b	Time limit exceeded	0.00%	cpp11	3.16 MB	>6.00 s	

New submission

**Best accepted solvers**

User	Language	Memory	Runtime	Submitted
diegogr	cpp11	3.21	0.00	2018-04-19 23:30:49
Mendoza-Bernal, Marco-Jair	cpp11	3.10	0.00	2018-05-14 09:40:44
luischonps	cpp	3.16	0.00	2018-05-14 21:18:58
SoyOscarRH	cpp11	3.12	0.00	2018-04-14 09:16:58
Ozymandias	cpp11	3.11	0.00	2018-04-15 14:35:56
ernestovsios	cpp11	3.15	0.00	2018-04-21 11:35:55
omi_2015_luis.060502	cpp11	3.16	0.00	2018-04-28 10:01:24
ivanalejandro2002	cpp	3.12	0.00	2018-04-28 12:21:26
omi_2015_j.a.e.b.1999a	cpp	3.09	0.00	2018-04-28 12:24:01

### 2.3. Idea del Algoritmo

- **Forma Estúpida:** Podemos tener un contador e ir incrementandolo, cada que encontremos un número que sea divisible por uno pero no por otro entonces reducimos en 1 la cantidad de números en el otro amigo, y así hasta que ambos tengan 0.

- **Forma Divide and Conquer:**

Sea  $f(n)$  una función booleana que diga si podemos armar la respuesta usando el conjunto  $\{ 1, 2, \dots, n \}$ . Vemos que a partir de un punto  $n = n_0$ ,  $f(n)$  comienza a ser siempre verdadera, pues si podemos armar la respuesta con el conjunto  $\{ 1, 2, 3, \dots, n \}$  también se podrá con  $\{ 1, 2, 3, \dots, n, n + 1 \}$ .

Ahora ve que calcular a  $f(n)$  es simplemente simular la forma estúpida, esto se logra con el principio de inclusión -exclusión.

Esto se logran con  $O(\log(n))$

Y si quieres saber porque obtenemos el logaritmo es porque  $f(n)$  se puede obtener en tiempo constante, el logaritmo tiene porque tenemos que hacer una búsqueda binaria para encontrar el  $n$  tal que  $f(n)$  sea verdadera  $f(n - 1)$  sea falso.

## 3. Cúmulo - Closest Pair of Points

### 3.1. Código de Refencia

```
1  /*
2   ===== CLOSEST PAIR OF POINTS =====
3   */
4 #include <cstdlib> // Include Libraries
5 #include <cmath> // Include Libraries
6 #include <algorithm>
7 #include <limits>
8
9 using namespace std; // Bad practice
10
11
12 /**
13  ===== POINT STRUCT =====
14  */
15 struct Point { // A real struct
16     double x, y; // Get the data
17 };
18
19 /**
20  ===== HELPER FUNCTIONS =====
21  */
22 inline double Distance(Point &P1, Point &P2) { // Distance
23     return (P1.x - P2.x) * (P1.x - P2.x) + (P1.y - P2.y)*(P1.y - P2.y); // Calculus
24 }
25
26 double BruteForce(Point Points[], size_t Size) { // Brute Force
27     double MinimunDistance = numeric_limits<double>::max(); // Get a cool max
28
29     for (auto i = 0; i < Size; ++i) { // Foreach point
30         for (auto j = i + 1; j < Size; ++j) { // Foreach point
31             double CurrentDistance = Distance(Points[i], Points[j]); // Get the distance
32             if (CurrentDistance < MinimunDistance) // If better
33                 MinimunDistance = CurrentDistance; // Add it!
34         }
35     }
36
37     return MinimunDistance; // Now return
38 }
39
40
41 /**
42  ===== REAL FUNCTIONS =====
43  */
44 double StripClosest(Point Strip[], size_t Size, double d) { // Start the Distance
45     double MinimunDistance = d; // First Sort the
46
47     std::sort(Strip, Strip + Size, // point about the
48               [](const Point& P1, const Point& P2) { // x - axis
49                   return P1.y < P2.y;
50               });
51
52     for (int i = 0; i < Size; ++i) { // For each point
53         for (int j = i + 1; j < Size; ++j) { // For each point
54             if ((Strip[j].y - Strip[i].y) >= MinimunDistance) break; // If not good, bye
55             double CurrentDistance = Distance(Strip[i], Strip[j]); // Get the distance
56             if (CurrentDistance < MinimunDistance) // If better
57                 MinimunDistance = CurrentDistance; // Add it!
58         }
59     }
60
61     return MinimunDistance;
62 }
```

```

1  double ClosesPoints(Point Points[], size_t NumberOfPoints) {           // Recursice function
2
3      if (NumberOfPoints < 80) return BruteForce(Points, NumberOfPoints);   // Simple case
4
5      size_t Middle = NumberOfPoints / 2;                                //Find the middle place
6
7      double MinDistanceSide = min(                                         //Smallest distance
8          ClosesPoints(Points, Middle),                                     //Left size
9          ClosesPoints(Points + Middle, NumberOfPoints - Middle));        //Right size
10
11     );
12
13     Point Strip[NumberOfPoints];                                         //Line at middle point
14     int SizeOfStrip = 0;                                                 //The Real Size
15
16     for (int i = 0; i < NumberOfPoints; i++) {                           //For each point
17         if (abs(Points[i].x - Points[Middle].x) < MinDistanceSide)    //Are in the radius
18             Strip[SizeOfStrip++] = Points[i];                            //Add it!
19     }
20
21     return min(                                                       //Return the minimum
22         MinDistanceSide,                                              //Of the sides
23         StripClosest(Strip, SizeOfStrip, MinDistanceSide));            //And the distance
24     );
25 }
26
27
28
29 double ClosestPairOfPoints(Point Points[], int NumberOfPoints) {           //Init Funcion
30
31     std::sort(Points, Points + NumberOfPoints,                         //First Sort the
32     [](const Point& P1, const Point& P2) {                           //point about the
33         return P1.x < P2.x;                                            //x - axis
34     });
35
36     return ClosesPoints(Points, NumberOfPoints);                         //Recursice Function
37 }
38
39
40
41 /**
42 *----- MAIN -----*/
43 /**
44 *----- -----*/
45 int main() {                                                               //General Main
46
47     int NumberOfPoints; scanf("%d", &NumberOfPoints);                      //Now reserve space
48     Point Points[NumberOfPoints];
49
50     for (int i = 0; i < NumberOfPoints; ++i) {                            //Now get the data
51         scanf("%f %f", &Points[i].x, &Points[i].y);
52     }
53
54     printf("%.3lf\n", sqrt(ClosestPairOfPoints(Points, NumberOfPoints)));
55
56     return 0;
57 }
```

## 3.2. Accepted

**Cumulo**

Points	20.18	Memory limit	32MB
Time limit (case)	1s	Time limit (total)	60s

**Entrada**  
La primera linea tendrá un entero que indica la cantidad de estrellas en el mapa. Las siguientes n líneas tendrán las coordenadas de las estrellas, dadas por dos reales X y Y. En todos los casos, .

**Salida**  
La distancia mínima entre los estrellas, expresada con un número real con tres cifras después del punto decimal. (La distancia se calcula como la raíz cuadrada de la suma de los cuadrados de las diferencias en X y Y)

**Ejemplos**

Entrada	Salida	Descripción
3 0.0 0.0 2.0 0.0 3.0 3.0 .0	2.000	Las estrellas más cercanas son las primeras dos.
5 10.8 142.5 20.2 14.85 112.4 6.2 11.1 17.65 0.0 512.3 .9	5.000	Las estrellas más cercanas son la segunda y la cuarta.

**Límites**  
• (X, Y reales)

Source: Alain\_Acevedo  
Uploaded by: Alain\_Acevedo  
Report inappropriate content in this problem.  
Rate problem

**Submissions**

Submitted	GUID	Status	Percentage	Language	Memory	Runtime	Details
2013-09-12 16:20:19	cbfc62af	Accepted	100.00%	cpp11	2.61 MB	0.78 s	<a href="#">@</a>
2013-09-12 16:18:13	8eaead6b	Accepted	100.00%	cpp11	2.65 MB	0.82 s	<a href="#">@</a>
2013-09-12 16:16:53	f11e35d8	Accepted	100.00%	cpp11	2.58 MB	0.71 s	<a href="#">@</a>
2013-09-12 16:14:32	34de6874	Compilation error	0.00%	—	—	—	<a href="#">@</a>
2013-09-12 16:09:07	a5c6c93a	Accepted	100.00%	cpp	4.02 MB	0.75 s	<a href="#">@</a>
2013-09-12 16:07:35	b6d1bbef	Compilation error	0.00%	—	—	—	<a href="#">@</a>
2013-09-12 16:05:47	9dc474fe	Accepted	100.00%	cpp	3.57 MB	0.69 s	<a href="#">@</a>
2013-09-12 16:03:39	bb6d6eae	Compilation error	0.00%	—	—	—	<a href="#">@</a>
2013-09-12 15:58:53	051e7334	Accepted	100.00%	cpp11	4.18 MB	1.02 s	<a href="#">@</a>
2013-09-12 15:56:39	2cad0d39	Accepted	100.00%	cpp11	4.09 MB	1.52 s	<a href="#">@</a>
2013-09-12 15:55:28	ebe5f134	Accepted	100.00%	cpp11	4.07 MB	0.98 s	<a href="#">@</a>
2013-09-12 15:54:04	5528ed6b	Accepted	100.00%	cpp11	3.98 MB	1.08 s	<a href="#">@</a>
2013-09-12 15:53:00	f0098a49	Accepted	100.00%	cpp11	4.09 MB	1.04 s	<a href="#">@</a>
2013-09-12 15:51:53	051e0382	Compilation error	0.00%	cpp11	—	—	<a href="#">@</a>
2013-09-12 15:50:53	050ab767	Wrong Answer	0.00%	cpp11	4.14 MB	0.81 s	<a href="#">@</a>
2013-09-12 15:48:52	a04e6d42	Accepted	100.00%	cpp11	4.04 MB	1.13 s	<a href="#">@</a>
2013-09-12 15:47:51	93a6dd33	Wrong Answer	0.00%	cpp11	4.08 MB	0.82 s	<a href="#">@</a>
2013-09-12 15:45:49	eef6027d	Wrong Answer	0.00%	cpp11	4.11 MB	0.60 s	<a href="#">@</a>
2013-09-12 15:44:38	88b95056	Wrong Answer	0.00%	cpp11	4.11 MB	0.82 s	<a href="#">@</a>
2013-09-12 15:43:20	7e835576	Wrong Answer	0.00%	cpp11	4.08 MB	0.88 s	<a href="#">@</a>
2013-09-12 15:35:13	f33eac24	Accepted	100.00%	cpp11	4.07 MB	1.06 s	<a href="#">@</a>
2013-09-12 15:33:17	aeb1ed41	Accepted	100.00%	cpp11	4.05 MB	1.89 s	<a href="#">@</a>
2013-09-12 15:31:52	c493a1a3b	Accepted	100.00%	cpp11	4.07 MB	1.73 s	<a href="#">@</a>
2013-09-12 15:30:51	54d1d58f	Accepted	100.00%	cpp11	3.97 MB	1.88 s	<a href="#">@</a>
2013-09-12 15:29:23	cd8b424f	Partially accepted	0.44%	cpp11	3.69 MB	1.87 s	<a href="#">@</a>
2013-09-12 15:27:24	bb7e834d	Compilation error	0.00%	—	—	—	<a href="#">@</a>

New submission

**Best accepted solvers**

User	Language	Memory	Runtime	Submitted
troyodi	cpp	12.11	0.28	2013-09-10 21:39:09
Juan_Perez	cpp	12.11	0.30	2013-11-20 05:56:03
DianaMendez	cpp	12.11	0.31	2013-11-28 18:46:22
Daniel Jeronimo Gomez Antonio	cpp	12.95	0.31	2013-09-10 23:42:49
gato.edgar21	cpp	12.95	0.31	2013-09-10 23:42:50
fernando.silva	cpp	14.39	0.36	2013-09-11 04:45:22
dialamas24	cpp	14.39	0.39	2013-09-09 02:26:21
charlyhms	cpp	13.63	0.41	2013-09-09 01:45:00
DerrickCastillo	cpp	13.73	0.46	2013-09-07 17:41:25
Emmanuel_Antonio	cpp	12.86	0.49	2013-09-10 02:35:14

### 3.3. Idea del Algoritmo

- **Forma Estúpida:** La forma estúpida, pero muy bonita es tomar cada punto y sacar la distancia con todos los demás puntos y guardar la distancia más pequeña, hacer lo mismo para cada punto y así obtener la distancia menor.

Este algoritmo hace operaciones constantes para cada punto para cada punto, es decir  $O(n^2)$

- **Forma Divide and Conquer:**

Ahora, para ahorrarnos pasamos a hacer una idea divide y vencerás, primero, vas a tomar tus puntos y vas a ordenarlos con respecto a  $x$ , ahora, vas a dividir tu conjunto de puntos en 2 mitades y llamar recursivamente a tu función, cuando llegues a una cantidad de puntos pequeña puedes resolverlo por fuerza bruta.

Ahora, una vez que tengas la distancia mínima, tienes que calcular a lo largo del punto medio con un radio de la distancia (strip) mínima para ver si existe otro posible candidato.

Ahora simplemente tu algoritmo recursivo regresa el menor de todos.

## 4. Inversion Count

### 4.1. Código de Refencia

```
1  /*
2   ===== COUNTER INVERSION =====
3   */
4 #include <stdio.h> // Include Libraries
5 #include <set>
6
7 using namespace std; // Bad practice
8
9 typedef long long int lli;
10
11 template <class T> inline void GetNumber(T &Number) { // Super cool way to read
12     lli s
13
14     Number = 0; // Set to 0
15     T NumberSign = 1; // Sign all ok
16     char CurrentChar = getchar_unlocked(); // Read a char
17
18     while (CurrentChar < '0' or CurrentChar > '9') { // If we are not reading numbers
19         if (CurrentChar == '-') NumberSign = -1; // If we have found the sign
20         CurrentChar = getchar_unlocked(); // Read the next char
21     }
22
23     while (CurrentChar >= '0' and CurrentChar <= '9') { // While we read numbers
24         Number = (Number << 3) + (Number << 1) + CurrentChar - '0'; // Multiply by 10
25         CurrentChar = getchar_unlocked(); // Read the next char
26     }
27
28     Number *= NumberSign; // If negative
29 }
30
31
32 lli MergeSort(lli Data[], lli Temporal[], lli Left, lli Right); // Declarations
33 lli Merge(lli Data[], lli Temporal[], lli Left, lli Middle, lli Right); // Declarations
34
35
36 lli CountInversions(lli Data[], lli Size) { // Count the inversions
37     lli Temporal[Size]; // Create temporal array
38     return MergeSort(Data, Temporal, 0, Size - 1); // Call merge
39 }
40
41 lli MergeSort(lli Data[], lli Temporal[], lli Left, lli Right) { // Return num of inversions
42     lli Middle, NumInversions = 0; // Local vars
43
44     if (Right > Left) { // If all ok
45         Middle = (Right + Left) / 2; // Middle
46
47         NumInversions = MergeSort(Data, Temporal, Left, Middle); // Find the left
48         NumInversions += MergeSort(Data, Temporal, Middle + 1, Right); // Find the right
49
50         NumInversions += Merge(Data, Temporal, Left, Middle + 1, Right); // Find in the middle
51     }
52
53     return NumInversions; // Return it!
54 }
55
56
57 }
```

```

1 lli Merge( lli Data[], lli Temporal[], lli Left, lli Middle, lli Right) { //Return the middle
2
3     lli NumInversions = 0; //The return var
4     lli i = Left, j = Middle, k = Left; //Local vars
5
6     while ((i <= Middle - 1) and (j <= Right)) { //While we can
7         if (Data[i] <= Data[j]) Temporal[k++] = Data[i++]; //If no inversions
8         else { //Add the inversion
9             Temporal[k++] = Data[j++];
10            NumInversions += (Middle - i); //Add it
11        }
12    }
13
14    while (i <= Middle - 1) Temporal[k++] = Data[i++]; //Move the other
15    while (j <= Right) Temporal[k++] = Data[j++]; //Move the others
16
17    for (i = Left; i <= Right; i++) Data[i] = Temporal[i]; //Move
18
19    return NumInversions; //Add inversions
20
21 }
22
23
24
25
26
27 /*=====
28 ====== MAIN ======
29 =====*/
30 int main() {
31
32     lli TestCases;
33     GetNumber<lli>(TestCases);
34
35     for (lli i = 0; i < TestCases; ++i) {
36
37         lli Size;
38         GetNumber<lli>(Size);
39
40
41         lli Data[Size];
42         for (lli j = 0; j < Size; ++j) {
43             GetNumber<long long int>(Data[j]);
44         }
45
46         printf("%d\n", CountInversions(Data, Size));
47     }
48
49     return 0;
50 }
```

## 4.2. Accepted

The screenshot shows the Sphere online judge interface. At the top, there are navigation links: PROBLEMS, STATUS, RANKS, DISCUSS, CONTESTS, PROFILE, and a dropdown for PROFILE. Below this, a breadcrumb trail shows 'Profile / History of submissions'. A search bar contains the placeholder ': submissions'. The main area displays a table of submission history:

ID	DATE	PROBLEM	RESULT	TIME	MEM	LANG
21650340	2018-05-13 09:33:33	Inversion Count	<b>accepted</b> edit ideone it	0.04	19M	CPP14
21650331	2018-05-13 02:17:49	Inversion Count	time limit exceeded edit ideone it	-	18M	CPP14
21650328	2018-05-13 02:16:32	Inversion Count	compilation error edit ideone it	-	-	CPP14

Below the table are buttons for 'Invert' and 'Execute', and a dropdown menu for 'Selected submissions'. At the bottom, there are links for About, Tutorial, Tools, Clusters, Credits, Jobs, API, Widgets, Terms, and RSS.

### 4.3. Idea del Algoritmo

- **Forma Estúpida:** La forma estupida, pero muy bonita es recorrer el arreglo y para cada elemento volver a recorrer el arreglo (desde el elemento en el que estoy hasta el final) y ver si  $A[i] > A[j]$

Como estamos recorriendo el arreglo  $n$  por  $n - i$  veces tenemos que toma  $O(n^2)$

- **Forma Divide and Conquer:**

Vamos a hacer un merge sort modificado, solo que en cada paso del merge() vamos a llevar una cuenta de la cantidad de inversiones que hacemos.

Solo que tenemos un detalle, supongamos que ya sabemos el número de inversiones en la mitad izquierda y la mitad derecha del arreglo

Entonces las inversiones que nos faltan por contar son las que son inversiones donde el primer elemento este en la mitad izquierda y el otro en la mitad derecha.

En el proceso de merge, estemos en el  $i$ -ésimo elemento del subarreglo derecho y el  $j$ -ésimo el arreglo izquierdo.

En cualquier paso de merge(), si a  $A[i]$  es mayor que a  $A[j]$ , entonces hay tenemos la cantidad de la derecha -  $i$  inversiones. Porque el subarreglo izquierdo y derecho se ordenan, por lo que todos los elementos restantes en subcampo izquierdo serán mayores que a  $A[j]$

## Referencias

- [1] Ethan Jimenez *Ninja Ref.*
- [2] *Divide and Conquer Problem Set 1, 2, 3.*