

Preparación Específica para Concursos ACM-ICPC.

Tema #7 : Teoría de números y Geometría.

Conferencia #14: Geometría. Puntos más cercanos.

Objetivos

- ❑ Caracterizar el problema de los puntos más cercano.
- ❑ Caracterizar las variantes del algoritmo Closest Pair Problem.

Contenidos

- ❑ Problemas de los puntos más cercanos.
- ❑ Algoritmo Closest Pair Problem y sus variantes.

Bibliografía

- *Manual de preparación para concursantes ACM-ICPC de la Universidad de Matanzas.*

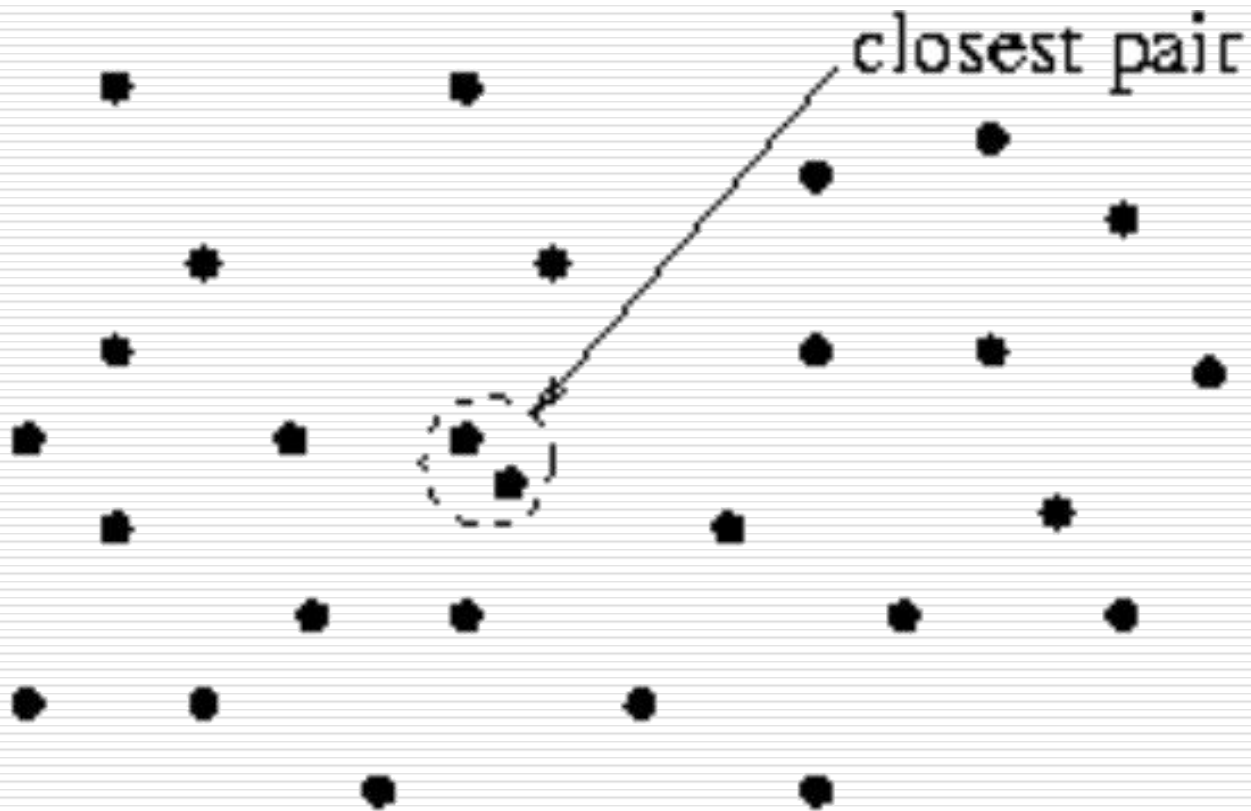
Puntos más cercanos

- El problema de los puntos más cercanos (Closest Pair Problem) parte de un conjunto de n puntos pertenecientes al plano XY , donde cada punto está representado por sus coordenadas X y Y ...

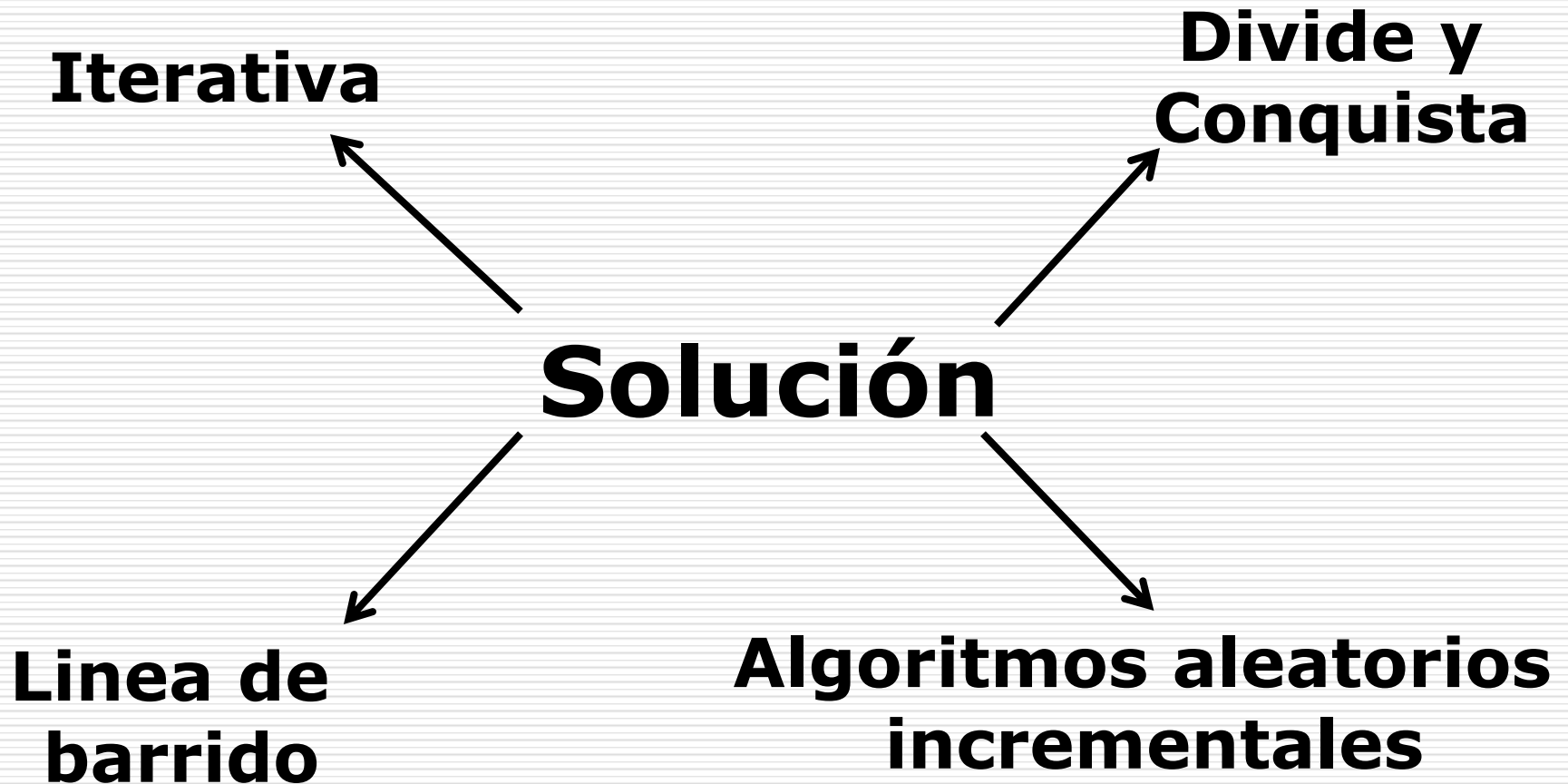
Puntos más cercanos

- ... se desea hallar encontrar el par de puntos tal que la distancia entre ambos puntos es mínima. El algoritmo debe devolver dichos puntos o la distancia.

Puntos más cercanos



Puntos más cercanos



Iterativa

- La solución iterativa a este problema es simple, se recorre la colección de puntos y por cada punto se le compara contra todos los demás calculando la distancia.

Iterativa

- La solución iterativa a este problema es simple, se recorre la colección de puntos y por cada punto se le compara contra todos los demás calculando la distancia ...

Iterativa

- ... cada vez que una distancia sea inferior al mínimo anterior actualizamos cuales son los puntos más cercanos. Cuando ya comparamos todos los puntos contra todos el resultado es el par de puntos cuya distancia es seguro es mínima.

Iterativa

- Es evidente que esta solución tanto en peor como en el mejor de los casos compara todos los puntos contra todos por lo que podemos decir que la complejidad de este algoritmo es $O(n^2)$ siendo n la cantidad de puntos en el plano.

Divide y Conquista

- Hasta este momento tenemos una solución al problema cuya complejidad es $O(n^2)$, es por eso que en este apartado vamos a intentar aplicar conocimientos en algoritmos de tipo "Dividir para Conquistar".

Divide y Conquista

- ❑ **Dividir:** Partimos que tenemos los puntos almacenados en alguna estructura de dato secuencial. Si son pocos puntos podemos aplicar la primera variante.

Divide y Conquista

- **Dividir:** Si hay más puntos de lo permisible entonces trazamos una línea vertical l que subdivide a la colección P de puntos en dos colecciones aproximadamente del mismo tamaño, P_l y P_r .

Divide y Conquista

- ❑ **Conquistar:** Recursivamente aplicamos el procedimiento en ambas colecciones por lo que obtenemos δ_l y δ_r distancias mínimas para ambas colecciones. De ambas obtenemos $\delta = \min(\delta_l, \delta_r)$.

Divide y Conquista

- ❑ **Combinar:** Lamentablemente δ no es el resultado final, ya que se podría darse la línea l que hemos elegido pasa justo entre dos puntos que están a distancia mínima. Debemos chequear si no hay dos puntos que están a distancia mínima.

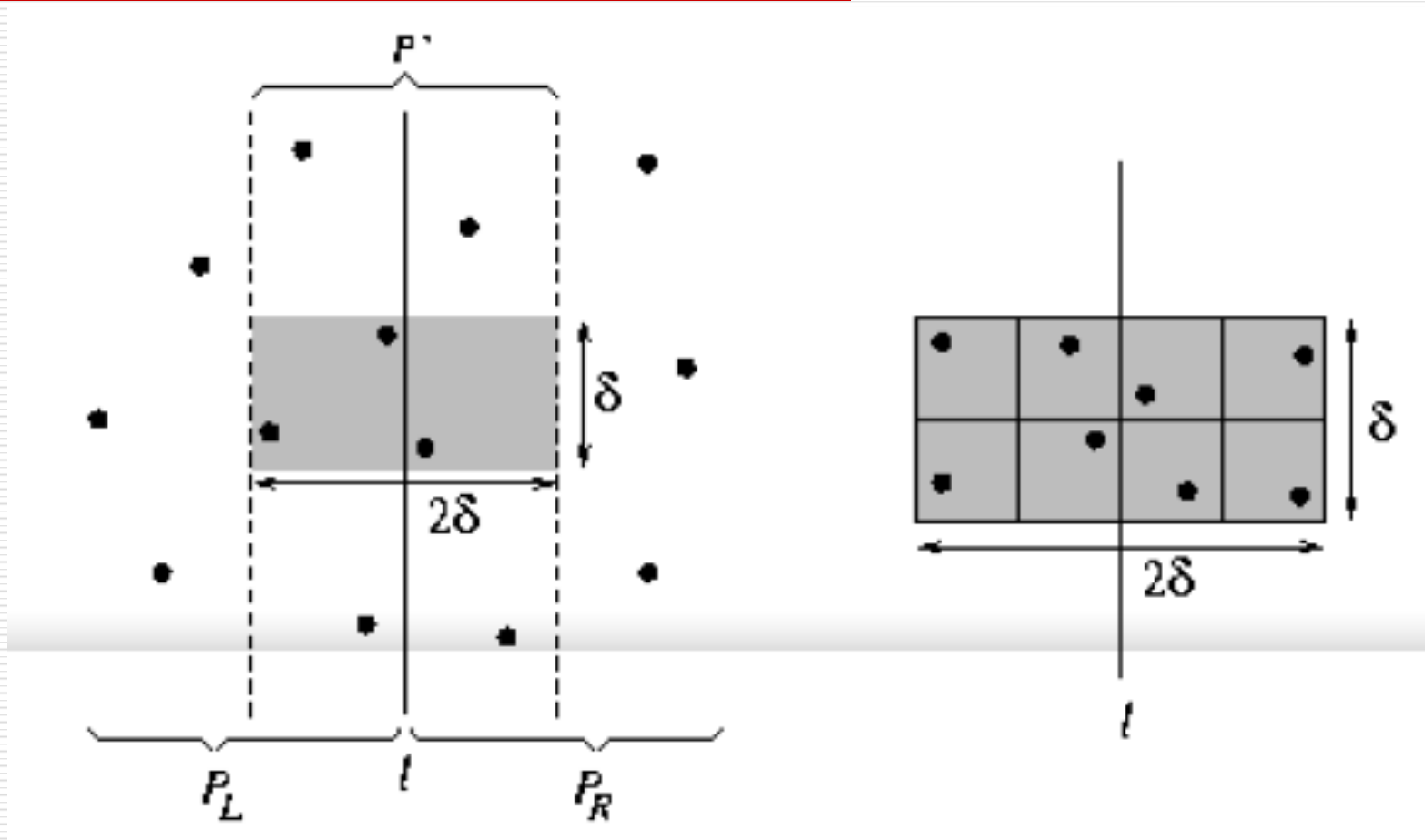
Divide y Conquista

- ❑ **Combinar:** Debemos chequear si no hay dos puntos, uno a cada lado de la línea, cuya distancia sea menor que δ . En primer lugar observemos que no necesitamos chequear todos los puntos, si un punto está a mayor distancia de l que δ entonces es seguro no hay vecino del otro lado de la línea que pueda formar una distancia menor que δ .

Divide y Conquista

- ❑ **Combinar:** Creamos una lista P' de puntos que están a menos de δ de cada lado de l . Determinamos entonces la distancia mínima para los puntos de P' que llamaremos δ y devolvemos la distancia sino los dos puntos que están a dicha distancia uno del otro.

Divide y Conquista



Divide y Conquista

- ❑ Resumiendo el funcionamiento del algoritmo:
 - ❑ **Presort:** Dada la lista P, hacemos dos copias PX y PY. Ordenamos PX por la coordenada x y la lista PY por las coordenadas y.
 - ❑ **Parte Recursiva:** DistMin(X,Y)
 - ❑ **Condición de corte:** Si la cantidad de puntos es menor que 4 entonces resolvemos el problema por fuerza bruta y devolvemos la distancia mínima δ analizando todas las distancias posibles.

Divide y Conquista

- ❑ Resumiendo el funcionamiento del algoritmo:
- ❑ **Parte Recursiva:** $\text{DistMin}(X,Y)$
 - ❑ **Dividir:** Sino, sea l la mediana de las coordenadas x de PX . Dividimos ambas listas PX y PY por esa línea, manteniendo el orden, creando Xl, Xr, Yl, Yr .
 - ❑ **Conquistar:** $\delta l = \text{DistMin}(Xl, Yl)$, $\delta r = \text{DistMin}(Xr, Yr)$
 - ❑ **Combinar:** $\delta = \min(\delta l, \delta r)$. Creamos la lista Y' copiando todos los puntos de Y que están a distancia menor que δ de l . Para i desde 1 hasta la longitud de Y' y para j desde $i+1$ hasta $i+7$ calcular la distancia entre $Y'[i]$ y $Y'[j]$. Sea δ' la distancia mínima entre dichas distancias. Devolver $\min(\delta, \delta')$.

Línea de barrido

- En la geometría computacional, un algoritmo de línea de barrido o barrido de plano es un paradigma algorítmico que utiliza una línea de barrido conceptual o una superficie de barrido para resolver diversos problemas en el espacio euclidiano.

Línea de barrido

- Con ese algoritmo, barreremos el plano de izquierda a derecha (o de derecha a izquierda es una posibilidad) y cuándo se alcance un punto computaremos todos los puntos candidatos cercanos a este (los candidatos que puede estar en el par más cercano).

Línea de barrido

- Por lo que haremos las siguientes operaciones:
 - Ordenamos los puntos de izquierda a derecha por el eje x.

Línea de barrido

- Por lo que haremos las siguientes operaciones:
 - Por cada punto:
 - Quitamos de los candidatos todo el punto que está más allá en el eje x de la distancia mínima actual. Tomamos a todos los candidatos que son localizados a igual o menor distancia que la distancia mínima del punto actual en eje vertical. Probamos para la distancia mínima todos los candidatos encontrados con el punto actual. Y finalmente le añadimos el punto actual a la lista de candidatos.

Algoritmos aleatorios incrementales

- Una sub-división de los algoritmos aleatorizados son los algoritmos incrementales en este caso vamos presentar un algoritmo tipo Las Vegas incremental para resolver el problema de los puntos mas cercanos en un plano.

Algoritmos aleatorios incrementales

- La idea es la siguiente, los puntos van ser insertados uno a uno en un conjunto inicialmente vacío y por cada punto que insertamos vamos a chequear si no es necesario actualizar cual es la distancia mínima.

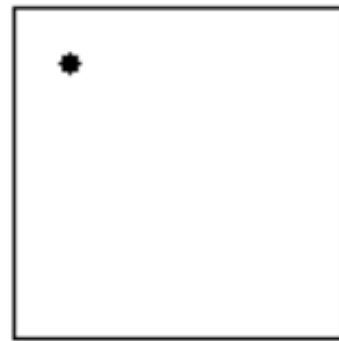
Algoritmos aleatorios incrementales

- El funcionamiento del algoritmo dependerá del orden en que se inserten los puntos, algunos ordenamientos serán particularmente malos ya que necesitaremos actualizar la distancia mínima por cada punto insertado lo cual nos insumirá tiempo de $O(n^2)$.

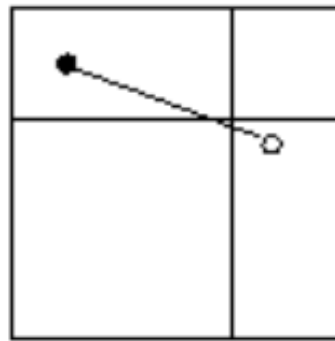
Algoritmos aleatorios incrementales

- Sin embargo al insertar los puntos en orden aleatorio el tiempo esperado del algoritmo será de $O(n)$. No lo vamos a demostrar pero la probabilidad de que el algoritmo insuma mas de $O(n \log n)$ es extremadamente chica.

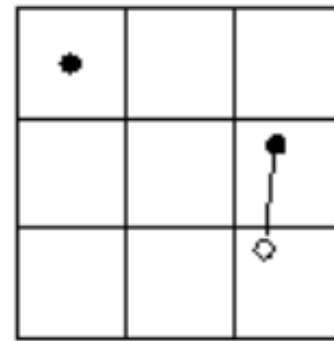
Algoritmos aleatorios incrementales



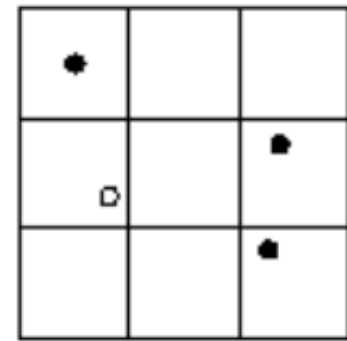
P_1



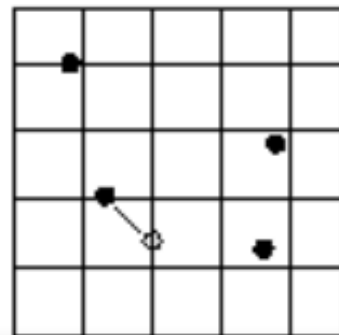
P_2



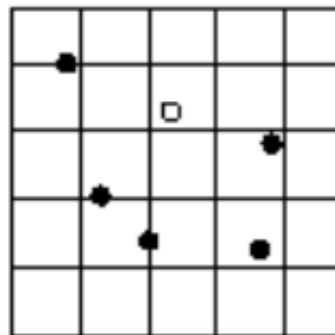
P_3



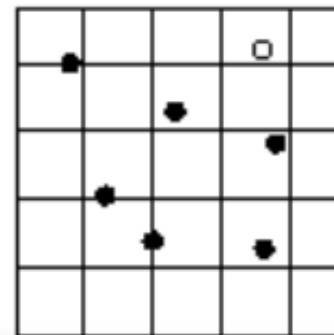
P_4



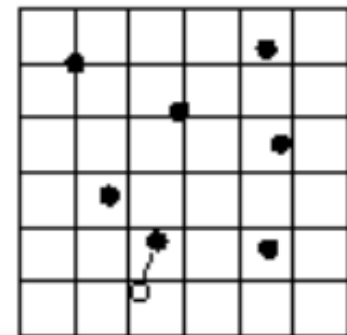
P_5



P_6



P_7



P_8

Conclusiones

- Aquí termina nuestro seguimiento del problema de la distancia mínima en el plano, hemos visto como a medida que estudiábamos distintas técnicas para el diseño de algoritmos podíamos resolver el problema en forma más eficiente.

Conclusiones

- Planteamos primero un algoritmo iterativo que usando fuerza bruta resolvía el problema en $O(n^2)$, luego mediante un algoritmo de tipo “Divide & Conquer” obtuvimos $O(n \log n)$, por último mediante un algoritmo aleatorizado resolvimos el problema en $O(n)$.

Estudio Independiente

- Profundizar en los temas abordados con la lectura del capítulo Geometría del manual mencionado en la bibliografía del curso.

Estudio Independiente

- ❑ Solucionar de Juez Caribeño Online COJ los siguientes problemas.
- ❑ 3655 – Inside the Matrix
- ❑ 1847 - Claustrophobic Cows
- ❑ 1080 - The Closest Pair Problem

Preparación Específica para Concursos ACM-ICPC.

Tema #7 : Teoría de números y Geometría.

Conferencia #13: Números primos.