.UBAfiuba
FACULTAD DE INGENIERÍA

Randomizado: Puntos más cercanos en el plano

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Problema

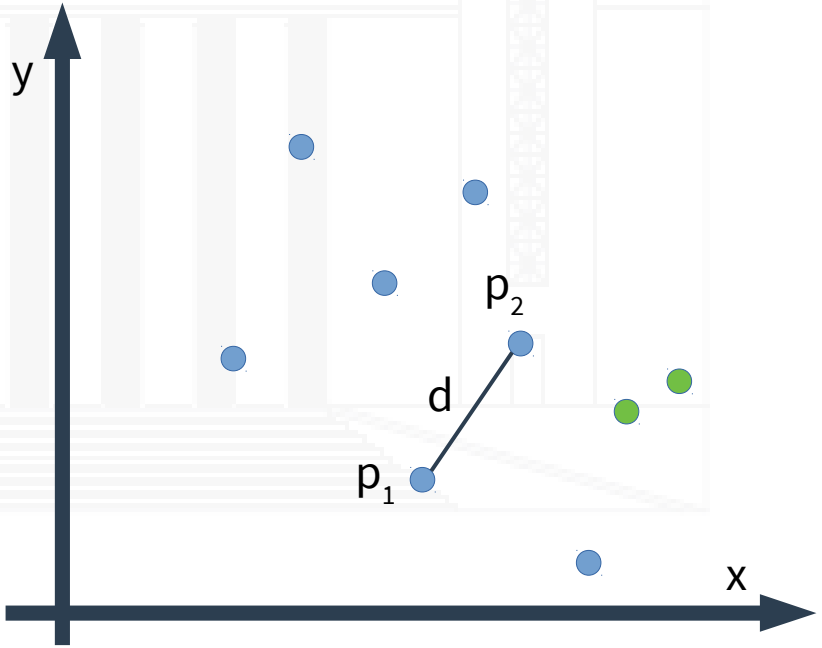
Sea

P un conjuntos de “n” puntos en el plano

$d(p_1,p_2)$ la función distancia entre $p_1=(x_1,y_1)$, $p_2=(x_2,y_2) \in P$

Queremos

Encontrar los puntos más cercanos en P



Un problema conocido

Ya planeamos una solución al problema

Utilizando división y conquista en O(nlogn)

¿Podemos hacerlo mejor?

Plantearemos una manera utilizando randomización

Esperaremos que funcione en O(n)

Utilizaremos el método propuesto en 1995 por

M. Golin, R. Raman, C. Schwarz y M. Smid

“Simple randomized algorithms for closest pair problems”

https://people.scs.carleton.ca/~michiel/simplerando.pdf

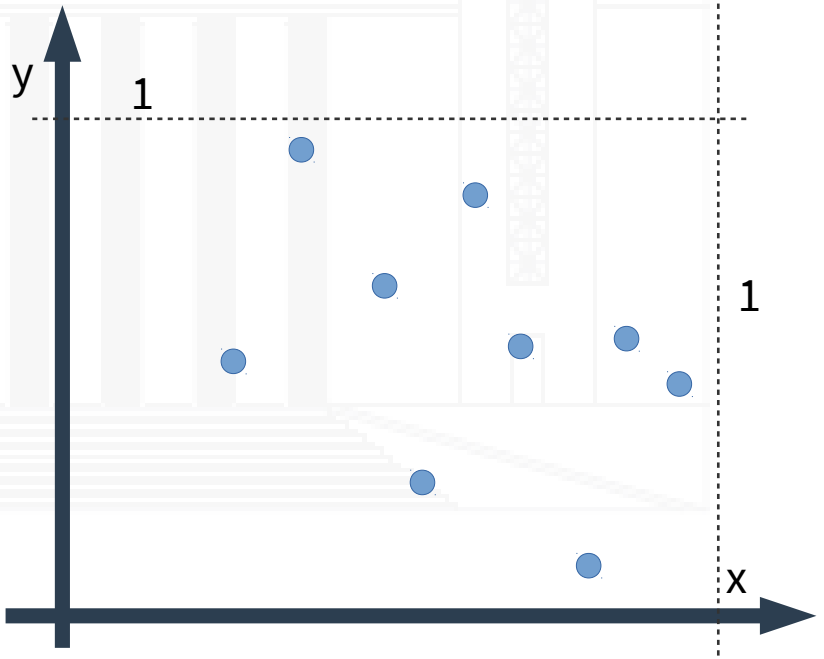
Consideraciones

Utilizaremos como supuesto

Que cada punto $i=(x_i,y_i)$ se encuentra entre las coordenadas $0 < x_i, y_i \leq 1$

Si no

Se pueden escalar en tiempo lineal



Un proceso gradual

Comenzaremos unicamente con 2 puntos: p_1 y p_2

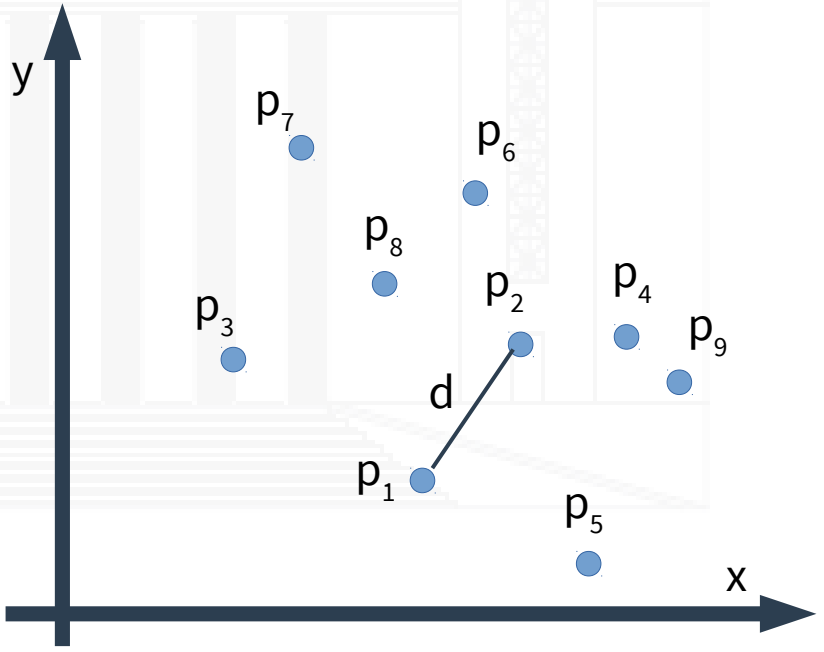
Consideraremos que esos son los más cercanos → $\delta = d(p_1, p_2)$

Gradualmente iremos agregando de a 1 punto (p_3, p_4,...)

Y verificaremos si el nuevo punto es más cercano a alguno de los anteriores analizados que el par actual

¿Cómo evitamos

tener que comparar un nuevo punto con todos los anteriores?



Regionalización del área

Podemos construir una regionalización

del área de los puntos

Utilizaremos como medida

La mitad de la distancia menor actual

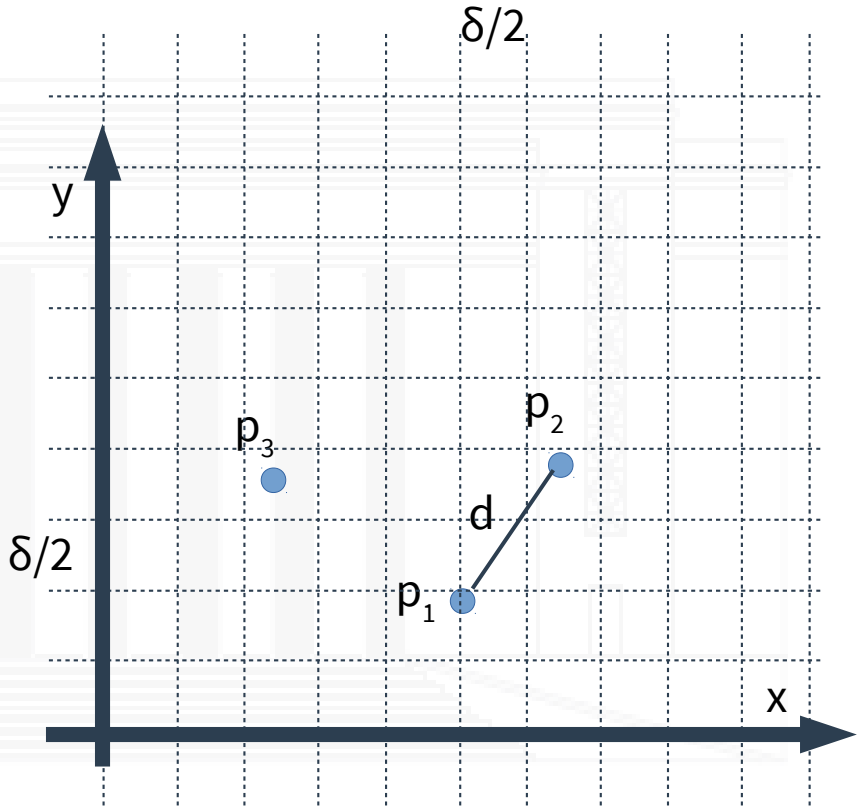
Para un nuevo punto ver

Si se encuentra en la misma celda

O en alguna de sus circundantes

En ese caso

Puede ser un nuevo punto mas cercano con alguno de los puntos preexistentes



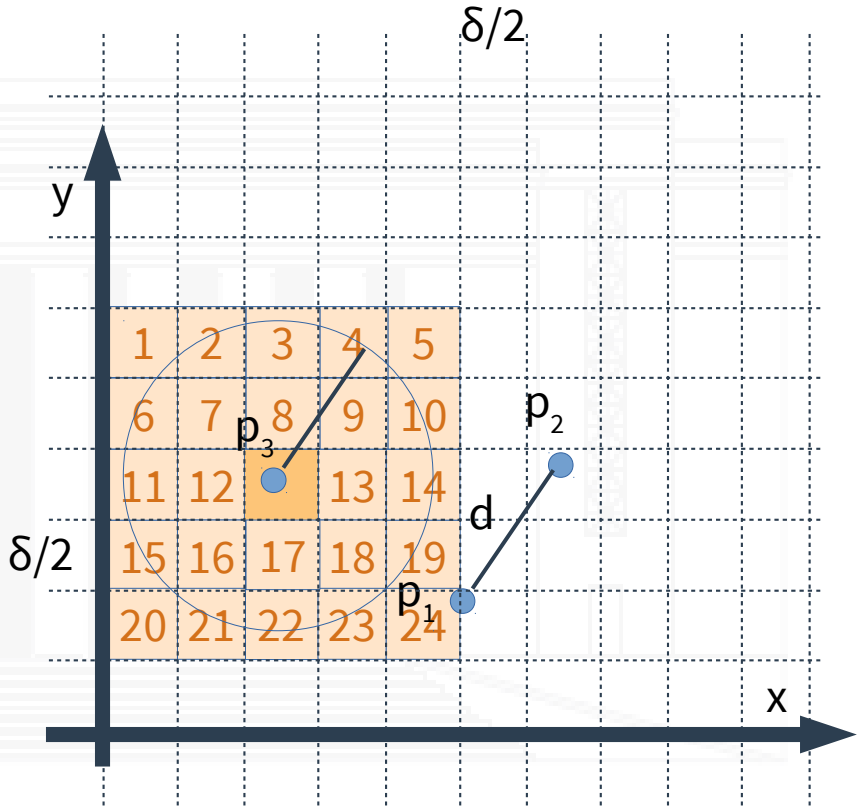
Regionalización del área (cont.)

Verificar si están en la misma o en alguna de las 24 celdas circundantes

- Y su distancia es menos al actual y al resto de los candidatos
- (pueden existir varios puntos en esa situación)
- En ese caso también reemplazan a los más cercanos

Si no

- Sigue siendo el par anterior el más cercano
- Lo agregamos a la grilla y probamos con el siguiente



Pseudocodigo (parcial)

```
Definir la menor distancia δ=d(p1,p2)
Crear grilla G con tamaño δ/2
Insertar p1 y p2 celda correspondiente de G

Desde i=3 a n
    Sea c celda correspondiente a pi
    Sean R los puntos en las 25 celdas cercanas a c en G
                                          // pueden ser de 0 o 25
    Calcular δr para rx ϵ R / min{d(pi,rx)}
    Si δr < δ
        δ = δr
        …
    Sino
        Agregar pi a grilla en celda correspondiente en G
Retornar δ
```

Análisis (parcial)

El proceso exterior se realiza O(n)

1 por cada punto

Analizar si el mínimo actual sigue siéndolo

Requiere realizar una inspección de 25 celdas → O(1) ? (usamos una matriz??)

Y calcular un máximo de k=25 distancias → O(1)

Si el punto analizado no conforma el mínimo

se agrega a la grilla → O(1) ?

Si el punto es el nuevo mínimo

…?

Un nuevo mínimo

Si un punto p_i recién insertado conforma con uno previo

El nuevo mínimo $\delta = d(p_i,p_r)$ se debe tomar a consideración ($r<i$)

Crear una nueva grilla con celdas de $(\delta / 2) \times (\delta / 2)$

Reinsertar los i-1 puntos previos y el recién evaluados

¿Cómo podemos caracterizar este proceso?

reinsertar cada punto es O(1)

Insertar el punto i requiere i operación O(1)

¿Cuantas veces se ejecuta el reinsertar?

Pseudocodigo (actualizado)

```
Definir la menor distancia δ=d(p1,p2)
Crear grilla G con tamaño δ/2
Insertar p1 y p2 celda correspondiente de G

Desde i=3 a n
    Sea c celda correspondiente a pi
    Sean R los puntos en las 25 celdas cercanas a c en G
                                          // pueden ser de 0 o 25
    Calcular δr para rx ϵ R / min{d(pi,rx)}
    Si δr < δ
        δ = δr
        Crear grilla G con tamaño δ/2
        Desde j=1 a i
            Insertar pj en la grilla
    Sino
        Agregar pi a grilla en celda correspondiente en G
Retornar δ
```

Análisis: El mejor caso

Supongamos

Que la distancia mínima corresponde a los puntos p1 y p2

En ese caso

El reinsertar no se llama ninguna vez!

El proceso

Se ejecuta 1 vez para cada punto

Para cada punto se verifican a lo sumo 25 distancias

Globalmente

Tiene una complejidad de O(n)

Análisis: El peor caso

Supongamos

Que cada punto ingresado corresponde a un cambio del mínimo

En ese caso

El reinsertar se llama n veces!

El proceso

Se ejecuta 1 vez para cada punto

Para cada punto se verifican a lo sumo 25 distancias

Se regenera la grilla

Se reinsertan los i puntos previos por cada punto

Globalmente

Tiene una complejidad de $O(n^2)$

Otro caso...

En los casos anteriores

Asumimos un determinado orden en el procesamiento de los puntos

Qué pasa si suponemos

que el orden de los puntos es aleatorio?

Será similar a alguno de los casos anteriores? O algo intermedio?

Analizaremos probabilisticamente esta situación

Estimación

Llamaremos

X a la variable aleatoria que especifica la cantidad total de operaciones de inserción en la grilla realizadas

X_i a la variable aleatoria igual a 1 si el i-esimo punto en el orden aleatorio causa que la distancia mínima cambie. Sino toma el valor de 0

Podemos representar

$$X = n + \sum_{i=1}^{n} i\, X_i$$

Si el i-esimo punto es 1 –> tengo que reinsertar los i puntos en la nueva grilla

Todos los puntos los inserto al menos 1 vez

Probabilidad de cambio de mínimo

Sean

$P^* = p_1, p_2, \ldots, p_i$ los primeros i puntos

$r, s \in P^*$ los puntos de menor distancia

$$P[X_i = 1] \leq 2/i$$

Solo hay cambio de mínimo

si $p_i = r$ o $p_i = s$

Dado que los puntos están en orden aleatorio

La probabilidad de que r (s similarmente) sea el ultimo punto es 1/i

Por lo tanto

La probabilidad de cambio es $P[X_i=1] = 1/i + 1/i = 2/i$

Esto corresponde a una cota superior

Dado que pueden existir otros pares de puntos con igual distancia a r y s en cuyo caso no se haría el cambio de mínimo

Valor esperado de inserciones

Con

la probabilidad calculada $P[X_i=1] \leq 2/i$

La variable aleatoria $X = n + \sum_{i=1}^{n} i\, X_i$

Podemos calcular

El valor esperado $E[X] = n + \sum_{i=1}^{n} i\, E[X_i] \leq n + \sum_{i=1}^{n} i * 2/i \Rightarrow E[X] \leq n + 2n = 3n$

En conclusión:

Si el orden de los puntos es aleatorio ESPERAMOS O(n) operaciones de inserciones.

Por lo tanto

El proceso global es O(n)

… pero debemos asegurarnos QUE LOS PUNTOS ESTÉN EN ORDEN ALEATORIOS (con alta probabilidad!)

Pseudocodigo (actualizado)

```
Mezclar aleatoreamente los puntos
Definir la menor distancia δ=d(p1,p2)
Crear grilla G con tamaño δ/2
Insertar p1 y p2 celda correspondiente de G

Desde i=3 a n
    Sea c celda correspondiente a pi
    Sean R los puntos en las 25 celdas cercanas a c en G
                                          // pueden ser de 0 o 25
    Calcular δr para rx ϵ R / min{d(pi,rx)}
    Si δr < δ
        δ = δr
        Crear grilla G con tamaño δ/2
        Desde j=1 a i
            Insertar pj en la grilla
    Sino
        Agregar pi a grilla en celda correspondiente en G
Retornar δ
```

Un problema de tamaño...

Al realizar el análisis de complejidad espacial

Se revela un problema

La cantidad celdas de la grilla

Puede crecer velozmente

La cantidad

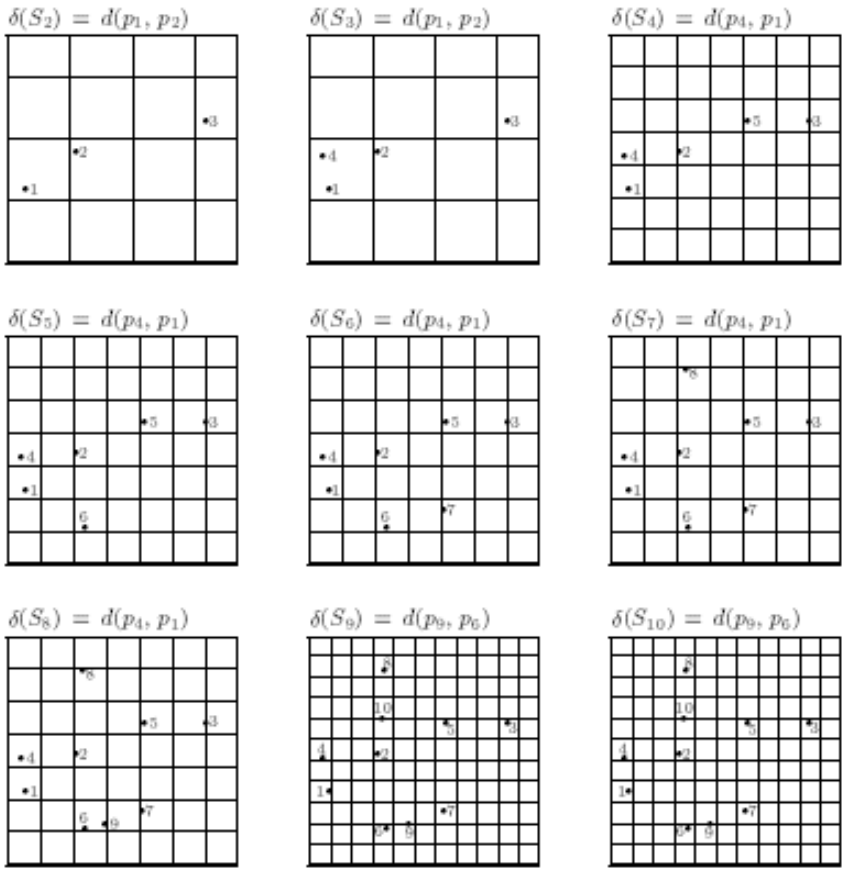
no depende de la cantidad de puntos

Depende de la distancia mínima encontrada

La utilización de una matriz

puede volverse inmanejable

Se debe encontrar una alternativa



Alternativas para la grilla

Existen varias alternativas.

Los autores de la solución proponen:

- Arboles de búsquedas balanceados
- Hashing perfecto dinámico

La utilización de arboles de búsqueda balanceados

- Permiten buscar el punto mas cercano en la etapa i en O(log(i))
- Llevando por lo tanto la complejidad temporal esperada del proceso a O(nlogn)

Diccionarios

El uso de un diccionario

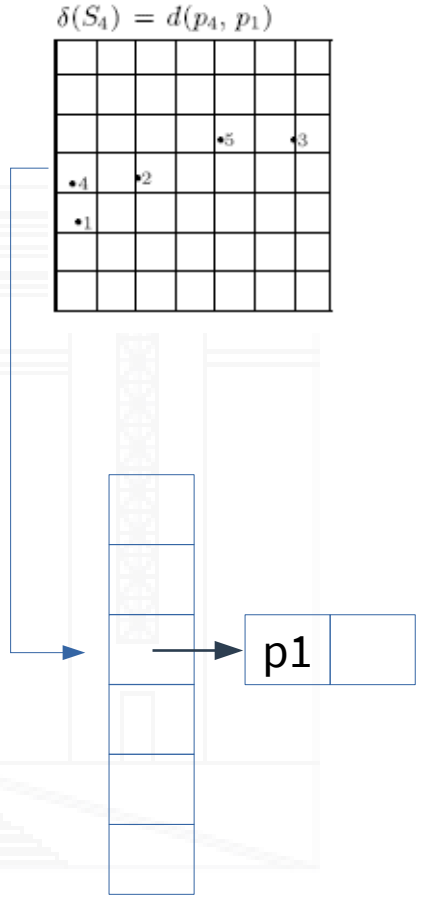
Parece ideal para este problema

Tenemos un universo grande de elementos (celdas)

(podemos nomenclar cada uno de las celdas de la grilla)

Y un subconjunto acotado de elementos a tratar

(a lo sumo 1 celda por punto)



Hashing perfecto dinámico

Corresponde a una estructura de datos

- Que “espera” restringir las colisiones
- Utiliza Clases universal de funciones de Hashing
- Requiere conocer el posible universo de items a insertar
- (lo sabemos! escalamos los puntos y conocemos por δ la cantidad de celdas)

Permite

- Crear el diccionario con n puntos (ocupando celdas) en tiempo esperado O(n)
- Buscar un punto en O(1)
- insertar un punto en una celda en esperado O(1)

Pseudocodigo (final)

```
Mezclar aleatoreamente los puntos
Definir la menor distancia δ=d(p1,p2)
Crear diccionario G con celdas de tamaño δ/2
Insertar p1 y p2 en el diccionario

Desde i=3 a n
    Sea c celda correspondiente a pi
    Sean R los puntos en las 25 celdas cercanas a c en G
                                          // pueden ser de 0 o 25
    Calcular δr para rx ϵ R / min{d(pi,rx)}
    Si δr < δ
        δ = δr
        Crear diccionario G con celdas de tamaño δ/2
        Desde j=1 a i
            Insertar pj en el diccionario
    Sino
        Agregar pi a grilla en celda correspondiente en G
Retornar δ
```

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Enero de 2021