PROBLEMA DEL PAR DE PUNTOS MÁS CERCANOS: DIVIDE Y VENCERÁS

ÁLVARO BERRÍO GALINDO

PRESENTACIÓN DEL PROBLEMA

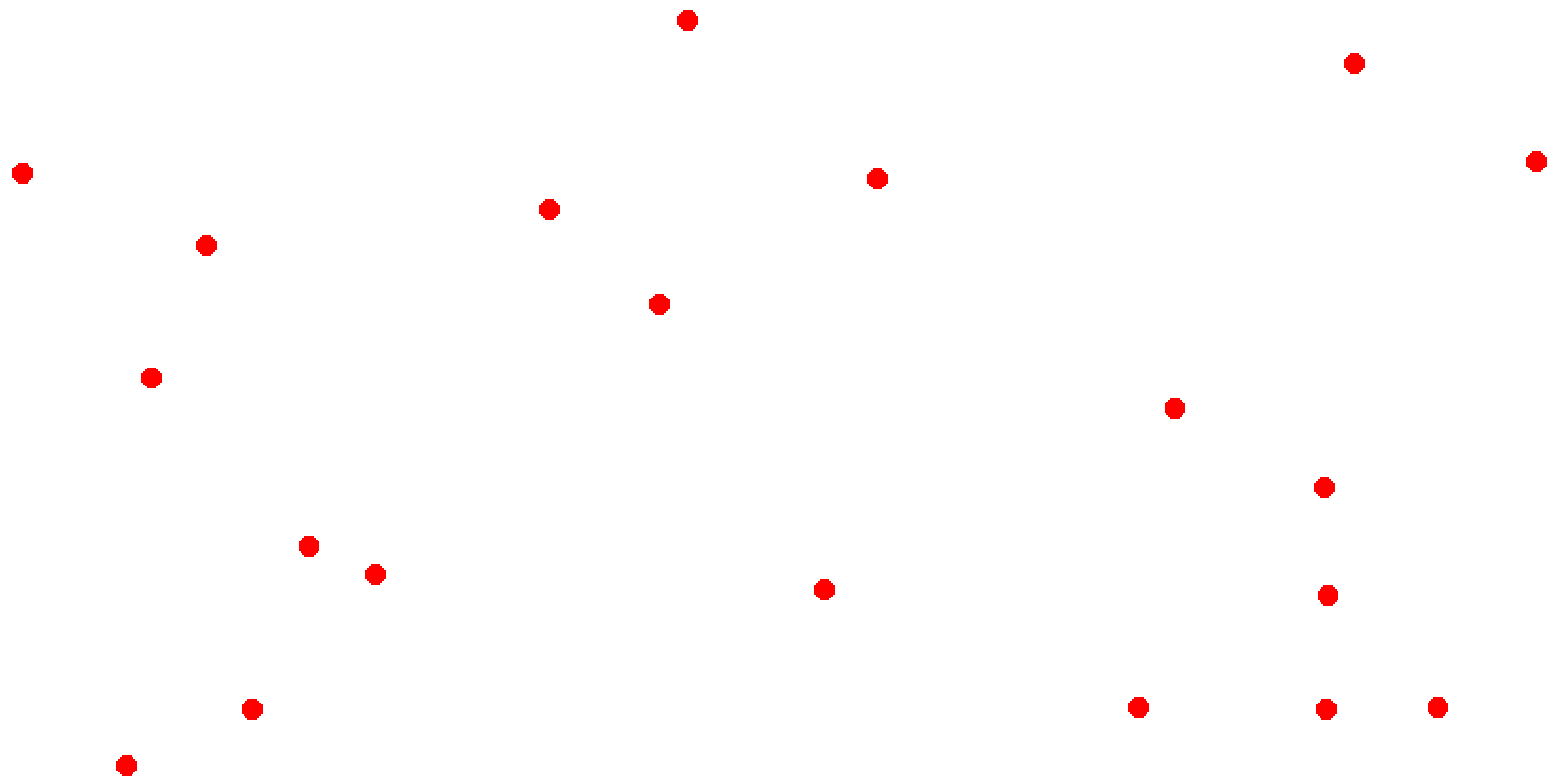
- Dada una masa de puntos, encontrar el par de puntos cuya distancia es la mínima.
- Pertenece a la problemática de la geometría computacional.
- Fue planteado por Shamos y Hoey en la década de los 70.
- De los primeros problemas geométricos en ser usados para calcular la complejidad de algoritmos.

- Se resuelve mediante algoritmos Divide y Vencerás.
- La idea principal de los algoritmos Divide y Vencerás consiste en dividir el problema principal en subproblemas con un tamaño manejable.
- Aplicaciones del problema del par de puntos más cercanos: finanzas, geografía, predicciones meteorológicas, etc. Debido a la relación con trabajos con grandes masas de datos se necesita eficiencia.

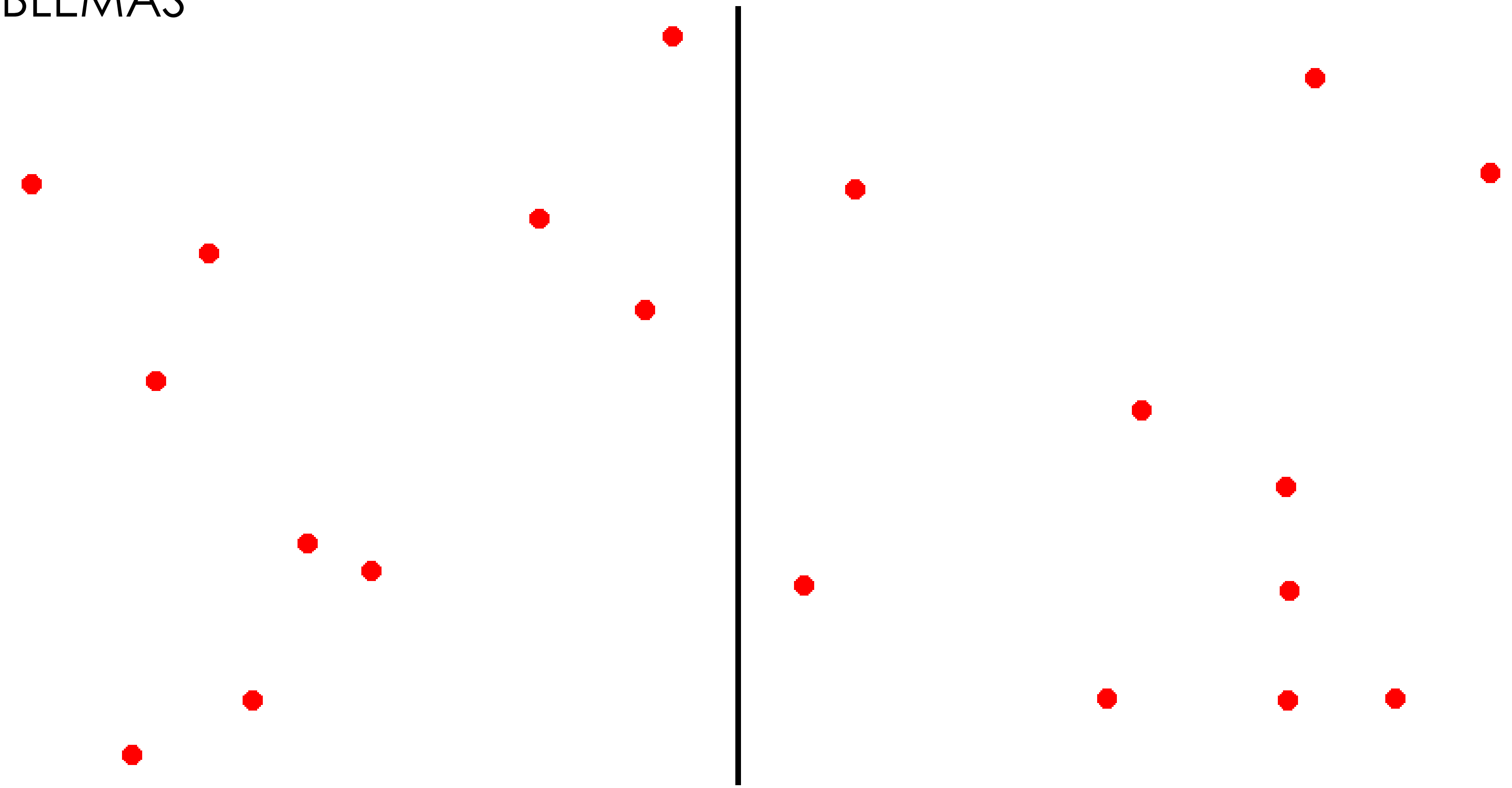
DESCRIPCIÓN DEL ALGORITMO

- Mediante la fuerza bruta el problema consiste en hallar la distancia entre todos los puntos del conjunto y devolver la mínima pero el coste es muy alto.
- Se plantea el algoritmo Divide y Vencerás para los casos de 2 y 3 dimensiones.
- Antes de comenzar el algoritmo Divide y Vencerás es necesario ordenar el conjunto de puntos según las coordenadas x e y.

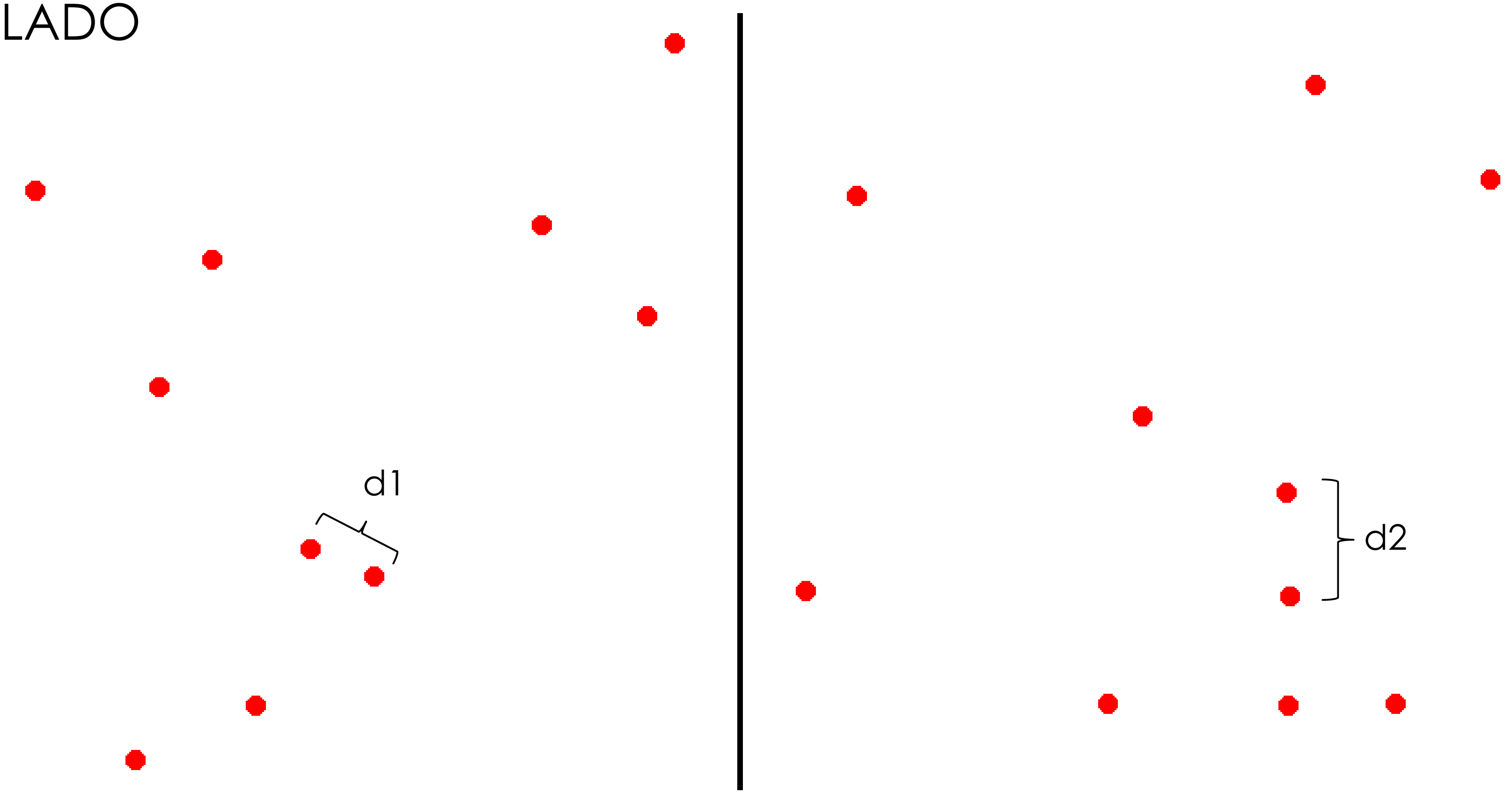
CONJUNTO DE PUNTOS



DIVISIÓN EN DOS SUBPROBLEMAS

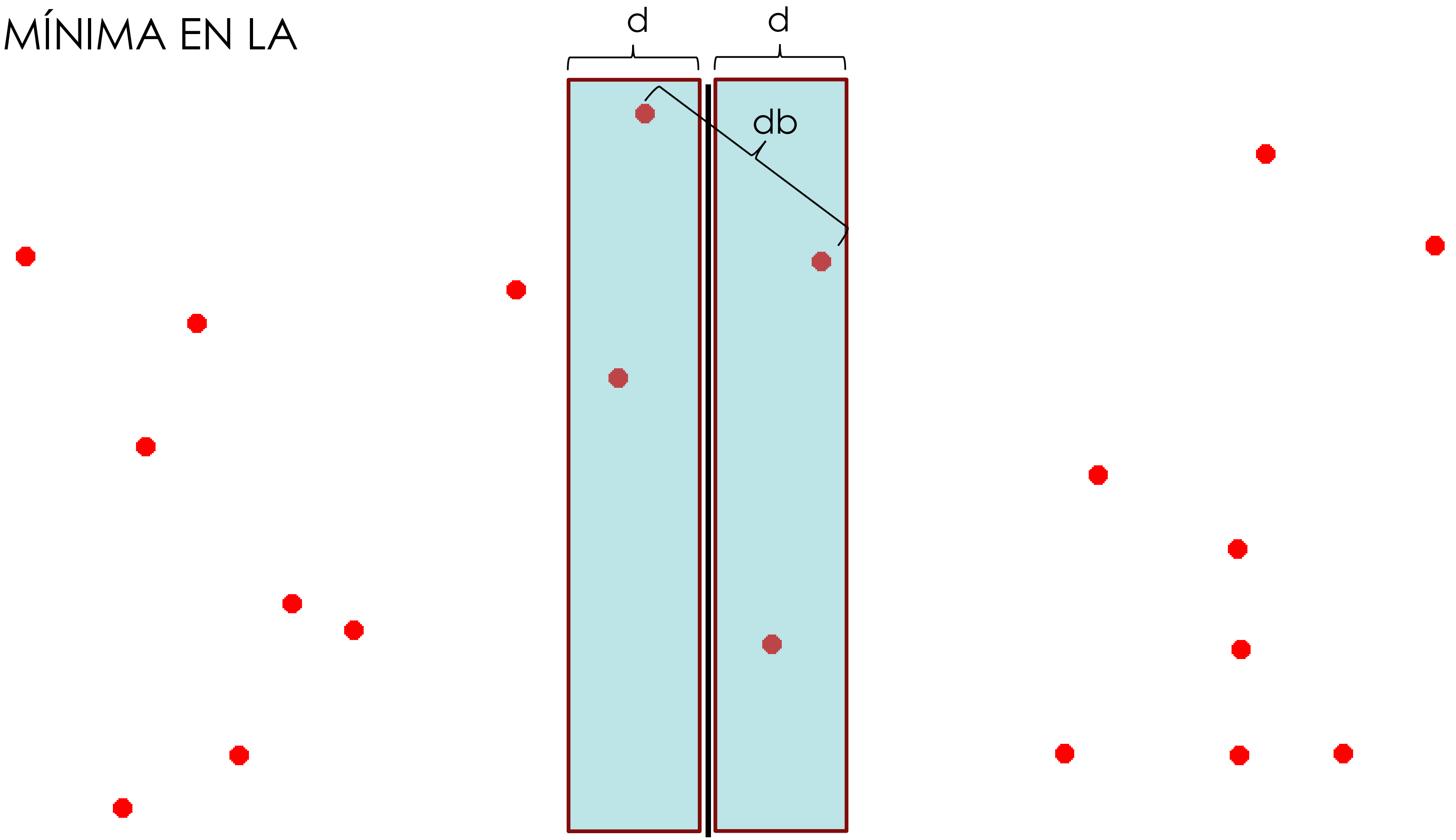


CÁLCULO DE LA DISTANCIA MÍNIMA EN CADA LADO



d=min(d1,d2)

DISTANCIA MÍNIMA EN LA BANDA

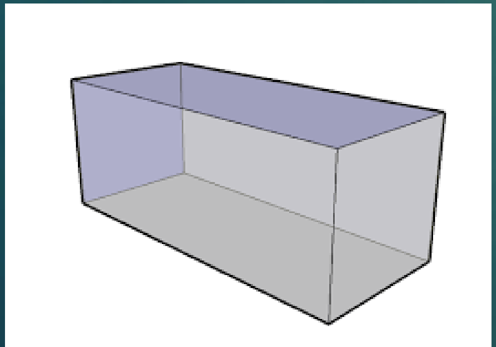


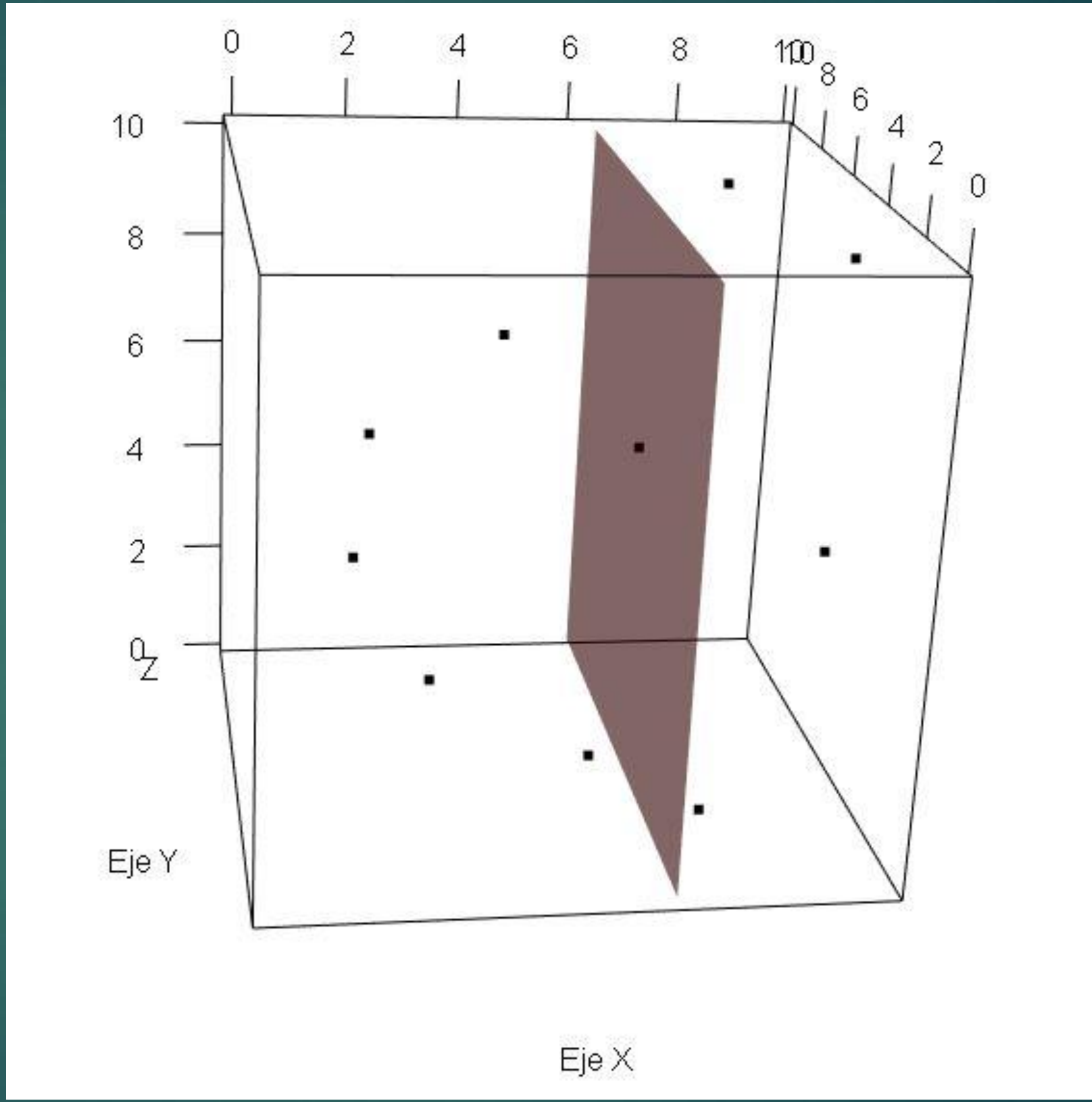
dTotal=min(d,db)

CASO DE 3 DIMENSIONES

La división se hace mediante un plano.

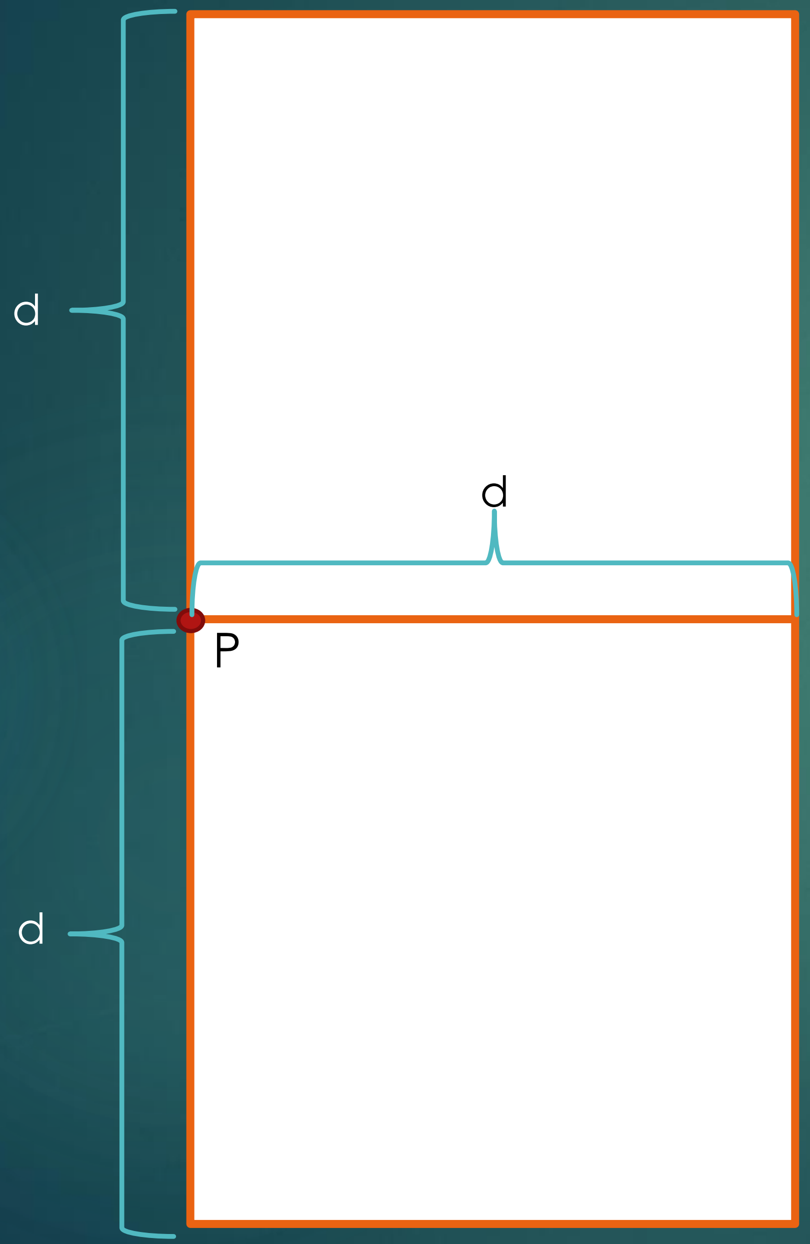
La banda es un ortoedro.



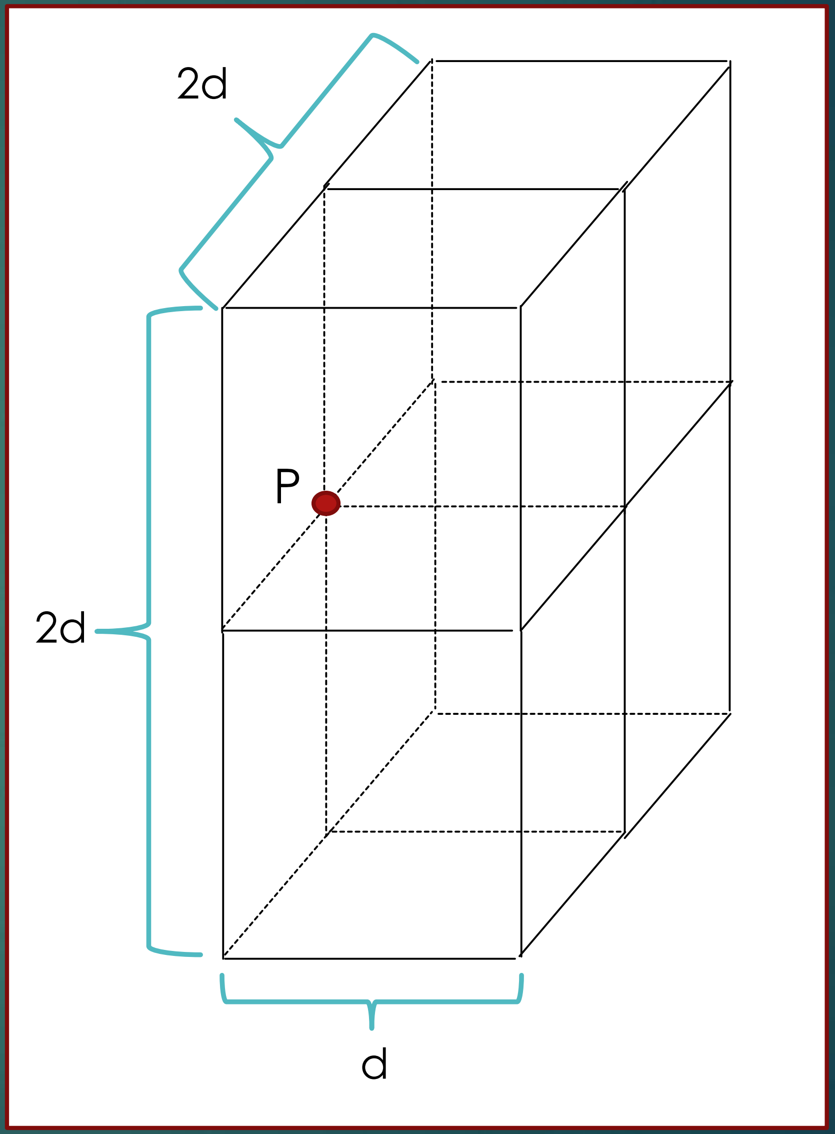


CÁLCULO DE LA DISTANCIA ENTRE LOS PUNTOS DE LA BANDA

2D



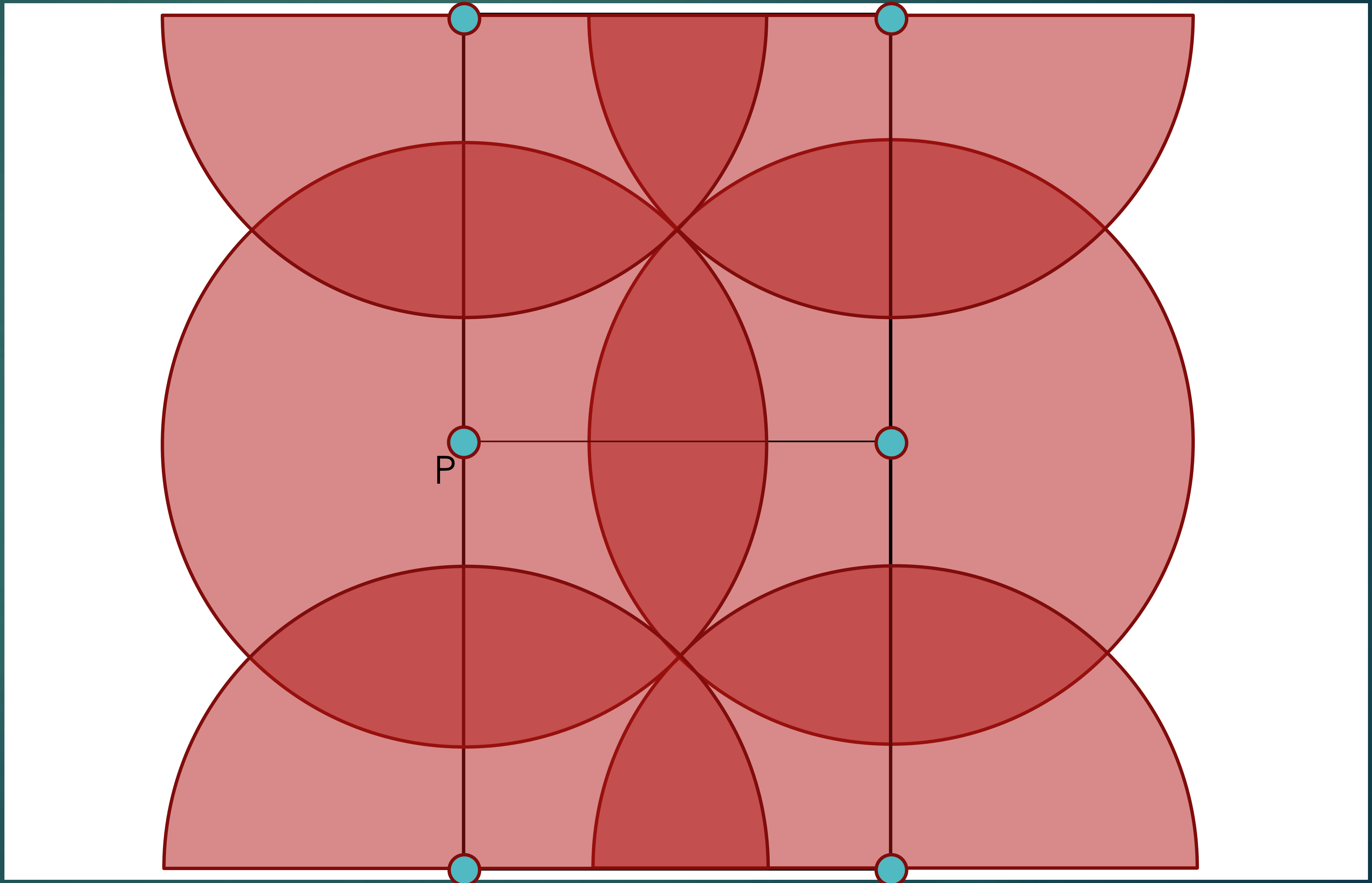
3D



DEMOSTRACIÓN

Este el caso de 2 dimensiones.

Para el caso de 3 dimensiones se formarían 18 esferas.



PSEUDOCÓDIGO

```
puntos_x = vector de puntos ordenados por la coordenada x
puntos_y = vector de puntos ordenados por la coordenada y

función puntos_cercanos (puntos_x, puntos_y)
        if n=1
                return infinito
        elseif 2<=n<=3
                return caso_base(puntos)
        elseif n>=4
                med=mediana de la coordenada x de puntos
                s1_x = vector de puntos en el lado izquierdo ordenados por x
                s1_y =  vector de puntos en el lado izquierdo ordenados por y
                s2_x = vector de puntos en el lado derecho ordenados por x
                s2_y = vector de puntos en el lado derecho ordenados por y
                d1=puntos_cercanos(s1_x, s1_y)
                d2=puntos_cercanos(s2_x, s2_y)
                d=min(d1,d2)
                candidatos = par de puntos cuya distancia es d
                db=dist_banda(puntos_y, med, d)
                dtotal=min(d,db)
                candidatos = par de puntos cuya distancia es dtotal
                return dTotal y candidatos
```

PSEUDOCÓDIGO

```
función dist_banda(puntos,mediana,dist)
        for each i in puntos
                banda = vector de puntos p tal que (mediana-dist < p.x < mediana+dist)
        if 2D
                for i from 0 to length(banda)
                        for j from i+1 to i+6
                                if j<length(banda) & dist < |i – j|
                                        dist = |i-j|
                                        Actualizar candidatos
        elseif 3D
                for i from 0 to length(banda)
                        for j from i+1 to i+18
                            if j<length(banda) & dist < |i – j| & j.z pertenece a (i.z-d, i.z+d)
                                        dist = |i-j|
                                        Actualizar candidatos

        return dist y candidatos
```

COSTE

```
puntos_x = vector de puntos ordenados por la coordenada x
puntos_y = vector de puntos ordenados por la coordenada y

función puntos_cercanos (puntos_x, puntos_y)
        if n=1
                return infinito
        elseif 2<=n<=3
                return caso_base(puntos)
        elseif n>=4
                med=mediana de la coordenada x de puntos
                s1_x = vector de puntos en el lado izquierdo ordenados por x
                s1_y = vector de puntos en el lado izquierdo ordenados por y
                s2_x = vector de puntos en el lado derecho ordenados por x
                s2_y = vector de puntos en el lado derecho ordenados por y
                d1=puntos_cercanos(s1_x, s1_y)
                d2=puntos_cercanos(s2_x, s2_y)
                d=min(d1,d2)
                candidatos = par de puntos cuya distancia es d
                db=dist_banda(puntos_y, med, d)
                dtotal=min(d,db)
                candidatos = par de puntos cuya distancia es dtotal
                return dTotal y candidatos
```

O(nlog(n))

O(cte)

O(n)

?

O(cte)

O(cte)

COSTE

```
función dist_banda(puntos,mediana,dist)
    for each i in puntos
        banda = vector de puntos p tal que (mediana-dist < p.x < mediana+dist)
    if 2D
        for i from 0 to length(banda)
            for j from i+1 to i+6
                if j<length(banda) & dist <|i – j|
                    dist = |i-j|
                    Actualizar candidatos
    elseif 3D
        for i from 0 to length(banda)
            for j from i+1 to i+18
                if j<length(banda) & dist <|i – j| & j.z pertenece a (i.z-d, i.z+d)
                    dist = |i-j|
                    Actualizar candidatos

    return dist y candidatos
```

O(n)

O(5n)=O(n)

O(17n)=O(n)

RESOLUCIÓN DE LA RECURSIVIDAD

MÉTODO MAESTRO

$T(n)=a*T(n/b)+f(n)$

a = 2 nº de subproblemas
b = 2 n/b es el tamaño de cada subproblema
f(n) = coste fuera de la recursión

CASO 2: si para algún k>=0, se cumple que $f(n)=O(n^c*\log^k(n))$ donde $c=\log_b a$

En este problema, $f(n)=O(n)$
Por lo tanto, si k= 0, $c=\log_2 2=1$ y por tanto, $f(n) = O(n^1*\log^0(n)) = O(n)$.
Para el caso 2, la ecuación de la recursividad se resuelve de la forma:

$T(n)=O(n^c * \log^{k+1}(n))$ Sustituyendo c y k obtenemos que la recursividad tiene un coste de $O(n*\log(n))$.

El único coste que no se tiene en cuenta en el método maestro es el coste de ordenación, pero dado que era $O(n*\log(n))$, igual que la recursividad, el coste total del algoritmo es $O(n*\log(n))$.