Il problema della coppia più vicina

Problema: vogliamo trovare la coppia di punti più vicina tra un insieme di punti del piano.

La distanza tra due punti $p_1 = (x_1, y_1)$ e $p_2 = (x_2, y_2)$ si calcola con la formula $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2}$ in tempo $O(1)$

Il problema può essere risolto in tempo $O(n^2)$ calcolando le distanze tra tutte le coppie di punti.

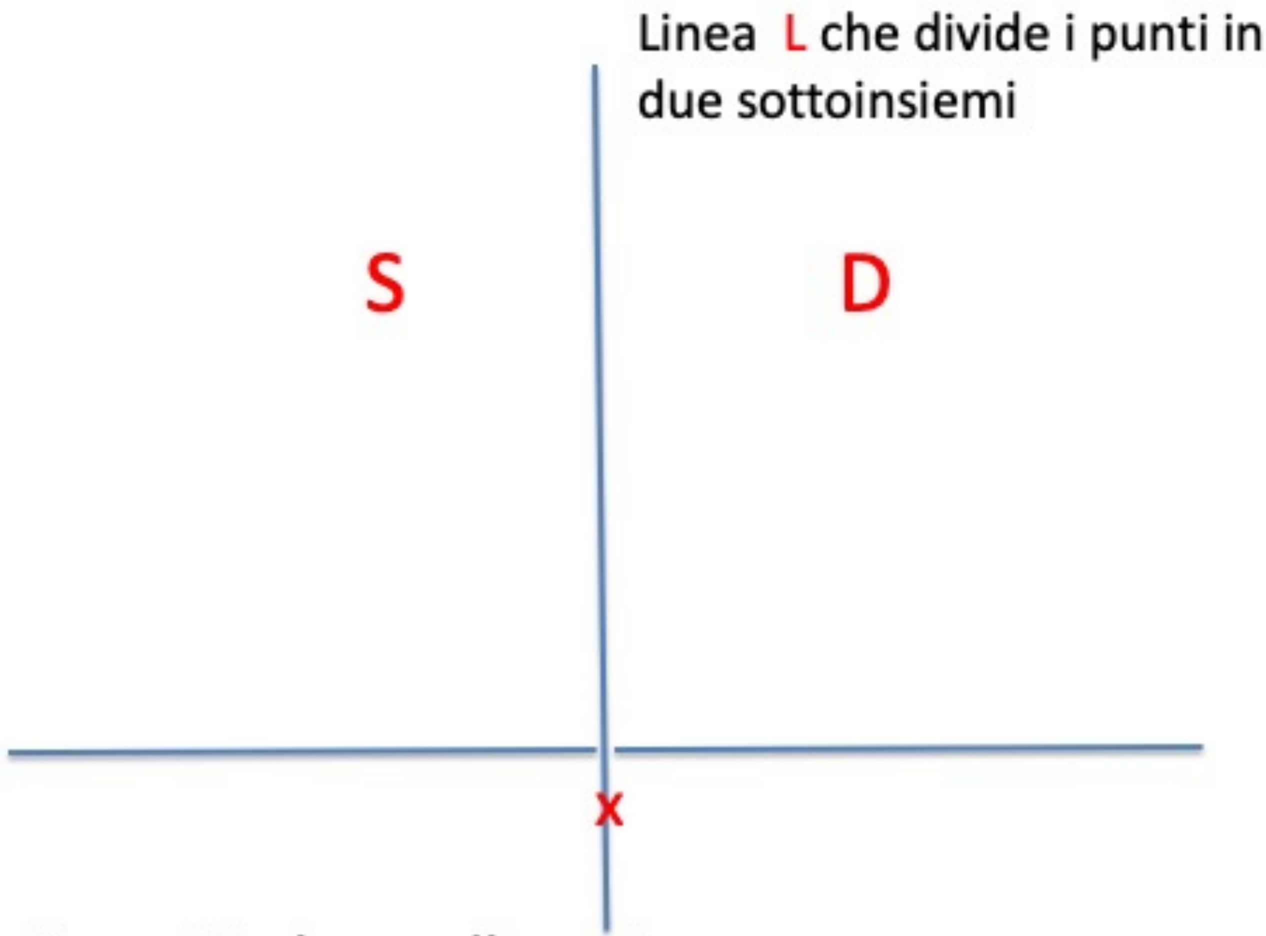
Utilizzando la tecnica del divide et impera, il problema può essere risolto in tempo $O(n \log n)$.

Il problema della coppia più vicina

Idea intuitiva.

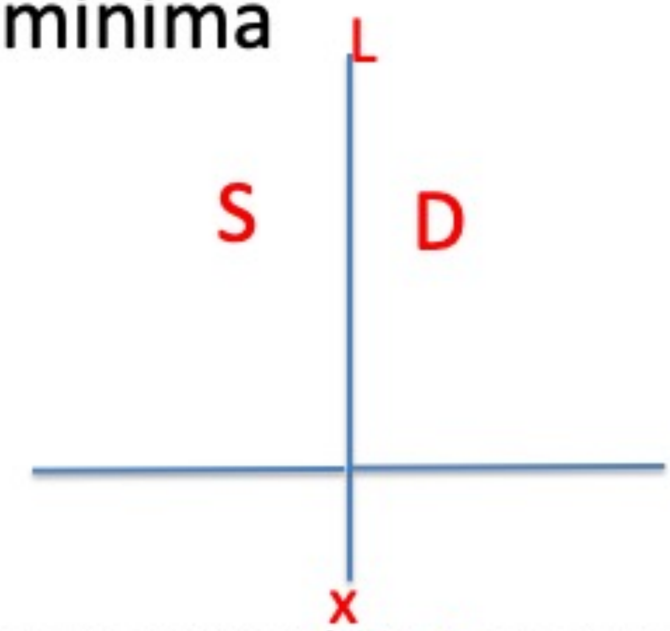
- l'insieme ha cardinalià minore o uguale di una certa costante: usiamo la ricerca esaustiva.
- altrimenti: lo dividiamo in due parti uguali S e D, per esempio quelli a sinistra e quelli a destra di una fissata linea verticale
  - troviamo ricorsivamente le soluzioni per S e quella per D individuando due coppie di punti a distanza minima, d_S e d_D
- soluzione finale: o una delle due coppie già individuate oppure può essere formata da un punto in S e uno in D
- se d_{SD} è la minima distanza tra punti aventi estremi in S e D, la soluzione finale è data dalla coppia di punti a distanza $\min\{d_{SD}, d_S, d_D\}$.

Partizione in due sottoinsiemi di n/2 punti ciascuno



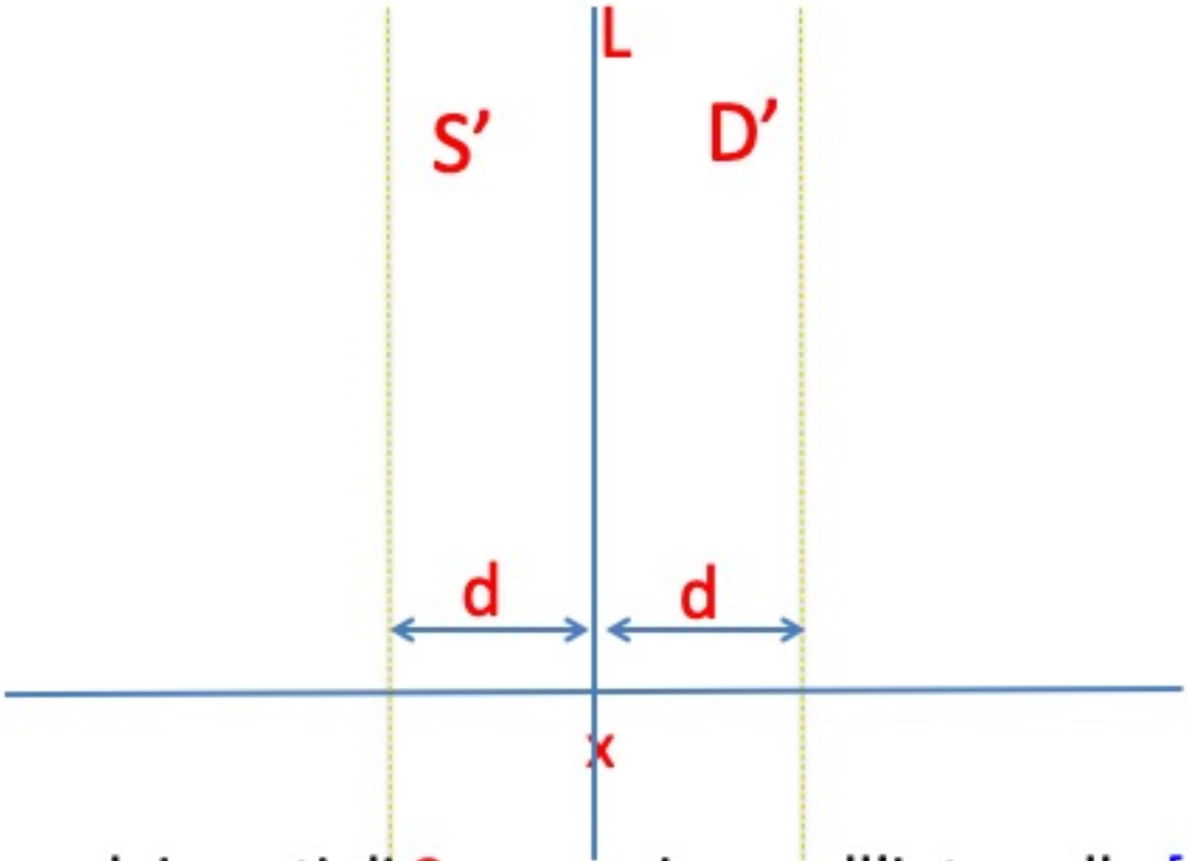
- Ordina i punti in base alle ascisse
- x= ascissa punto **pc** centrale nell'ordinamento
- S= insieme dei punti a sinistra di **pc** nell'ordinamento
- D= insieme dei punti a destra di **pc** nell'ordinamento

Individuazione della coppia di punti a distanza minima



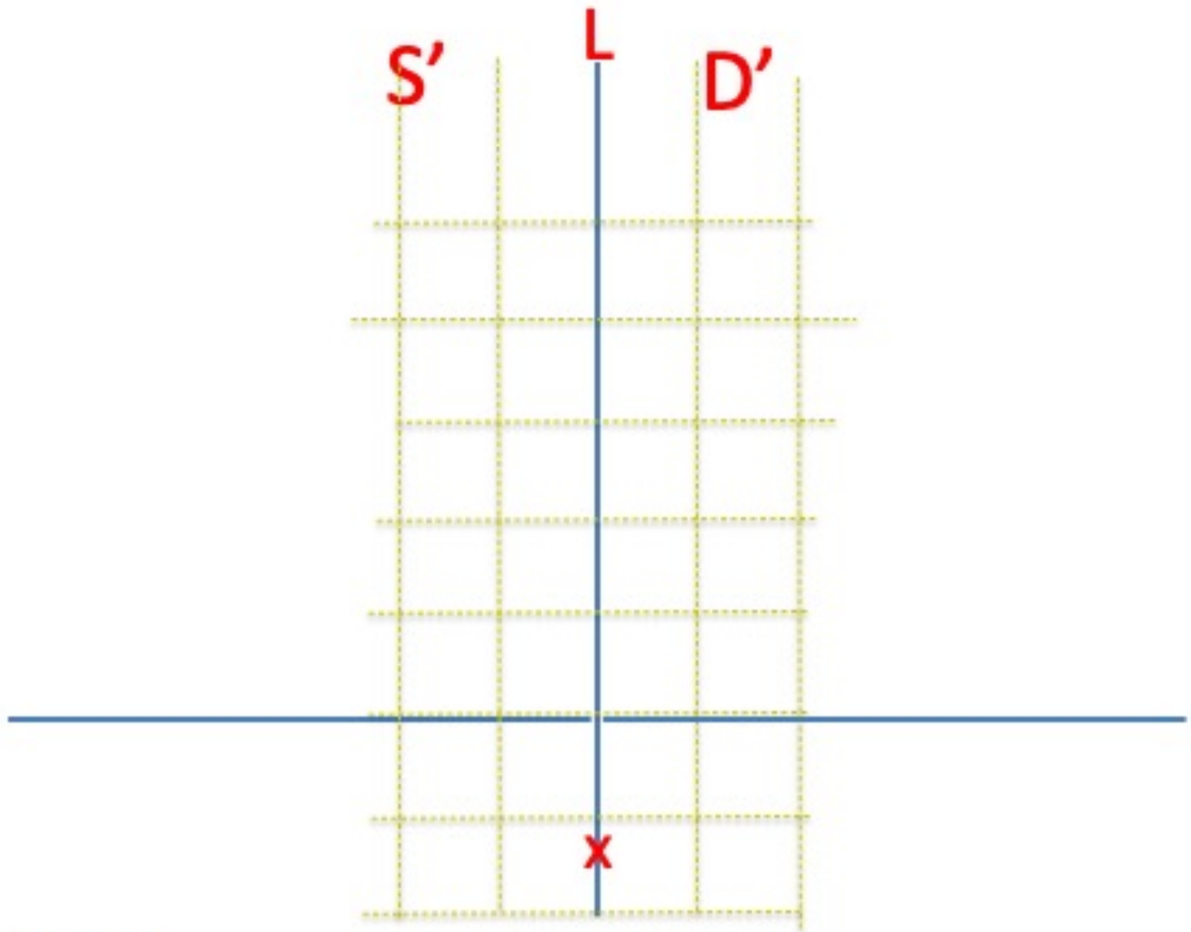
- I 2 punti a distanza minima o sono entrambi in S, o sono entrambi in D, o uno dei due si trova in S e l'altro in D
- Divide et impera:
- Decomposizione: partiziona l'insieme di punti in S e D
- Soluzione sottoproblemi: cerca la coppia a distanza minima d_S in S e la coppia a distanza minima d_D in D. $d=\min\{d_S, d_D\}$
- Ricombinazione: Cerca tra le coppie (p,q) con p in S e q in D quella a distanza minima d_{SD} e nel far questo ignora le coppie che evidentemente sono a distanza maggiore di d. Alla fine restituisce la coppia con distanza pari a $\min\{d, d_{SD}\}$.

Ricerca della coppia (p,q) a distanza minima con p in S e q in D



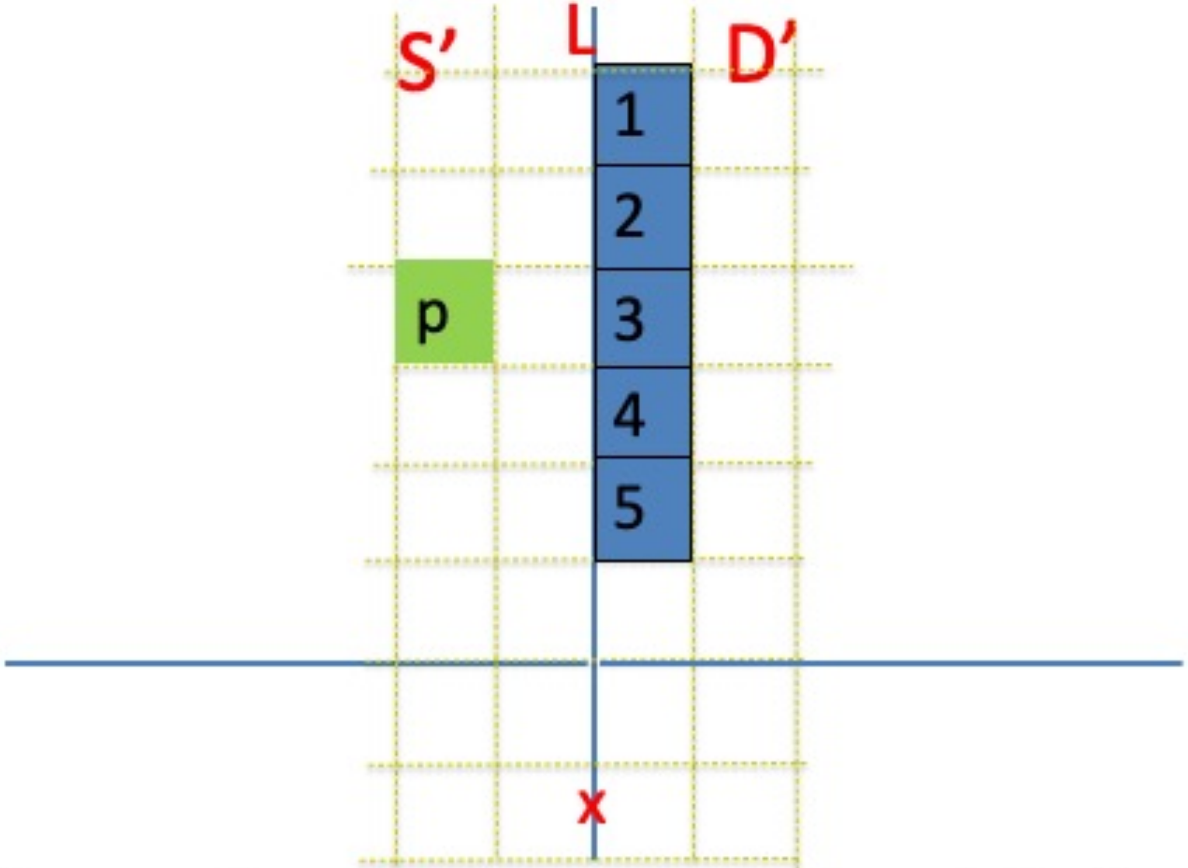
- S’= insieme dei punti di S con ascissa nell’intervallo [x-d,x]
- D’= insieme dei punti di D con ascissa nell’intervallo [x,x+d]
- è sufficiente considerare coppie (p,q) con p in S’ e q in D’ ’ in quanto le altre coppie (p,q) con p in S e q in D sono a distanza maggiore di d

Dividiamo S’ e D’ in tanti quadrati di lato uguale a d/2



- Osservazione 1: Ciascun quadrato contiene un unico punto altrimenti esisterebbe una coppia di punti entrambi in S’ o entrambi in D’, a distanza minore di d

Dividiamo S’ e D’ in tanti quadrati di lato uguale a d/2

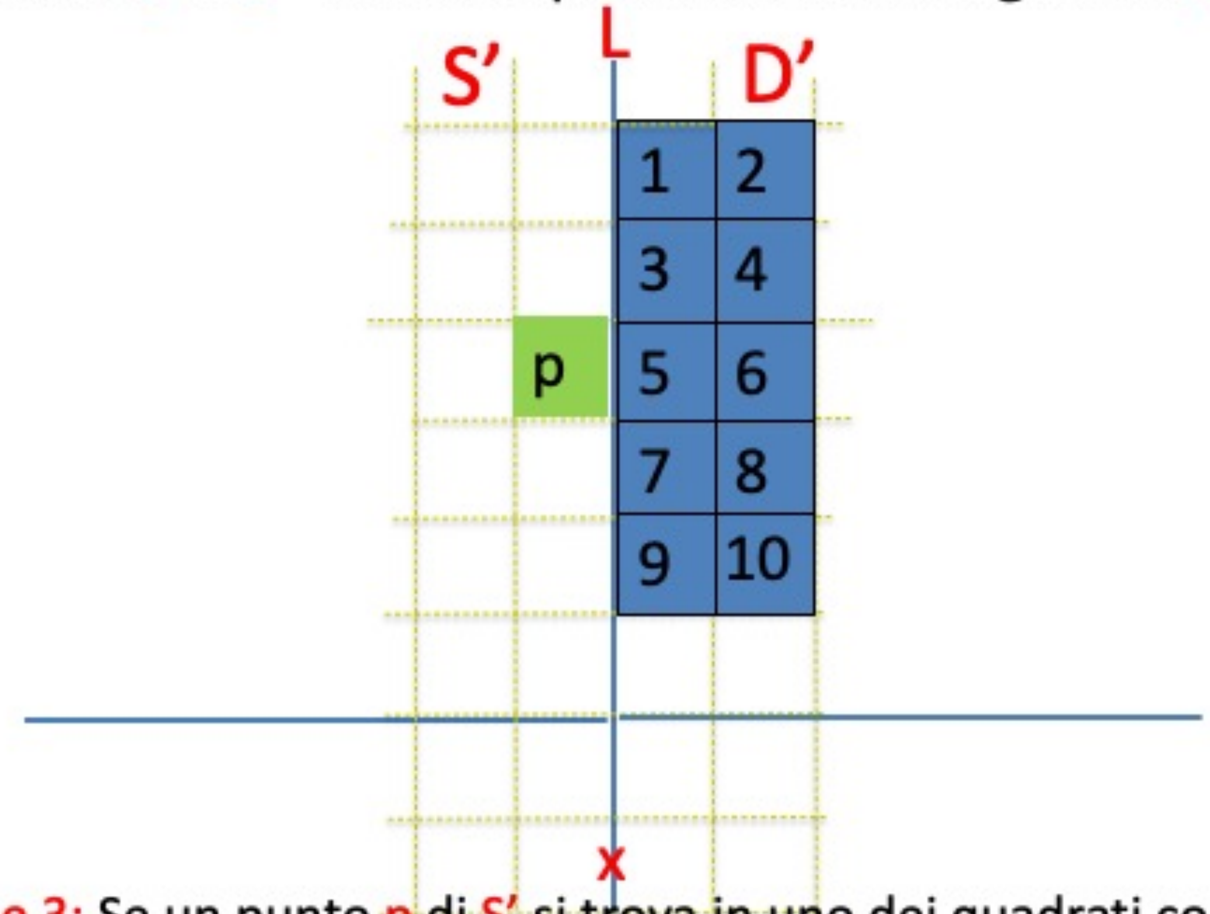


Osservazione 2: Se un punto p di S’ si trova in uno dei quadrati più a sinistra allora i punti di D’ a distanza minore di d da p possono trovarsi solo nei quadrati di D’ confinanti con L e in particolare in 5 di questi quadrati, in quello alla stessa altezza del quadrato contenente p, nei 2 quadrati al di sopra di questo e nei due al di sotto.

Ad esempio se p è nel quadrato verde, allora un punto q di D’ a distanza minore di d da p può trovarsi solo in uno dei quadrati in D’ colorati di azzurro

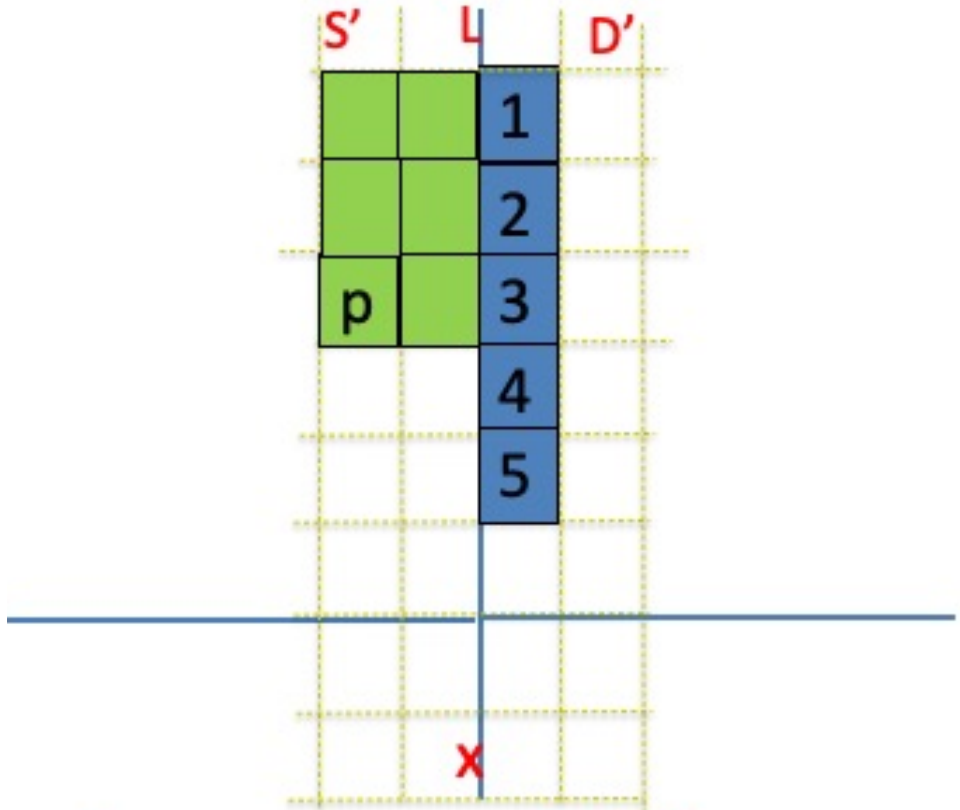
Se q è più in alto rispetto a p allora q si trova in uno dei quadrati 1, 2, 3; altrimenti si trova ir uno dei quadrati 3, 4, 5.

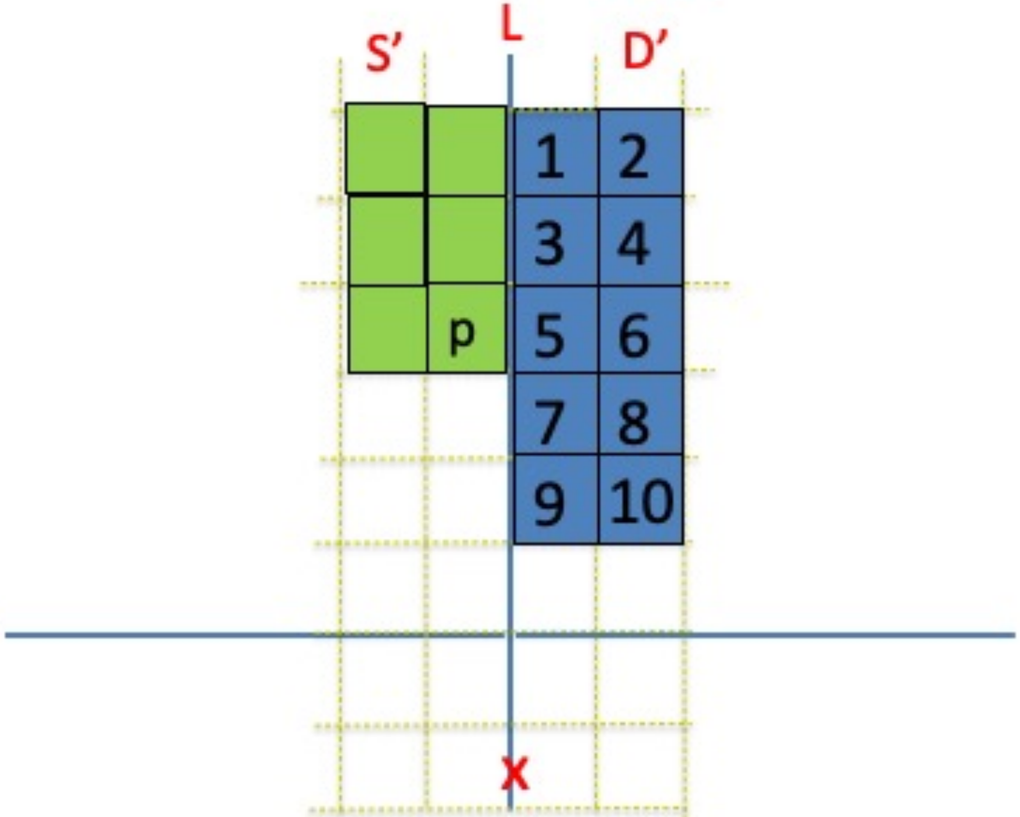
Dividiamo S’ e D’ in tanti quadrati di lato uguale a d/2



- Osservazione 3: Se un punto p di S’ si trova in uno dei quadrati confinati con L allora i punti di D’ a distanza minore di d da p possono trovarisi solo nei due quadrati di D’ alla stessa altezza di quello contenente p o nei quattro quadrati al di sopra di questi due quadrati o nei quattro al di sotto.
- Se p si trova nel quadrato verde allora un punto q di D’ a distanza minore di d da p può trovarsi solo in uno dei 10 quadrati in D’ colorati di azzurro
  - Se q è più in alto rispetto a p allora q si trova in uno dei quadrati 1-6; altrimenti q si trova in uno dei quadrati 5-10

Dividiamo S’ e D’ in tanti quadrati di lato uguale a d/2





- P_d = array dei punti di S’ e D’ in ordine non decrescente di altezza
- per ogni punto p in P_d cerchiamo il punto a distanza minima da p tra quelli più in alto di p
- Ciascun quadrato contiene al più 1 punto → un punto q di D’ a distanza al più d da p si trova al più 11 locazioni in avanti nell’array P_d rispetto a p
  - tra p e q possono esserci infatti al più 5 punti di D’ e 5 punti di S’ (II figura): ad esempio se q è più in alto rispetto a p allora tra p e q può esserci al più un punto di D’ per ciascuno dei quadrati 1-6 (meno quello contenente q) e un punto di S’ per ciascuno dei quadrati verdi (meno quello contenente p)

L'algoritmo che trova la coppia più vicina

Input: P_x = array dei punti ordinato in modo non decrescente rispetto alle ascisse; P_y = array dei punti ordinato in modo non decrescente rispetto alle ordinate, n dimensione degli array P_x e P_y

1. Se $n <= 3$, calcola le distanze tra le tre coppie di punti per trovare la coppia a distanza minima.
2. Se $n > 3$, esegue i seguenti passi:
3. Inserisce nell'array S_x i primi $\lfloor n/2 \rfloor$ punti di P_x e nell'array D_x gli ultimi $\lceil n/2 \rceil$ punti di P_x
4. Inserisce nell'array S_y i primi $\lfloor n/2 \rfloor$ punti di P_x nell'ordine in cui appaiono in P_y e nell'array D_y gli ultimi $\lceil n/2 \rceil$ punti di P_x nell'ordine in cui appaiono in P_y
5. Effettua una chiamata ricorsiva con input S_x, S_y e $\lfloor n/2 \rfloor$ e una chiamata ricorsiva con input D_x, D_y e $\lceil n/2 \rceil$. Siano d_S e d_D i valori delle distanze delle coppie di punti restituite dalla prima e dalla seconda chiamata rispettivamente. Pone $d = \min\{d_S, d_D\}$ e (p, q) uguale alla coppia a distanza d.
6. Copia in P_d i punti a distanza minore di d dalla retta verticale passante per l'elemento centrale di P_x nello stesso ordine in cui appaiono in P_y
7. Per ciascun punto p' in P_d esamina gli 11 punti che seguono p' in P_d; per ciascun punto q' (tra questi 11) computa la sua distanza da p' e se questa risulta minore di d, aggiorna il valore di d e pone $(p, q) = (p', q')$
8. Restituisce la coppia (p, q)

Analisi del costo dell'algoritmo che trova coppia più vicina

Assumiamo per semplicità che n sia un potenza di 2

1. Se n è $<=3$, il costo è limitato superiormente da una certa costante c_0
2. Se n è >3, il costo dell'algoritmo è così computato:
3. il costo del passo 3 è $O(n)$
4. il costo del passo 4 è $O(n)$: i punti di P_y vengono scanditi a partire dalla prima locazione.e vengono man mano inseriti in S_y o in D_y a seconda che si trovino in locazioni di P_x di indice minore di $\lfloor n/2 \rfloor$ oppure in locazioni di P_x di indice maggiore o uguale di $\lfloor n/2 \rfloor$
5. Il costo delle due chiamate ricorsive è $2T(n/2)$; il costo delle altre operazioni eseguite al passo 5 è costante
6. Il passo 6 richiede tempo $O(n)$: i punti di P_y vengono scanditi a partire dalla prima locazione e quelli la cui ascissa differisce al più d dall'ascissa dell'elemento centrale di S_x vengono man mano inseriti in P_d
7. il passo 7 richiede tempo $O(n)$ perché P_d contiene al più n punti e per ciascuno di essi vengono computate al più 11 distanze, 11 confronti e 11 aggiornamenti di d, p e q.
8. il passo 8 richiede tempo $O(1)$

Costo computazionale dell'algoritmo per la coppia più vicina definito mediante relazione di ricorrenza

$$T(n) \leq \begin{cases} c_0 & \text{se } n \leq 2 \\ 2T\left(\frac{n}{2}\right) + cn & \text{altrimenti} \end{cases}$$

dove c_0, c sono costanti.

Abbiamo $T(n) = O(n \log n)$.

```
CoppiaPiuVicina(Px,Py,n):
IF(n<=3){Return RicercaEsaustiva(Px,Py,n);}
 ELSE{ p=Px[n/2];}
j=k=0;
FOR(i=0;i<n/2;i=i+1){
Sx[i]=Px[i]; Dx[i]=Px[i+n/2];}
if n%2==1 Dx[n-1]=Px[n-1];
FOR(i=0;i<n;i=i+1){
IF(Py[i].x<=p.x){Sy[j]=Py[i]; j=j+1;}
ELSE {Dy[k]=Py[i]; k=k+1;}
}
(ps,qs)= CoppiaPiuVicina(Sx,Sy,n/2);
(pd,qd)=CoppiaPiuVicina(Dx,Dy,(n+1)/2);
IF(Dist(ps,qs)<Dist(pd,qd)){d=Dist(ps,qs); (p,q)=(ps,qs);}
ELSE {d=Dist(pd,qd); (p,q)=(pd,qd);}
FOR(i=m=0;i<n;i=i+1){
 IF(|Py[i].x-p.x|<=d){Pd[m]=Py[i]; m=m+1;}
}
FOR(i=0;i<m;i=i+1){
  FOR(j=i+1;j<=min{i+11,m};j=j+1){
     IF(Dist(Pd[i],Pd[j])<d){ d=Dist(Pd[i],Pd[j]); (p,q)=(Pd[i],Pd[j]);}
 }
}
RETURN(p,q);
}
```