#Se asume que S está ordenado por x, S2 por y.

6.

```
Closest-points (S, low, high, S2):
    if(low = high) return INT_MAX

    else if(high-low = 1) return dist(S, low, high) //Euclidean distance

    mid = ⌊(low+high)/2⌋

    S_low = Closest-points(S, low, mid)

    S_high = closest_point(S, mid+1, high, S2)

    S_across = across(s_low, s_high, S, low, high, S2)

    return min(s_low, s_high, s_across)
```

```
Across (s_low, s_high, S, S2, low, high):
    min_value = min(S_low, S_high) , values = [ ]
    for i = low to high:
        if(|S2[i].x - S[mid].x|) < min_value:
            values.push(S2[i])

    copy = min_value
    for i to values.size:
        temp = i+1
        while temp < values.size and (values[temp].y - values[i].y)
                                    < min_value:

                min_value = dist(values[i], values[temp])
                temp += 1

    return min(copy, min_value)
```

El time complexity es

$$T(n) = 2T(n/2) + \underbrace{O(n)}_{\text{across}}$$

por master method

$n^{\log_2 2} = n$

$f(n) = n$

$\boxed{n = n}$

$$\Theta(n \lg n)$$