9.3.5

Divide into $\frac{n}{5}$ groups

Let median subroutine = median(Array) $\Rightarrow O(N)$ time

Black-Box: Median subroutine, divides into medians

```
Select(A, p, r, i)
  If p = r
    return A[p]
  x = median(A)
  q = partition(A, x)
  k = q - p + 1
  if i = k then
    return A[q]
  else if i < k
    return select(A, p, q-1, i)
  else return select(A, q+1, r, i-k)
```

get median of A

$T(n) \leq T(\frac{n}{2}) + O(n)$

$= d \sum_{i=0}^{\log n} \frac{n}{2^i}$

$= n + \frac{n}{2} + \frac{n}{4} + \dots + 1$

$= O(n)$

$\sum_{i=1}^{\infty} (\frac{1}{2})^i \quad \frac{\frac{1}{2}}{1-\frac{1}{2}}$

$\frac{\frac{1}{2}}{\frac{1}{2}} = \frac{2}{2} = 1$

$n \rightarrow O(n)$

$\frac{n}{2} \rightarrow O(\frac{n}{2})$

$\frac{n}{4} \rightarrow O(\frac{n}{4})$

$\vdots$

$T(1)$

9.3-7

have an Array A of n distinct numbers
positive integer $k \leq n$

median will be $\frac{n}{2}$ least order statistic
Find $\frac{n}{2} - \frac{k}{2}$ and $\frac{n}{2} + \frac{k}{2}$ order statistics

Find order statistic $(\frac{n-k}{2})$, then partition all lesser elements to the left. Then find $\frac{(n+k)}{2}$ and find the greater elements then partition them to the right of $(\frac{n+k}{2})$ order statistic. Now all elements inbetween should be the k closest medians (Between $\frac{n-k}{2}$ and $\frac{n+k}{2}$ th order statistic)

Find k medians closest

```
X = Select(A, 5, n, (n-k)/2)
partition(A, X) // Partition to the left side
X = Select(A, 5 - (n-k)/2, n, (n+k)/2)
partition(A, X) // Partition to the right side
...
return A[(n-k)/2] → A[(n+k)/2] except for A[n/2]
```

9.3-8

Steps:

1) Find median of both arrays $\Rightarrow O(1)$
2) If medianA = medianB we are done $\Rightarrow O(1)$
3) If medianA = medianB return median $\Rightarrow O(1)$
4) If medianA > medianB, we know median is in $\rightarrow T(\frac{n}{2})$
   $A[0 \rightarrow \frac{n}{2}]$ and $B[\frac{n}{2} \rightarrow n]$
5) Vice versa for medianB > medianA $\rightarrow T(\frac{n}{2})$
6) Return if both subarrays are of size $1 \Rightarrow O(1)$

```
Median(A, B, n) {
  if n=0        // invalid
    return -1
  if n=1        // choose lower median
    return min(A[0], B[0])
  //
  //
  //
  medianA = A[n/2]
  medianB = B[n/2]

  if(medianA > medianB)
    return Median(A+n/2, B, n-n/2)
  else
    return Median(B+n/2, A, n-n/2)
}
```

39.3-4) Because the vertices are already in counterclockwise order, we can just do a Graham's scan without sorting for a convex hull problem. Graham's scan without pre-sorting provides $O(N)$ time if and only if the vector/array is already sorted.

33-1 convex layers

a) Use Jarvis march for finding each layer.
First, take an array A of size N, then find the h vertices it takes to construct a layer.
Reiterate while the current size $N_c \neq 0$, get all layers i.

Let h be the total # of vertices on each layer
Let h_i be the number of vertices on the ith layer
$N = h_1 + h_2 + \dots + h_i$

$T(N) = T(N - h_i) + O(Nh_i)$ #of layers = i

# of layers

$Nh_1 \rightarrow O(Nh_i)$
$\downarrow$
$(N-h_1)h_2 \rightarrow O((N-h_1)h_2)$
$\downarrow$
$(N-h_1-h_2)h_3 \rightarrow O((N-h_1-h_2)h_3)$
$\downarrow$
$(N-h_1-\dots-h_{i-1})h_i \rightarrow O((N-h_1-\dots-h_{i-1})h_i)$

$T(N) = Nh_1 + (N-h_1)h_2 + \dots + (N-h_1-\dots-h_{i-1})h_i$
$= Nh_1 + Nh_2 - h_1h_2 + \dots + Nh_i - h_ih_i - \dots - h_i - 1$
$= \underbrace{N(h_1 + h_2 + \dots h_i)}_{N} \underbrace{- h_1h_2 - \dots - (h_1h_i + \dots h_{i-1}h_i)}_{\text{constant}}$
$= N(N) - O(1)$
$= N^2 - O(1)$
$\boxed{T(N) = O(N^2)}$

b) Prove lowerbound is $\Omega(n \lg n)$

model of computation that requires $\Omega(n \lg n)$ time to sort n real numbers

- On any unsorted set of integers, we have only a best time of $O(n \lg n)$ to sort these numbers. (Merge sort, Heap sort)
- For each layer, we only need to use $O(N)$ time to find the convex hull. if we have a sorted list.
- Because convex hull depends on N elements, its lower bound is $\Omega(N)$
- Therefore, since we can solve the convex layers in $O(N)$ on a sorted list, $\Omega(N \lg N)$ is the lower bound of complexity on a sorted list.

33.4-2) We are looking for all points that are lesser than the current minimum distance δ. That is, for some δ', $\delta' < \delta$. There is an emphasis here on the strictly lesser than. If it were less than or equal, then there could be a worst case of every point on the line. However, since we only need to check what is strictly within a rectangle of size $\delta \times 2\delta$ then there are only 6 points that can exist within the box. Therefore, it will take only the 5 points following the element in Y'.