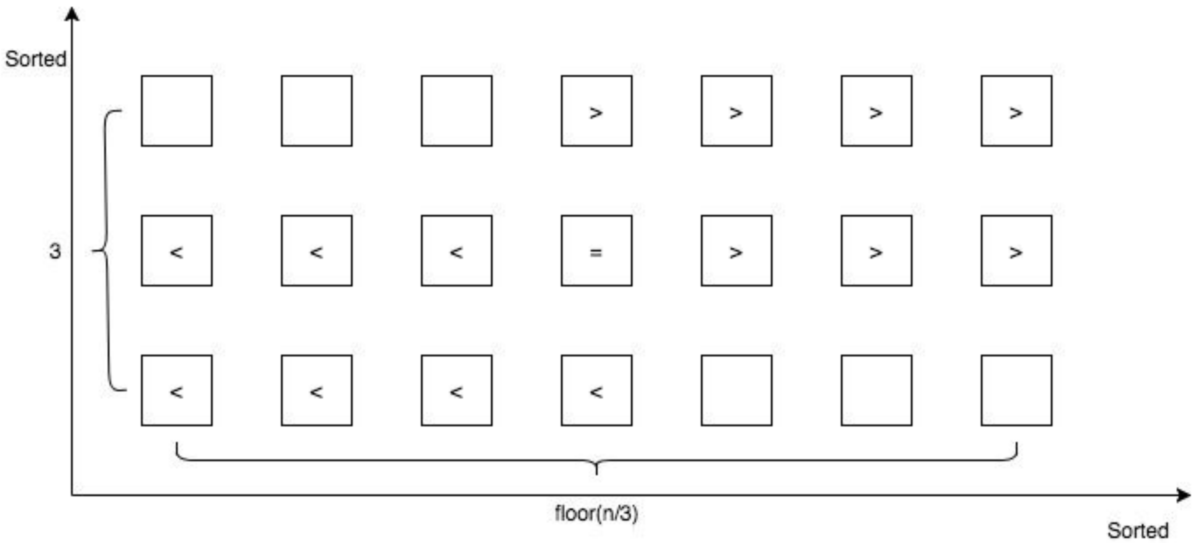
1) DSELECT with group of 3 elements:

Algorithm:

function DSelect(k, S)
    if |S| ≤ 50
        then sort and return the kth smallest element in S
    choose an element m∗ from S:

    divide S into ⌊|S|/3⌋ groups of 3 elements each
        (there may be up to 2 leftover elements)
    find the median of each group, by sorting and taking middle el’t
    m∗ ← the median of the set of ⌊|S|/3⌋ group medians

    distribute elements from S into sets L, E, G:
        L = {el’ts < m∗}, E = {el’ts = m∗}, G = {el’ts > m∗}
    if k ≤ |L| then return DSelect(k, L)
    else if k ≤ |L| + |E| then return m∗
    else return DSelect(k − |L| − |E|, G)



Upper Bound on |L| and |G|

The number of elements which will be in G can be calculated as $\lceil \lceil n/3 \rceil /2 - 2 \rceil \times 2$ which is basically counting the number of elements in G in the above given picture.

So there are at least these many elements greater than median.
Similarly, the number of elements in L will be $\lceil \lceil n/3 \rceil /2 - 2 \rceil \times 2$

Now, $\lceil \lceil n/3 \rceil /2 - 2 \rceil \times 2 \geq (n/3-4)$

So, the total number of elements on which we will call DSELECT recursively will be atmost :
$n - (n/3 - 4) = (2n/3 + 4)$
Let $T(n)$ = time taken by DSelect on set of size n.

Cost of each step:
• dividing into groups of 3 — no time
• sort each group and find medians — $\theta(1) * \lfloor n / 3 \rfloor$ groups
• compute m∗ using recursive call — $T(\lfloor n / 3 \rfloor)$
• partition S into L, E, G — $\theta(n)$
• recursive call — $\leq T(2n/3 + 4)$
So, $\mathbf{T(n) \leq T(n/3) + T(2n/3 + 4) + c_1 n}$ **, for** $\mathbf{n \geq 50}$

Also,
For n<50,
• Let $I = \{1, 2, \ldots, 49\}$
• Let $j \in I$ be index that maximizes the ratio $T(i)/i$
• Let $c_2 = T(j)/j$, then $\forall\ i \in I$, $T(i)/i \leq c_2$
• Therefore, $\mathbf{T(n) \leq c_2 n}$**, for n<50**

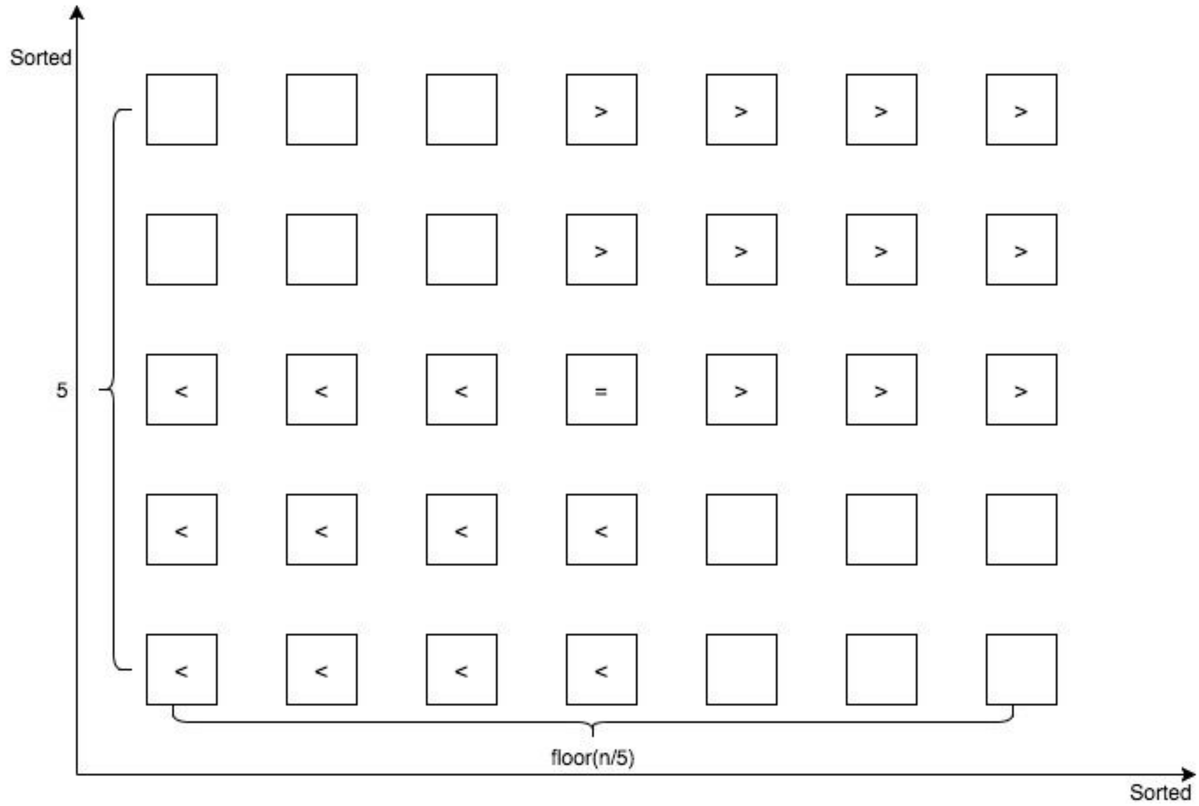
Now, $T(n) \leq T(n/3) + T(2n/3 + 4) + c_1 n$, for $n \geq 50$
And $T(n) \leq c_2 n$, for $n < 50$
which solves to $T(n) =$ **O(n log n)**

2) DSELECT with group of 5 elements:

Algorithm:
```
function DSelect(k, S)
        if |S| ≤ 50
                then sort and return the kth smallest element in S
        choose an element m∗ from S:

        divide S into ⌊|S|/5⌋ groups of 5 elements each
                (there may be up to 4 leftover elements)
        find the median of each group, by sorting and taking middle el't
        m∗ ← the median of the set of ⌊|S|/5⌋ group medians

        distribute elements from S into sets L, E, G:
                L = {el'ts < m∗}, E = {el'ts = m∗}, G = {el'ts > m∗}
        if k ≤ |L| then return DSelect(k, L)
        else if k ≤ |L| + |E| then return m∗
        else return DSelect(k − |L| − |E|, G)
```



Upper Bound on |L| and |G|

- Half of the $\lfloor n/5 \rfloor$ groups contain at least 3 elements $\geq m*$
- None of these $3\lfloor n/10 \rfloor$ elements can be in L so $|L| \leq n - 3\lfloor n/10 \rfloor$
- Letting $n = 10k + r \Rightarrow |L| \leq (10k + r) - 3k = 7k + r$ $7k + r = .7(10k + r) + .3r \leq 7n/10 + 2.7$ (because $r \leq 9$)
- $|L|$ is integer $\Rightarrow |L| \leq \lfloor 3n/4 \rfloor$, for $n \geq 50$
- Symmetric reasoning gives $|G| \leq \lfloor 3n/4 \rfloor$, for $n \geq 50$

Let T (n) = time taken by DSelect on set of size n.

Cost of each step:
- dividing into groups of 5 — no time
- sort each group and find medians — $\theta(1) * \lfloor n / 5 \rfloor$ groups
- compute $m*$ using recursive call — $T(\lfloor n / 5 \rfloor)$
- partition S into L, E, G — $\theta(n)$
- recursive call — $\leq T(3n/4)$

So, $\mathbf{T(n) \leq T(n/5) + T(3n/4) + c_1 n}$ **, for n ≥ 50**

Also,
For n<50,
- Let $I = \{1, 2, ..., 49\}$
- Let $j \in I$ be index that maximizes the ratio $T(i)/i$
- Let $c_2 = T(j)/j$, then $\forall\ i \in I$, $T(i)/i \leq c_2$
- Therefore, $\mathbf{T(n) \leq c_2 n}$**, for n<50**

Now, $T(n) \leq T(n/5) + T(3n/4) + c_1 n$, for $n \geq 50$
And $T(n) \leq c_2 n$, for $n < 50$
which solves to T(n) = **O(n)**

Because for T(n) <= cn + T(an) + T(bn), where a+b<1, the total time is c(1/(1-a-b))n
So, Here the total time will be **20cn**, where c = max{c1,c2}

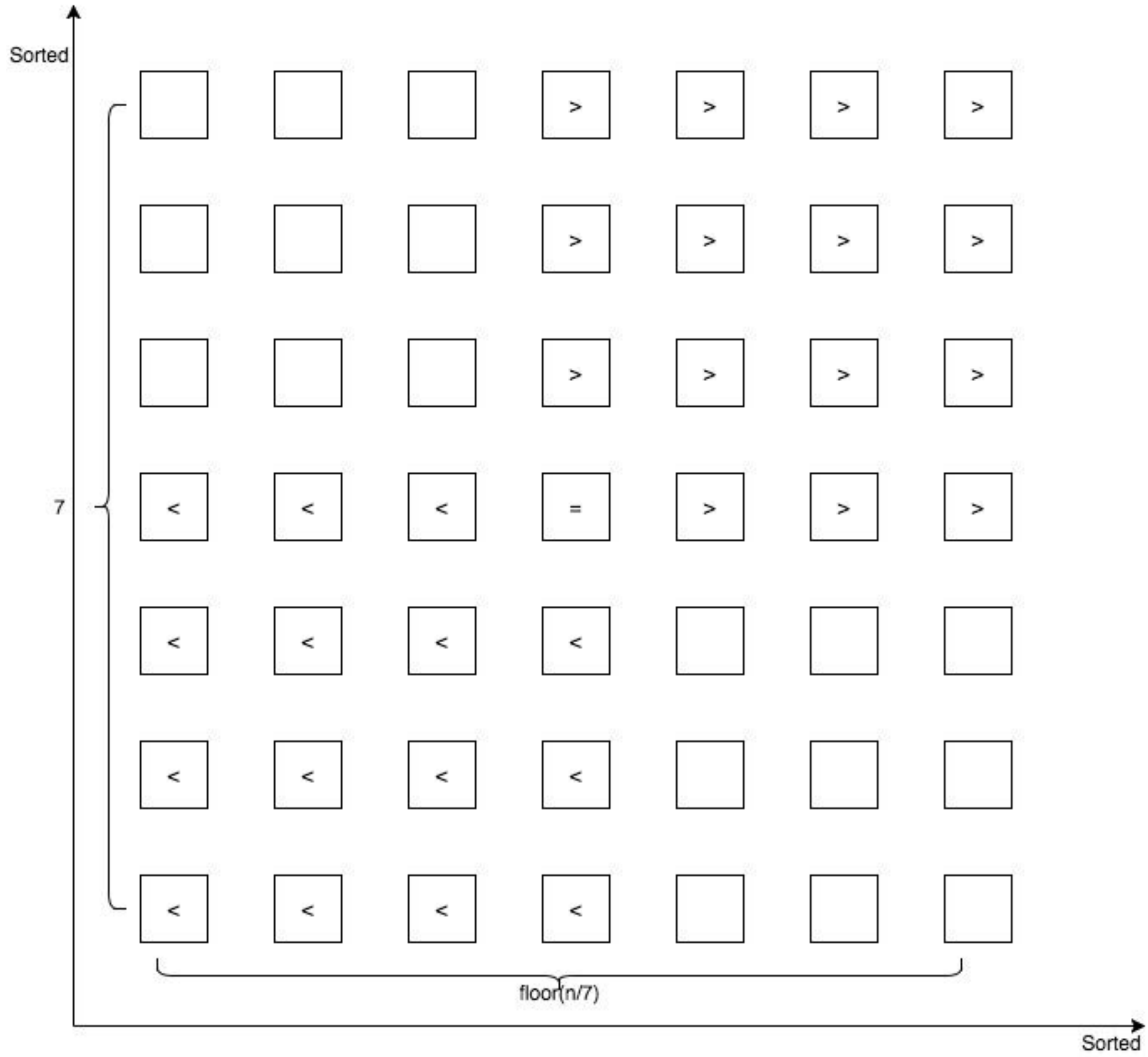
3) DSELECT with group of 7 elements:

Algorithm:

```
function DSelect(k, S)
        if |S| ≤ 50
                then sort and return the kth smallest element in S
        choose an element m∗ from S:

        divide S into ⌊|S|/7⌋ groups of 7 elements each
                (there may be up to 6 leftover elements)
        find the median of each group, by sorting and taking middle el't
        m∗ ← the median of the set of ⌊|S|/7⌋ group medians

        distribute elements from S into sets L, E, G:
                L = {el'ts < m∗}, E = {el'ts = m∗}, G = {el'ts > m∗}
        if k ≤ |L| then return DSelect(k, L)
        else if k ≤ |L| + |E| then return m∗
        else return DSelect(k − |L| − |E|, G)
```



Upper Bound on |L| and |G|

- Half of the $\lfloor n/7 \rfloor$ groups contain at least 4 elements $\geq m*$
- Ignoring the sublist containing the pivot and the last one that can contain at most 6 elems
- The number of elements which will be in G can be calculated as $4(\lceil 1/2 \times \lceil n/7 \rceil\rceil) - 2$
- Similarly, the number of elements in L will be $4(\lceil 1/2 \times \lceil n/7 \rceil\rceil) - 2$

Now, $4(\lceil 1/2 \times \lceil n/7 \rceil\rceil) - 2 \geq (2n/7 - 8)$

So, the total number of elements on which we will call DSELECT recursively will be atmost :
$n - (2n/7 - 8) = (5n/7 + 8)$

Let T (n) = time taken by DSelect on set of size n.

Cost of each step:
• dividing into groups of 7 — no time
• sort each group and find medians — $\theta(1) * \lfloor n / 7 \rfloor$ groups
• compute m∗ using recursive call — $T (\lfloor n / 7 \rfloor)$
• partition S into L, E, G — $\theta (n)$
• recursive call — $\leq T (5n/7 + 8)$

So, $\mathbf{T (n) \leq T (n/7) + T(5n/7 + 8) + c_1 n}$**, for** $\mathbf{n \geq 50}$

Also,
For n<50,
• Let I = { 1, 2, . . . , 49 }
• Let $j \in I$ be index that maximizes the ratio T(i)/i
• Let $c_2 = T(j)/j$, then $\forall\ i \in I, T(i)/i \leq c_2$
• Therefore, $\mathbf{T (n) \leq c_2 n}$**, for n<50**

Now, $T (n) \leq T (n/7) + T(5n/7 + 8) + c_1 n$, for $n \geq 50$
And $T(n) \leq c_2 n$, for $n < 50$
which solves to T(n) = O(n)

Because for T(n) <= cn + T(an) + T(bn), where a+b<1, the total time is c(1/(1-a-b))n
So, Here the total time will be **7cn**, where c = max{c1,c2}

Description of the Test-cases:

There are 2 types of test-cases:

1. To get worst-case for QuickSelect, we have used a sorted input with 30,000 elements. To simulate the worst-case for QuickSelect, we are always choosing the last element as the median rather than getting it from random number generator. Then we ran the test-case for all four selection algorithms.
2. For average case, We generated a random input of 30,000 elements and ran the four selection algorithms on it. Here, in QuickSelect, we used the random number generator to select the median.

Comparative analysis and conclusions drawn:

According to our theoretical analysis, we have found following results:

1. DSELECT with group of 3 elements has O(n logn) (Worst Case)
2. DSELECT with group of 5 elements has O(n) (Worst Case) with 20cn time
3. DSELECT with group of 7 elements has O(n) (Worst Case) with 7cn time
4. QuickSelect has O(n) with 8c'n time (Average Case) and $O(n^2)$ (Worst Case)

So, the time taken should be in the following order if we assume that the QuickSelect takes average case time and is not hitting its worst case.

QuickSelect ~ DSELECT(7 elements) < DSELECT(5 elements) < DSELECT(3 elements)

The output obtained by running the four selection algorithm is as follows and it aligns with our analysis results where time taken is in the order:
DSelect (7 elements) < DSelect (5 elements) < DSelect(3 elements)

```
******** AVERAGE CASE  ********
Input is an unsorted array with size: 30000
Picking the kth element, K = 3678

QuickSelect- Output: 210.156, Time taken: 0
DSelect (groups of 3)- Output: 210.156, Time taken: 0.104
DSelect (groups of 5)- Output: 210.156, Time taken: 0.076
DSelect (groups of 7)- Output: 210.156, Time taken: 0.052
```

If we consider that QuickSelect hits its worst case, then the order will be following:

DSELECT(7 elements) < DSELECT(5 elements) < DSELECT(3 elements) < QuickSelect

The output obtained by running the four selection algorithm is as follows and it aligns with our analysis results where time taken is in the order:
DSelect (7 elements) < DSelect (5 elements) < DSelect(3 elements) < QuickSelect

```
******** WORST CASE ********
Input is a sorted array (descending order) with size: 30000
Picking the kth element, K = 3678

QuickSelect- Output: 209767, Time taken: 2.184
DSelect (groups of 3)- Output: 209767, Time taken: 0.084
DSelect (groups of 5)- Output: 209767, Time taken: 0.06
DSelect (groups of 7)- Output: 209767, Time taken: 0.036
```