```
function selectionSort(array):
  for i = 0 to size - 1
    minIndex = i
    for j = i + 1 to size
      if array[j] < array[minIndex]
        minIndex = j
    swap array[i] with array[minIndex]
```
BEST TC = n^2
WORST TC = n^2

```
function QUICKSORT(ARRAY, START, END)
  if START >= END then
    return
  end if
  PIVOTINDEX = PARTITION(ARRAY, START, END)
  QUICKSORT(ARRAY, START, PIVOTINDEX – 1)
  QUICKSORT(ARRAY, PIVOTINDEX + 1, END)
end function
```
BEST TC = n log n
WORST TC = n^2

```
function binary_search(list, target):
  left = 0
  right = length(list) - 1
  while left <= right:
    mid = (left + right) // 2
    if list[mid] == target:
      return mid
    elif list[mid] < target:
      left = mid + 1
    else:
      right = mid - 1
  return -1
```
BEST CASE:1, WC: log n

```
procedure insertionSort( A : array of items )
  int holePosition
  int valueToInsert
  for i = 1 to length(A) inclusive do:
    valueToInsert = A[i]
    holePosition = i
    while holePosition > 0 and A[holePosition-1] > valueToInsert do:
      A[holePosition] = A[holePosition-1]
      holePosition = holePosition -1
    end while
    A[holePosition] = valueToInsert
  end for
end procedure
```
BEST TC = n ; WORST TC = n^2

1 to n, $i{+}{=}2$: $\frac{n}{2}$ steps : $\Theta(n)$
1 to n, $i{+}{=}3$: $\frac{n}{3}$ steps "
n to 1, $i{-}{=}6$: $\frac{n}{6}$ steps "
1 to n, $i{*}{=}3$: $\log_3 n$ steps $\Theta(\log n)$
1 to n, $i/{=}3$: $\log_3 n$ steps $\Theta(\log n)$

$$A\cdot P \Rightarrow \sum_{i=1}^{k} i = 1+2+3+4+\cdots+k = \frac{k(k+1)}{2}$$

$$H\cdot N \Rightarrow \sum_{i=1}^{k} = \frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{k} = \log k = \Theta(\log k)$$

$$G\cdot P \Rightarrow \sum_{i=0}^{k} x^i = x^0+x^1+x^2+\cdots+x^k = \frac{x^{k+1}-1}{x-1}$$

$$\infty\ G\cdot P \Rightarrow \text{when } |x|<1 \text{ . eg } x=\frac{1}{2} ; \sum_{i=0}^{k} = \frac{1}{1-x}$$

$\lim_{n\to\infty} \frac{f(n)}{g(n)}$
a) $0 \Rightarrow f(n) \in o(g(n))$
b) $c \neq 0 \Rightarrow f(n) \in \Theta(g(n))$
c) $\infty \Rightarrow g(n) \in o(f(n))$

```
function MERGESORT(ARRAY, START, END)
  if END - START + 1 == 1 then
    return
  end if
  if END - START + 1 == 2 then
    if ARRAY[START] > ARRAY[END] then
      TEMP = ARRAY[START]
      ARRAY[START] = ARRAY[END]
      ARRAY[END] = TEMP
    end if
    return
  end if
  HALF = int((START + END) / 2)
  MERGESORT(ARRAY, START, HALF)
  MERGESORT(ARRAY, HALF + 1, END)
  MERGE(ARRAY, START, HALF, END)
end function
```
TC = n log n

$T(n) = aT(n/b) + f(n)$ has the following asymptotic bound

b^(log_n b)=n

1) If $f(n) \in O(n^{\log_b a-\varepsilon})$ for some $\varepsilon>0$, then $T(n)\in\Theta(n^{\log_b a})$

2) If $f(n) \in \Theta(n^{\log_b a})$, then $T(n) \in \Theta(n^{\log_b a}\cdot \log n)$

3) If $f(n) \in \Omega(n^{\log_b a+\varepsilon})$ for some $\varepsilon>0$ and if $a\cdot f(n/b) \leq c\cdot f(n)$ for $c<1$, then $T(n)\in\Theta(f(n))$

O-notation (big-oh) : $f(n)\in O(g(n)) \Leftrightarrow \exists c>0, \exists n_0>0, \forall n \geq n_0\ f(n) \leq c\cdot g(n)$
Intuition: $f(n)$ is $O(g(n))$ if the asymptotic growth of $f(n)$ is at most the asymptotic growth of $g(n)$

Ω-notation (omega) : $f(n)\in\Omega(g(n)) \Leftrightarrow \exists c>0, \exists n_0>0, \forall n\geq n_0\ f(n)\geq c\cdot g(n)$
Intuition: $f(n)$ is $\Omega(g(n))$ if the asymptotic growth of $f(n)$ is at least the asymptotic growth of $g(n)$

Θ-notation (theta) : $f(n)\in O(g(n))$ and $f(n)\in\Omega(g(n))$

a) $4n^5-50n^2+10n \in \Theta(n^5)$

First : $4n^5-50n^2+10n \in O(n^5)$
$4n^5-50n^2+10n \leq 4n^5+10n^5 \leq 14n^5$
$4n^5-50n^2+10n \leq 4n^5+10n^5$, for all $n\geq1$
$\therefore f(n) \leq c\cdot g(n)$ for $c=14$

Second : $4n^5-50n^2+10n \in \Omega(n^5)$
$4n^5-50n+10n \geq 4n^5-50n^2 \geq n^5+3n^5-50n^2 \geq n^4$
$4n^5-50n+10n \geq$ $(3n^5-50n^2\geq0$ when $n=3)$
$\therefore f(n)\geq c\cdot g(n)$ for $c=1$ and $n_0=3$

```
function PARTITION(ARRAY, START, END)
  PIVOTVALUE = ARRAY[END]
  PIVOTINDEX = START
  loop INDEX from START to END
    if ARRAY[INDEX] <= PIVOTVALUE
      TEMP = ARRAY[INDEX]
      ARRAY[INDEX] = ARRAY[PIVOTINDEX]
      ARRAY[PIVOTINDEX] = TEMP
      PIVOTINDEX = PIVOTINDEX + 1
    end if
  end loop
  return PIVOTINDEX – 1
```

P[E] = (1/6)*1+(5/6)*0
Ev[X=5]=(1/6*5)

Base Case : $T(1) = d'$. $cn\log n = c\cdot1\cdot\log 1 = 0$ ✗
$T(2) = 2T(1) + d2 = 2d' + d2 = 2(d'+d)$.
$c\cdot2\cdot\log 2 = 2c$. True when $c\geq d'+d$

Inductive step: Assume $\boxed{T(k) \leq c\cdot k\cdot\log k}$ for all $k<n$.
$T(n) = 2\boxed{T(\frac{n}{2})} + dn \leq 2\boxed{c\frac{n}{2}\log\frac{n}{2}} + dn$
$\log n - \log 2$, $\log n - 1$
$= cn(\log n - 1) + dn$
$= \underbrace{cn\log n}_{\text{desired}} \underbrace{- cn + dn}_{\text{residual}}$

Inductive steps: Initialization; iteration (before and after same); termination

1. Big oh Notation – Use the def^n of Bigoh & Omega to show that $4n^2-8n+6 \in \Theta(n^2)$

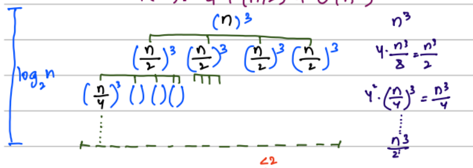
$O : 4n^2-8n+6 \leq 4n^2+6 \leq 4n^2+6n^2 = 10n^2$

$f(n)\leq c\cdot g(n)$ for $c=10\ \forall\ n\geq n_0=1$

$\Omega : 4n^2-8n+6 \geq 4n^2-8n = n^2+(3n^2-8n) \geq n^2$
when $3n^2-8n\geq0 \Rightarrow n\geq\frac{8}{3}$

$f(n)\geq c\cdot g(n)$ for $c=1\ \forall\ n\geq n_0=3$

2. Recursion Tree – use recursion tree to generate a guess of the asymptotic complexity of a divide & conquer algorithm with the running time
$T(n) = 4T(n/2) + O(n^3)$



$\sum_{i=0}^{\log n}\frac{n^3}{2^i} = n^3\sum_{i=0}^{\log n}(\frac{1}{2})^i < n^3\sum_{i=0}^{\infty}\overbrace{(\frac{1}{2})^i}^{<2} < 2n^3$

3. Induction
let $T(n) = 3T(n/3)+dn$ with $T(1)=a$ for constants d and a. Use induction to prove that $T(n) \in O(n\log_3 n)$

Base case : $n=3$ $T(3)=3T(1)+3d \Rightarrow 3a+3d \Rightarrow 3(a+d)$

Inductive step Assume $T(k) < c\cdot k\log_3 k\ \forall\ k<n$
$T(n) = 3T(\frac{n}{3})+dn \leq 3\left[c\frac{n}{3}\log_3\frac{n}{3}\right]+dn$
$= cn\left[\log_3 n - \log_3 3\right] + dn$
$= cn\log_3 n - cn + dn$
$\rightarrow \leq cn\log_3 n$ (when $c\geq d$)

$\Rightarrow T(n)\leq c\cdot n\log_3 n$ for $c=a+d\ \forall\ n\geq3$.

4. Master Method
Solve the following recurrences using the master method. Justify your answers shortly (i.e., specify ε and check the regularity condition if necessary).

(i) $T(n) = 2T(n/2) + n^2$
$a=2$, $b=2$, $f(n)=n^2$, $n^{\log_b a} = n^{\log_2 2} = n$!
$n^2\in\Omega(n^{1+\varepsilon})$ for $\varepsilon=1$
$af(\frac{n}{b}) = 2(\frac{n}{2})^2 = \frac{n}{2} = cn$ for $c=\frac{1}{2}$. case 3 holds
$T(n)\in\Theta(n^2)$

(ii) $T(n) = 27T(n/3)+n^3$
$a=27$ $b=3$ $f(n)=n^3$ $n^{\log_3 27} = n^3$
case 2 holds
$\Rightarrow T(n)\in\Theta(n^3\log n)$

(iii) $T(n) = 9T(n/3) + n\log n$
$a=9$ $b=3$ $f(n)=n\log n$ $n^{\log_b 9} = n^2$
$n\log n \in O(n^{2-\varepsilon})$ for $\varepsilon=\frac{1}{2}$ NOTE Any $\varepsilon\in(0,1)$ works
case 1 holds $\varepsilon=1$ doesnot work
$T(n)\in\Theta(n^2)$

$T(n) = aT(n/b)+f(n)$ has the following Asymptotic bounds
(i) If $f(n)\in O(n^{\log_b a-\varepsilon})$ for some $\varepsilon>0$, then $T(n)\in\Theta(n^{\log_b a})$
(ii) If $f(n)\in\Theta(n^{\log_b a})$, then $T(n)\in\Theta(n^{\log_b a}\cdot\log n)$
(iii) If $f(n)\in\Omega(n^{\log_b a+\varepsilon})$ for some $\varepsilon>0$, and if $a\cdot f(n/b)\leq c\cdot f(n)$ for a $c<1$, then $T(n)\in\Theta(f(n))$
regularity cond^n

5. Randomized Analysis
Suppose someone offers to let you play a game. They will randomly (and independently) roll three dice: a red die, a green die, and a blue die. If you choose to play, you will pay $5 to enter the game, and you will try to guess the number that is rolled for each of the three dice (note you submit a guess for each color, e.g. red = 5, green = 2, and blue = 1). If you guess exactly one of the three correctly, then they will give you $1 back. If you guess exactly two out of three correctly, they will give you $5 back. If you guess all three correctly, they will give you $100 back. What is the expected value of this game from your perspective?

$X(s) = \{-4, 0, 95, -5\}$
miss 2, miss 1, miss 0, miss 3

$E(X) = P(X=-4)\cdot-4 + \underbrace{P(X=0)\cdot0}_{=0} + P(X=95)\cdot95 + P(X=5)\cdot-5$

$P(X=95) = \frac{1}{6}\times\frac{1}{6}\times\frac{1}{6} = (\frac{1}{6})^3$
$P(X=-5) = \frac{5}{6}\times\frac{5}{6}\times\frac{5}{6} = (\frac{5}{6})^3$
$P(X=-4) = \frac{1}{6}\times\frac{5}{6}\times\frac{5}{6}\times3 = \frac{75}{6^3}$
Prob of getting first 1 & missing last two
3 ways to prove the die you got correct.

$E(X) = 95(\frac{1}{6})^3 + -5(\frac{5}{6})^3 + -4(\frac{75}{6^3})$ dont need to simplify on an exam

5. Divide and Conquer
Suppose we have a two-dimensional array of (positive or negative) integers A[][] such that each each row and column is monotonically increasing. That is, if you consider a row and scan from left to right, the numbers increase. Similarly, if you scan a column from bottom to top, the numbers increase. We are interested in finding the largest number in the array that is less than 0. Give a divide and conquer algorithm which runs in o(n) time (i.e. strictly faster than linear time), where n is the total number of integers in A.

Idea :- check # in middle. If <0, then answer cannot be in lower left quadrant (all numbers here are less than middle value)
similarly, if it is ≥0 then answer cannot be in upper right (all numbers here are greater than middle)

If middle <0, we call 3 subproblems | If middle ≥0, we call 3 subproblems

Rec Search (A, rb, rt, cl, cr) where rt, rb are the top & bottom rows of our subproblem. cl & cr are the left & right coloumns of our subprob.
```
{
  if (rt-rb== 1 or cr-cl==1) {
     binary search that coloumn or row for 0 and
     then return the largest #<0. If all are
     ≥0, return -∞
  }
  mid low = 1/2 ; mid col = m/2 ;
  middle = A[mid Row][mid Col];
  if (middle < 0) {
     upleft = RecSearch (A, midRow+1, rt, cl, midcol);
     downRight = RecSearch(A, rb, midrow, midcol+1, cr);
     upRight = RecSearch(A, midrow, ct, midcol, cr);
     Return Max(upleft, downRight, upRight);
  }
  else {
     upleft = RecSearch(A, midRow, rt, cr, midCol-1);
     downRight = RecSearch (A, rb, midRow-1, midCol, cr);
     bottom left = RecSearch(A, rb, midRow, cr, midCol);
     Return max(upleft, downRight, bottom left);
  }
}
```
$T(n) = 3T(\frac{n}{4}) + O(1)$
$\Rightarrow T(n)\in\Theta(n^{\log_4 3})$

$A.P. \Rightarrow \sum_{i=1}^{k} i = 1+2+3\cdots+k = \frac{k(k+1)}{2}$ / $H.P \Rightarrow \sum_{i=1}^{k} = \frac{1}{1}+\frac{1}{2}+\frac{1}{3}\cdots+\frac{1}{k} = \log k$ $\Theta(\log k)$

$G.P. \Rightarrow \sum_{i=0}^{k} x^i = x^0+x^1+x^2+\cdots+x^k = \frac{x^{k+1}-1}{x-1}$ / ∞ G.P $\Rightarrow$ when $|x|<1$ e.g $x=\frac{1}{2}$; $\sum_{i=0}^{k} = \frac{1}{1-x}$

$\lim_{n\to\infty}\frac{f(n)}{g(n)}$ a) $0\Rightarrow f(n)\in o(g(n))$ b) $c\neq0\Rightarrow f(n)\in\Theta(g(n))$
c) $\infty\Rightarrow g(n)\in o(f(n))$