$n^2, n^3, n^2\log n, \sqrt{n}, \underline{\log n}, \underline{10^n}, 2^n, \underline{8^{\log_2 n}}$

Of course;

$\sqrt{n} < n^2 < n^3$

$\lim_{n\to\infty} \frac{\log n}{\sqrt{n}} \Rightarrow$ Derivative $\frac{\frac{1}{n}}{\frac{1}{2}\cdot\frac{1}{\sqrt{n}}} \Rightarrow n=\infty \quad \log n < \sqrt{n}$

$\lim_{n\to\infty} \frac{n^2\log n}{n^2} \quad \log n = \infty \Rightarrow n^2 < n^2\log n$

$\hookrightarrow \frac{n^2\log n}{n^3} \Rightarrow n^2\log n < n^3$

$\lim_{n\to\infty} \frac{2^n}{n^2} \Rightarrow f(n) = \infty \quad n^3 < 2^n \leq 10^n$

$\log n < \sqrt{n} < n^2 < n^2\log n < n^3 < 8^{\log_2 n} < 2^n \leq 10^n$

```
a-) int p-1(int my_array[]){
        for(int i=2; i<=n; i++)                    \
            if(i%2==0){             \                \
                count++;}            > Θ(1)           > Θ(n)
            else{                   /                /
                i=(i-1)i;}         /                /
    }
```

→ Time complexity: Θ(n)

→ Space complexity: Θ(n)

```
b-) int p-3(int my_array[]){
        first_element=my_array[0];   -> O(1)
        second_element=my_array[0];  -> O(1)
        for(int i=0; i<sizeofArray; i++){              ┐
          if(my-array[i]<first_element){       ┐       │
            second_element =first element;     │       │ O(n
            first_element =my-array[i];}       │ O(1)  │
          else if(my-array[i]< second_element){│       │
              if(my_array[i]! = second_element){       │
                 second_element =my_array[i];} ┘       ┘
        } }
```

→ Time complexity: O(n)

→ Space complexity: O(n)

```
c-) int p-3(int array[]){
        return array[0]*array[2]; → Θ(1)
```

→ Time complexity: Θ(1)

→ Space complexity: Θ(1)

```
d-) int p-4(int[] array, int n){
        Int sum =0                          -> Θ(1)
        for(int i=0; i<n; i=i+5)                         \
            sum+= array[i] * array[i]; -> Θ(1)           / Θ(n)
        return sum; -> Θ(1)
```

→ Time complexity: $\Theta(1)$

→ Space complexity: $\Theta(1)$

```
e-) int p-5(int[] array, int n){
        for( int i=0; i<n; i++)                                   ] Θ(nlog2n)
            for(int j=1; j<i; j=j*2)                              ] Θ(log2n)
                printf("%d", array[i] * array[j]); -> Θ(1)        ]
    }
```

→ Time complexity: $\Theta(n \log_2 n)$

→ Space complexity: $\Theta(1)$

```
f-) int p-6(int array[], int n){
        if(p-4(array,n) >1000) -> Θ(n)
            p-5(array,n) -> Θ(nlog2n)
        else
            printf("%d", p-3(array) * p-4(array,n) -> Θ(1.n)=Θ(n)
    }
```

→ $T_{worst}(n) = \Theta(n) + \Theta(n \log_2 n) = \Theta(n \log_2 n)$

→ $T_{best}(n) = \Theta(n) + \Theta(n) = \Theta(n)$

→ Space complexity: $\Theta(1)$

I→ $2^{n+1} = \Theta(2^n) \iff c_1.2^n \geq 2^{n+1} \geq c_2.2^n$, $c_1;c_2>0$ for all $n \geq n_0$

$c_1.2^n \geq 2^{n+1}$ $\quad$ $2^{n+1} \geq c_2 2^n$

for n=1 $c_1.2 \geq 4$ $\quad$ for n=1 $4 \geq c_2.2$ $\quad$ $2 \geq c_2$

$c_1 \geq 2$

$n \geq n_0 = 1$, $\underline{c_1 = c_2}$ ✓

II⇒ $2^{2n} = \Theta(2^n) \iff c_1.2^n \geq 2^{2n} \geq c_2.2^n$, $c_1,c_2>0$ for all $n \geq n_0$

$c_1.2^n \geq 2^{2n}$

$c_1 \geq 2^n$, false, c_1 is constant.

```
public void pairs(int[] array, int value){
  for(int i=0; i<array.length-1; i++){
    for(int j=0; j< array.length; j++){
      if(array[i] + array[j] ==value){
        System.out.println(array[i] +" and"+ array[j]
          +" is pairs");
      }
    }
  }
}
```

```
public static void rec_pairs(int[] array, int index, int iter, int valu
  if(array[index] + array[iter] == value){
    System.out.println(array[index] + array[iter] " is pairs"); }
  if( index == array.length-1){
    return; }
  if(iter == array.length-1){
    rec_pairs(array, index+1, 0, value);
  else
    rec_pairs(array, index, iter+1, value);
}
```