Consider the problem of searching an elemet x in an array 'arr[]' of size n. The problem can be solved in O(Logn) time if.

- Array is sorted
- 2) 2) Array is sorted and rotated by k. k is given to you and k <= n
- 3) 3) Array is sorted and rotated by k. k is NOT given to you and k <= n</p>
- 4) 4) Array is not sorted

1 Only

1 & 2 only

1, 2 and 3 only

1, 2, 3 and 4

## What does the following function do?

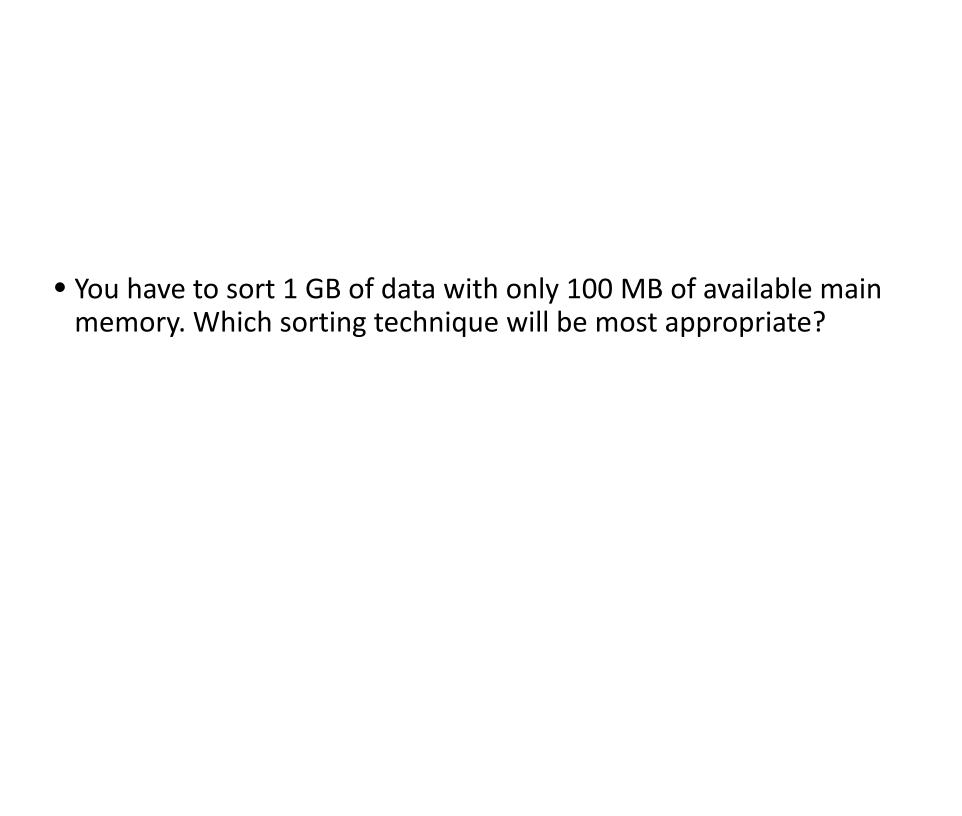
```
int fun(int x, int y)
{
    if (y == 0)     return 0;
    return (x + fun(x, y-1));
}
```

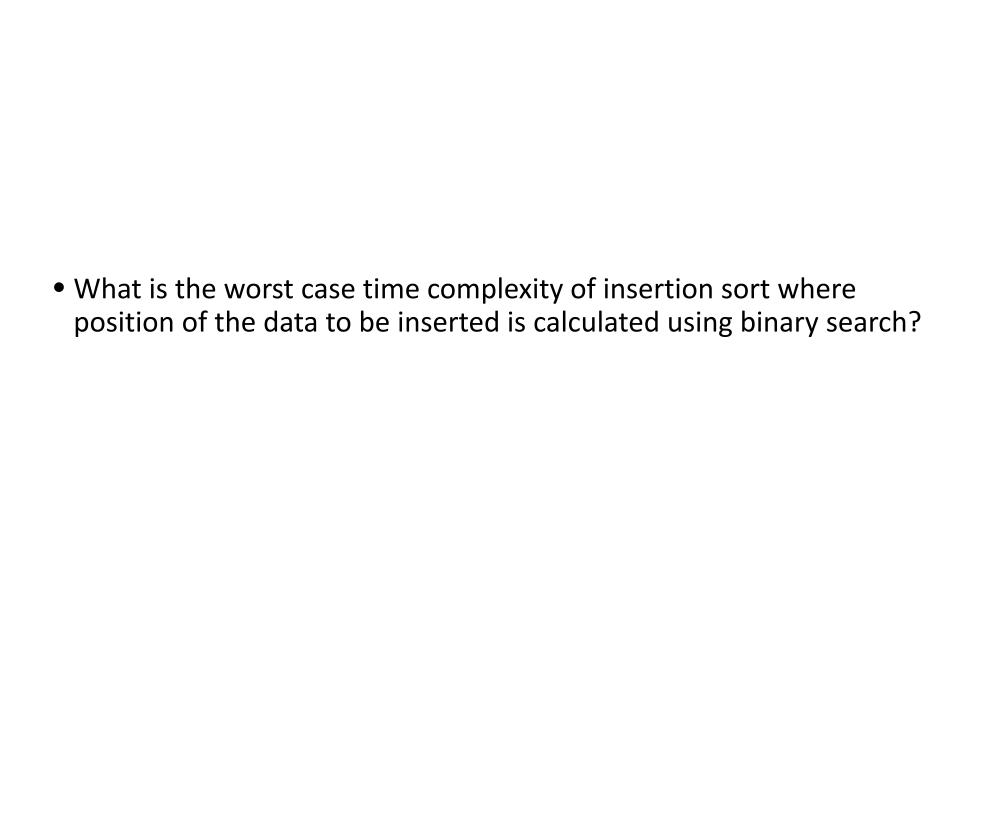
```
A x + y
B x + x*y
C x*y
D x<sup>y</sup>
```

## What does the following function do?

```
int fun(int x, int y)
{
    if (y == 0)         return 0;
    return (x + fun(x, y-1));
}

int fun2(int a, int b)
{
    if (b == 0) return 1;
    return fun(a, fun2(a, b-1));
}
```





Consider a sorted array of n numbers. What would be the time complexity of the best known algorithm to find a pair 'a' and 'b' such that |a-b| = k, k being a positive integer.

O(n)
O(n log n)
O(n ^ 2)
O(log n)