Sorting Algorithms via Haskell

Rishav Gupta

November 2021

1 Insertion Sort

Code ->

```
insert :: Ord a => a->[a] -> [a]
insert  v [] = [v]
insert v (x:xs) = if v<=x then (v:x:xs) else x:(insert v xs)
isort :: Ord a => [a] -> [a]
isort [x] = [x]
isort (x:xs) = insert x (isort xs)
```

Worst Case Time Complexity $-> O(n^2)$

2 Merge Sort

Code ->

```
merge :: Ord a => [a] -> [a] -> [a]
merge [] l = l
merge l [] = l
merge (x:xs) (y:ys) = if x<=y then x:(merge xs (y:ys)) else y:(merge (x:xs) ys)
msort :: Ord a => [a] -> [a]
msort [] =[]
msort [x] =[x]
msort l = merge (msort fr) (msort bc)
 where
     n= length l
     (fr,bc) = splitAt (n `div` 2) l
```

Worst Case Time Complexity$-> O(n \cdot \log n)$

3 Quick Sort

Code ->

```
qsort :: Ord a => [a] -> [a]
qsort []=[]
qsort [x] = [x]
qsort (x:xs) = (qsort (bc)) ++ [x] ++ (qsort (fr))
 where
     fr  = [ y| y <- xs , y>x]
     bc = [ y | y<- xs , y<=x]
```

Worst Case Time Complexity $-> O(n^2)$

4 Bubble Sort

Code ->

```
run :: Ord a => [a] -> [a]
run [] = []
run [x] = [x]
run (x:y:zs) = if x<=y then x:(run (y:zs)) else y:(run (x:zs))
bsort :: Ord a => [a] -> [a]
bsort [] = []
bsort [x] = [x]
bsort l = iter run (length l) l
 where
     iter :: (a->a) -> Int -> a ->a
     iter f 0 x =x
     iter f i x = f ( iter f (i-1) x)
```

Worst Case Time Complexity $-> O(n^2)$