#LECTURE 17: Mutable Algorithms II

"Intelligence is the ability to adapt to change"

— Stephen Hawking, 1942–2018

```
> class Hashable a where
>   hash :: a -> Int
```

```
hash "Hello"  = 73
hash "World"  = 42
hash "algorithms" = -4721
hash "purple" = 73
```

↖ hash collision

nub removes duplicates (not necessarily stable).

```
nub [3, 2, 5, 3, 3, 5, 8]
=
[3, 2, 5, 8]
```

hash 3 = 42 hash 5 = 255

hash 2 = 0 hash 8 = 42

[Int]

```
[[ #0 : [] 2:[]
   #1 : []
   #2 : []
   ⋮
   #42 : [] 3:[] 8:3:[]
   ⋮
   #255 : [] 5:[]
]]
```

```
nub :: [Int] -> [Int]
nub xs = concat (
  runST $ do
    axss <- newListArray (0,255) (repeat [])
    sequence[ let hx = (hash x) `mod` 256
              ys <- readArray axss hx
              unless (x `elem` ys) $do
                writeArray axss hx (x:ys)

            | x <- xs ]

    xss <- getElems axss
    return xss
           )
           [[Int]]
```

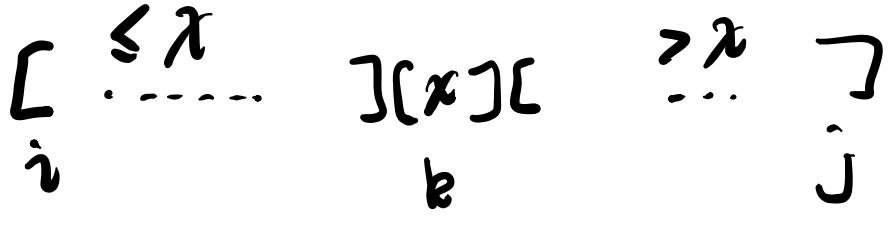
We want to define an in-place quicksort:

```
> swap :: STArray s Int a -> Int -> Int -> ST s ()
> swap axs i j = do
>     x <- readArray axs i
>     y <- readArray axs j
>     writeArray axs i y
>     writeArray axs j x
```

readArray axs j >>=
writeArray axs i

```
> qsort :: Ord a => [a] -> [a]
> qsort xs = runST $ do
>         axs <- newListArray (0,n) xs
>         aqsort axs 0 n
>         xs' <- getElems axs
>         return xs'
>   where
>       n = length xs

> aqsort :: Ord a => STArray s Int a -> Int -> Int -> ST s ()
> aqsort axs i j
>       | i >= j = return ()
>       | otherwise = do
>           k <- apartition axs i j
>           aqsort axs i (k-1)
>           aqsort axs k j
```



```
> apartition :: Ord a => STArray s Int a
>            -> Int -> Int -> ST s Int
> apartition axs p q = do
>   x <- readArray axs p
>   let loop i j
>       | i > j = do swap axs p j
>                    return j
>       | otherwise = do
>           u <- readArray axs i
>           if u < x
>             then do loop (i+1) j
>             else do swap axs i j
>                     loop i (j-1)
>   loop (p+1) q
```