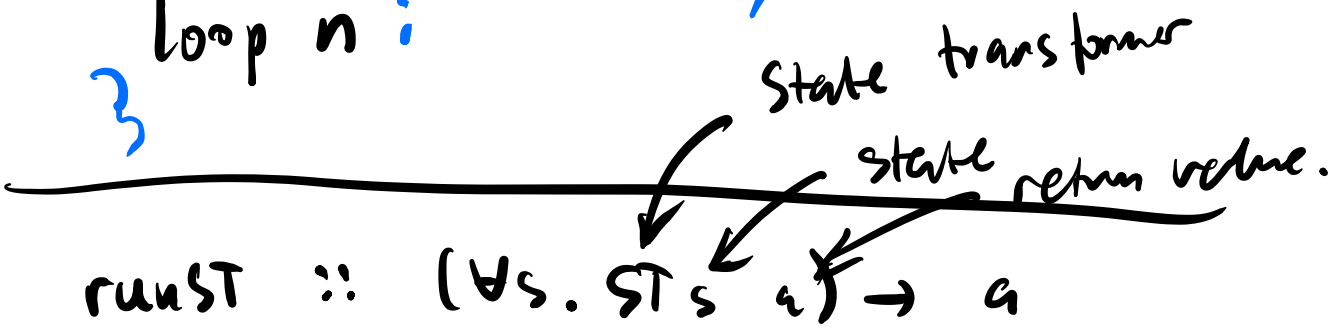
#LECTURE 16 : Mutable Algorithms I

```
>fib :: Int -> Integer
>fib  n  =  loop  n  0  1
>     where
>         loop  0  x y  =  x
>         loop  n  x y  =  loop (n-1) y (x+y)
```

```
> fib' :: Int -> Integer
> fib' n = runST $ do {
>      rx <- newSTRef 0 ;
>      ry <- newSTRef 1 ;
>      let loop 0 = do {
>            x <- readSTRef rx ;
>            return x ;  }
>          loop n = do
>          { x <- readSTRef rx ;
>            y <- readSTRef ry ;
>            writeSTRef rx y ;
```

```
>        writeSTRef ry (x+y);
>        loop (n-1) ;};
>    loop n;
};
```




```
runST :: (∀s. ST s a) → a

newSTRef :: a → ST s (STRef s a)
readSTRef :: STRef s a → ST s a
writeSTRef :: STRef s a → a → ST s ()

return :: a → ST s a
```

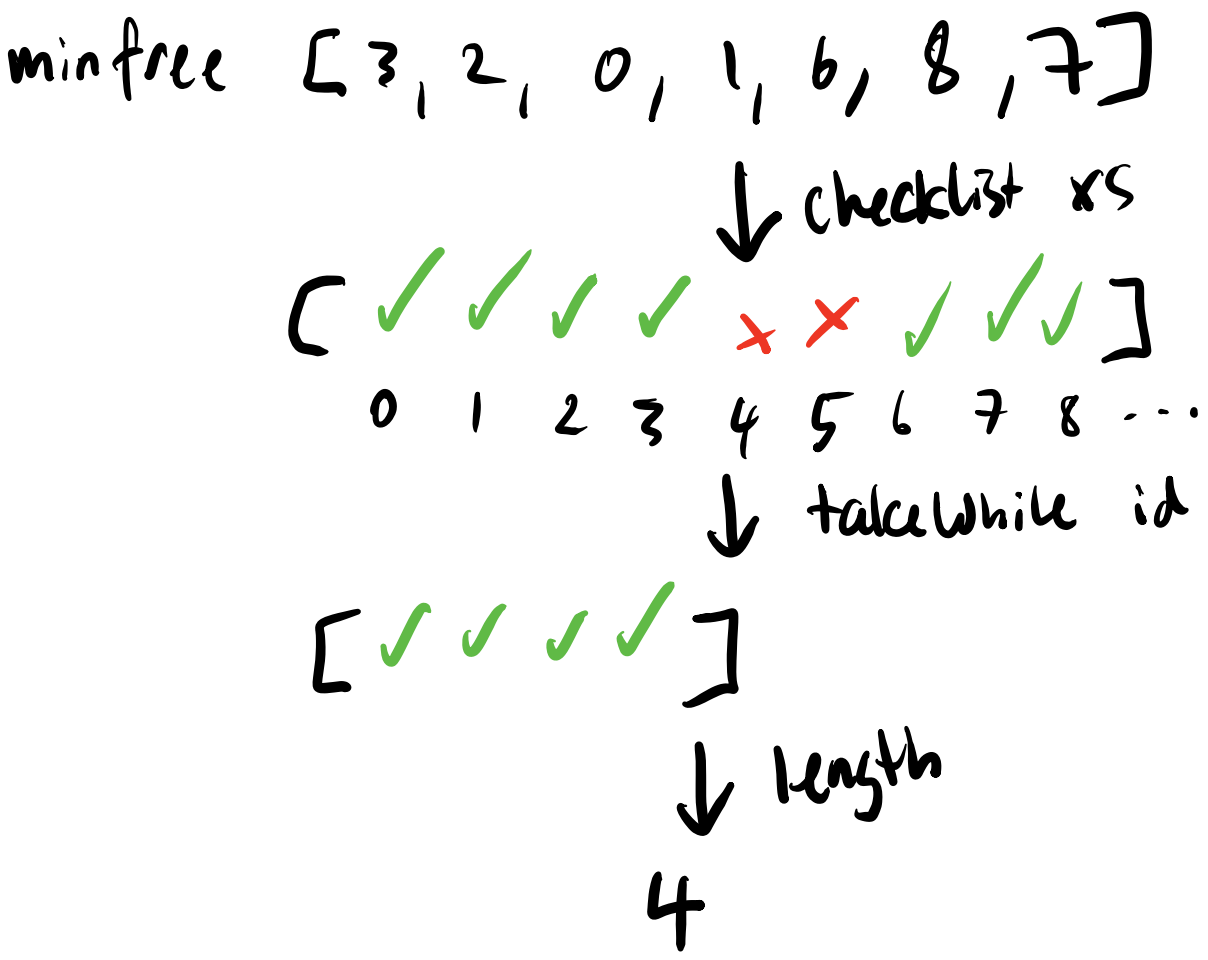
```
newArray :: Ix i ⇒ (i, i) → a → ST s (STArray s i a)
```
(indices range; default)

```
readArray :: Ix i ⇒ STArray s i a → i → ST s a

writeArray :: Ix i ⇒ STArray s i a → i → a → ST s ()
```

---

minfree xs returns the smallest free natural number in xs.




```
minfree :: [Int] -> Int
minfree xs = length (takeWhile id (checklist xs))

checklist :: [Int] -> [Bool]
checklist xs = runST $ do
    ays <- newArray (0, m-1) False :: ST s (STArray s Int Bool)

    sequence [writeArray ays x True | x <- xs, x<m]
    xs' <- getElems ays
    return xs'
```

where
  n = length xs

extracts all elements.

```
getElems :: Ix i => STArray s i a -> ST s [a]
```

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

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

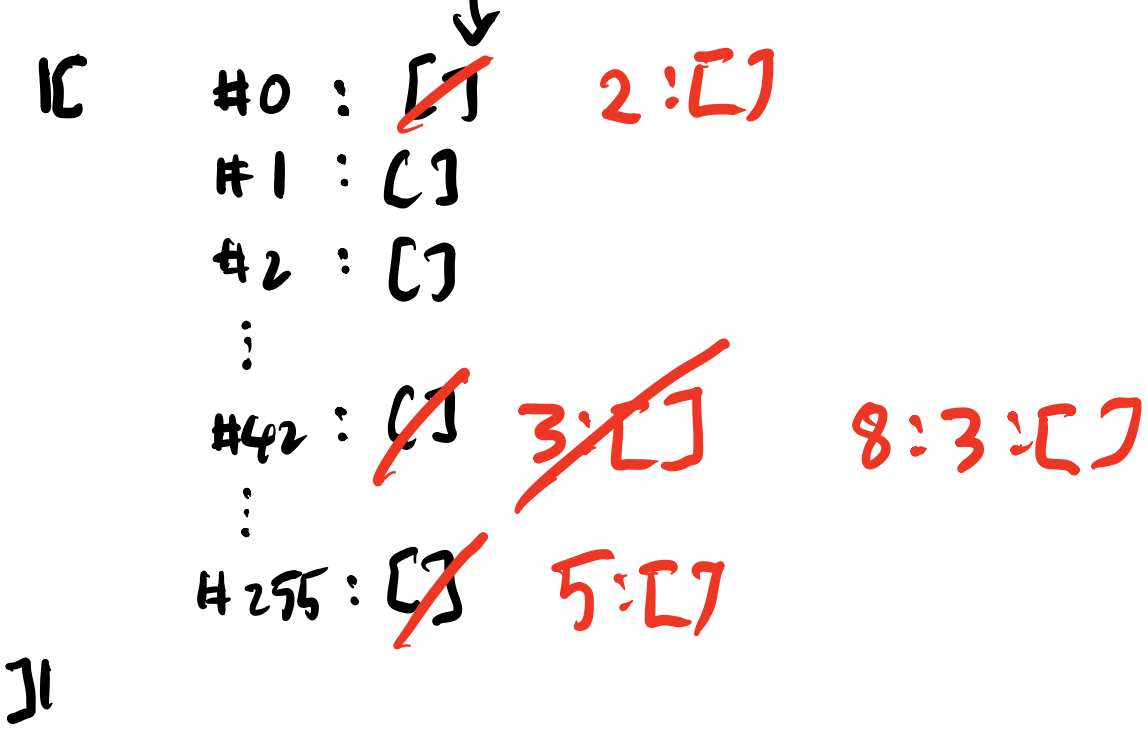
↖ hash collision

nub removes duplicates (not necessarily stable).

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

hash 3 = 42
hash 2 = 0
hash 5 = 255
hash 8 = 42

```
[Int]
```


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