卐 Shrinath
Date 22-7-23
Page No.

DS

Sorting

intro & e.g. in java notes

Bubble sort ⇒ also known as
sinking sort
exchange sort

Complexity of Bubble sort ⇒

Space complexity = O(1) // constant // no extra space required i.e. copying the array etc.

also known as inplace sorting algorithm

Time complexity Best case O(N) ⇒ sorted

worst case (N^2) ⇒ sorted in opposite

As the size of array is growing, the no. of comparitions is also growing

Shrinath
Date
Page No.

Best case - 1, 2, 3, 4, 5

i = 0 th pass , j = n

Note - when j never swaps for a value of i

hence you can end the program

Best case comparisions = N

Worst case -

5, 4, 3, 2, 1

1st pass 4, 5, 3, 2, 1

4, 3, 5, 2, 1

i = 0 4, 3, 2, 5, 1

4, 3, 2, 1, 5

2nd pass

4, 3, 2, 1 3rd pass i = 2

i=1 3, 4, 2, 1 3, 2, 1

3, 2, 4, 1 2, 3, 1

3, 2, 1, 4 2, 1, 3

4th pass i = 3

2, 1, 3

1, 2, 3

total comparision $(N-1)+(N-2)+(N-3)+(N-4)$

$4N - (1+2+3+4)$

$4N - \left(\frac{N*(N+1)}{2}\right)$

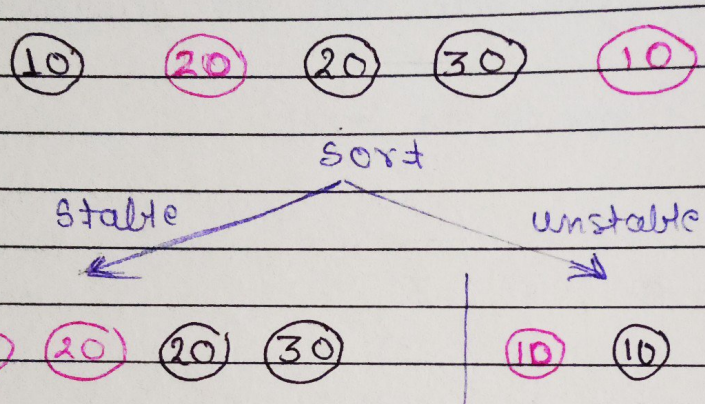
$4N - \left(\frac{N^2+N}{2}\right) = \frac{8N-N^2-N}{2}$

$O\left(\frac{7N-N^2}{2}\right) = O(N^2)$

Shrinath
Date
Page No.

Bubble sort is a stable sorting algorithm

given array ⇒ value is same & color is different



stable sorting algorithm

in original array black ball of 10 was before red ball of 10

in the sorted array this order is maintained

~~Ex:~~ when the original order is maintained for values that are equal. stable sorting

unstable sorting algorithm

even though it is sorted by values, it is not sorted according to the order that was given initially.

when the original order is not maintained when two or more values are equal. unstable sorting

卐 Shrinath
Date
Page No.

```
static void bubble (int[] arr)
{
    // run the steps n-1 time
    boolean swapped;
    for (int i=0; i< arr.length; i++)
    {
        swapped = false;
        // for each step max item will come at the last respective index
        for (int j=1; j < arr.length-i; j++)
        {
            // swap if the item is smaller than the previous
            if (arr[j] < arr[j-1])
            {
                int temp = arr[j];
                arr[j] = arr[j-1];
                arr[j-1] = temp;
                swapped = true;
            }
        }
        // if you did not swap for a particular value of i it means the array is sorted, hence stop the program
        // break when i when swapped == false
        if (!swapped)    // !false == true
            break;
    }
}
```

Shrinath
Date
Page No.

Selection sort

(better then Java notes it is best ✓ read it)

~~explanation in java notes~~

as the name suggest, we are going to select an element and we are going to sort that element or put it on its right position.

max

4, 5, 1, 2, 3

for finding max comparisons $n-1$

4, 3, 1, 2, 5 $n-2$

2, 3, 1, 4, 5 $n-3$

2, 1, 3, 4, 5 $n-4$

1, 2, 3, 4, 5

$n-4$: 3 : 2 : 1 : 0

min

4, 5, 1, 2, 3

1, 5, 4, 2, 3

1, 2, 4, 5, 3

1, 2, 3, 4, 5

1, 2, 3, 4, 5

Total comparisons

$0+1+2+3+\cdots+n-2+n-1$

$= \frac{n(n+1)}{2}$

$= \frac{(n-1)(n-1+1)}{2}$

$= \frac{(n-1)n}{2}$

$= \frac{n^2-n}{2}$

$O(n^2)$

worst case $O(n^2)$

best case also $O(n^2)$

Selection sort is an unstable sorting algo

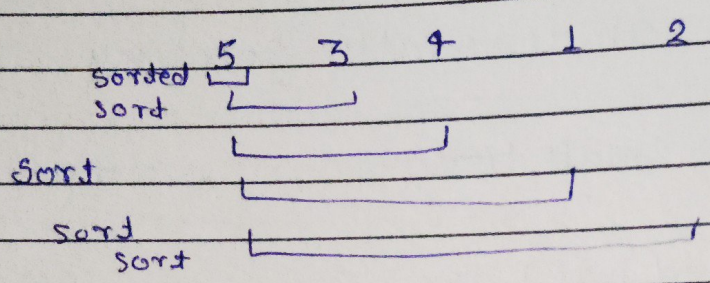
it performs well on small lists.

```
static void selection ( int[] arr )
{
    for ( int i=0 ; i < arr.length ; i++ )
    { // find the max item in the remaining array and swap with correct index
        int last = arr.length - i - 1 ;
        int maxIndex = getMaxIndex ( arr , 0 , last );
        swap ( arr , maxIndex , last )
    }
}

static void swap ( int[] arr , int maxIndex , int last )
{
    int temp = arr [maxIndex] ;
    arr[maxIndex] = arr [last];
    arr [last] = temp ;
}

static int getMaxIndex ( int[] arr , int start , int end )
{   int max = start ;
    for ( int i = start ; i <= end ; i++ )
    {   if ( arr[max] < arr[i] )
            max = i ;   }
    return max ;
}
```

Shrinath
Date
Page No.

Insertion Sort :-

explanation in java notes



Complexity -

worst case $O(N^2)$

array 5, 4, 3, 2, 1

Comparitions 0 1 2 3 4

$\Rightarrow 0+1+2+3+4 \cdots +(n-1)$

$\Rightarrow \frac{n(n+1)}{2} \Rightarrow \frac{(n-1)(n+1-1)}{2} = \frac{n(n-1)}{2}$

$\Rightarrow O(n^2)$

Best case Array is already sorted

1, 2, 3, 4, 5

Comparitions 0 1 1 1 1

total $(n-1)$

$O(n)$

Shrinath
Date
Page No.

Why use insertion sort ?

- it is adaptive : steps get reduced if array is sorted. no. of swaps reduced as compared to bubble sort
- stable
- used for smaller values of n ⇒ works good when array is partially sorted → It takes part in hybrid sorting algorithms.

another way by shifting in java notes.

```
static void insertion ( int[] arr )
{
    for ( i=0 ; i < arr.length-1 ; i++ )
    {
        for ( j= i+1 ; j >0 ; j-- )
        {
            if ( arr[j] < arr[j-1] )
                swap ( arr , j , j-1 )
            else
                break ;
        }
    }
}
static void swap ( int[] arr , int start , int end )
{
    int temp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp ;
}
```

卐 Shrinath
Date
Page No.

Cyclic Sort -

3 , 5 , 2 , 1 , 4

★★★ when given numbers from range 1 , N ⇒ use cyclic sort

after sortin 1 , 2 , 3 , 4 , 5

index → 0 1 2 3 4

index = value - 1

why ? because index value start from 0 and we are given elements from 1 to n.

0 1 2 3 4

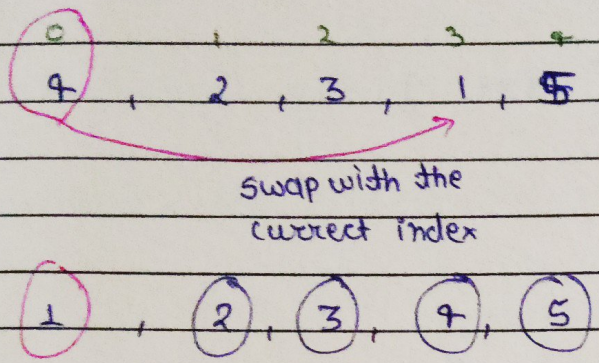
3 , 5 , 2 , 1 , 4

swap with correct index

2 , 5 , 3 , 1 , 4

swap with correct index

5 , 2 , 3 , 1 , 4



every unique item is only getting swapped once

number of comparision.

$(N-1)$ swaps + N swaps

4 swaps + N swaps

$(2N-1)$ swaps $O(N)$

\# i pointer, we will only move the i pointer forward when i^{th} is at current index.

code → आगे

when we have continuous element, or given arrays from $1-n$, $0-n$.

Shrinath
Date
Page No.

```
static void sort ( arr )
{
    int i = 0
    while ( i < arr.length )
    {
        int currectindex = arr[i] - 1 ;
        if ( arr[i] != arr[ currectindex] )
        {
            swap ( i , (currectindex) ) ;
        }
        else
            { i++ ; }
    }

static void swap( int first , int second )
{
    int temp = arr [first] ;
    arr [first] = arr [second] ;
    arr [second] = temp ;
}
```

Shrinath
Date
Page No.

Q. Missing number

watch in leet soln ⇒ best is use formula $\frac{n(n+1)}{2}$

2nd approch ⇒ use cyclic sort.
revise again

Q. Find all numbers disappeared in an Array

Tips

if range ⇒ [0, N]
* every element will be at index = value

if range ⇒ [1, N]
* every element will be at index = value - 1

N = 8

[4, 3, 2, 7, 8, 2, 3, 1]
(0 1 2 3 4 5 6 7)

swap with the currect index

[7, 3, 2, 4, 8, 2, 3, 1]
(0 1 2 3 4 5 6 7)

swap with the currect ans

[3, 3, 2, 4, 8, 2, 7, 1]
(0 1 2 3 4 5 6 7)

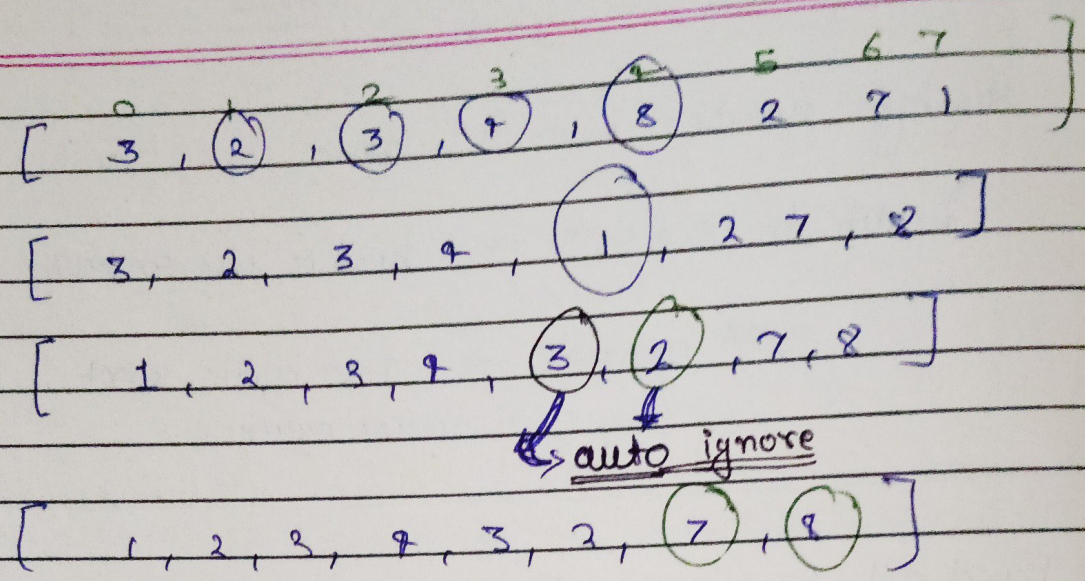
[2, 3, 3, 4, 8, 2, 7, 1]

[3, 2, 3, 4, 8, 2, 7, 1]

arr[i] == arr[currect]

// else it++ // automatically handle by cyclic sort condition

Shrinath
Date
Page No.



means after cyclic sort ⇒ अगर element present है तो, वो अपने correct index पर पहुंच जाते है।

अगर नहीं present है तो उसकी जगह कोई से भी duplicate element आ जाते हैं।

So, 5, 6 are missing

```
{ int i=0;
  while ( i < nums.length )
  { int correct = nums[i]-1;
    if( num[i] != nums[correct])
        swap ( nums, i, correct)
    else
  }       i++;
    for ( int index=0; index< nums.length ; index++)
    { If (nums[index] != index+1)
      {   lst.add ( index+1 );
    return list ; }
```

Shrinath
Date
Page No.

Q: Find the Duplicate Number

Soln revision in leet 9.

Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n] inclusive.

There is only one repeated number in nums, return this repeated number

(3+1)

मतलब अगर 4 size का array है तो [1, 3] range

[- - - -] 1, 2, 3

एक place पर repeated element होगा।

0 1 2 3 4

[1, 3, 4, 2, 2]

[1, 3, 4, 2, 2]

[1, 4, 3, 2, 2]

[1, 2, 3, 4, 2]

```
if ( nums[i] != i+1 )
{ int correct = arr[i] - 1;
  if ( arr[i] != arr [correct]
      swap ( ——— );
  } else    return arr[i];
else  i++;
```

~~if (nums[i] != nums (correct))~~

~~{ if (nums [correct] != correct + 1;)~~

~~swap~~

~~else return nums[i];~~

~~}~~

else i++;

Shrinath
Date
Page No.

Q Find all duplicates in an array.

solved in leet watch again for revision

Q. Set mismatch

by mistake one is dublicate which one number is dublicate and what there should be present.

[2 , 1 , 4 , 2 , 6 , 5] Ans [2 , 3]

2 duply. 3 missing

after cyclic sort

```
while (i < n)
{ int currect = nums[i] - 1;
  if ( nums[i] != nums[currect] )
      { swap( ——— ); }
  else  i++;
}
```

[1 , 2 , 2 , 4 , 5 , 6]
0 1 2 3 4 5

missing → wrong index + 1

duplg → nums [wrong index]

-1 0

(0 > 0 && arr[nums[n] && arr[n] != arr[n]

-1 0

卐 Shrinath
Date
Page No.

```
class SetMismatch
{
    public int[] function ( int[] arr)
    {   int i = 0;
        while ( i < arr.length )
        {
            int correct = arr[i] - 1 ;
            if ( nums[i] != nums[correct] )
            {  swap (arr, i, correct)
            }
            else  i++ ;
        }
    for ( int index = 0 ; index < arr.length ; index++)
    {
        if ( arr[index] != index+1 )
        return  new int[] { arr[index], index+1}
    }
    return  new int[] {-1, -1} ;
}
```

\+ leet hard

See again all the questions
these are important.