Lecture 7 - January 28

Asymptotic Analysis of Algorithms, Arrays and Linked Lists

Deriving Upper Bounds from Code
Inserting into an Array
Sorting Orders

Announcements/Reminders

- **Assignment 1** solution to be released **soon**
- ***splitArrayHarder***: an **extended** version released
- Office Hours: 3pm to 4pm, Mon/Tue/Wed/Thu
- Contact Information of TAs on common eClass site

$[a, b]$

↳ # of integers in this closed interval is

$$b - a + 1$$

$[0, n-1]$

↳ $(n-1) - 0 + 1 = n$

$[2, 99]$

↳ $99 - 2 + 1$

Determining the Asymptotic Upper Bound (3.1)

```
1  boolean containsDuplicate (int[] a, int n) {
2    for (int i = 0; i < n; ) {
3      for (int j = 0; j < n; ) {
4        if (i != j && a[i] == a[j]) {
5          return true; }
6        j ++; }
7      i ++; }
8    return false; }
```

P1 (lines 4–6) O(1); P2 (line 7) O(1); line 8 O(1)

* P1 gets executed for each combination of (i, j)
* P2 gets executed for each value of i.

Pattern of Loop Counters

outer loop runs $[0, n-1]$ — n iterations

i	j					
0	0	1	2	⋯	n-1	n iterations
1	0	1	2	⋯	n-1	
2	0	1	2	⋯	n-1	
⋮			⋮			
n-1	0	1	2	⋯	n-1	

$(i, j) = (2, 2)$ → when L4~L6 exec. once.

$$O(\underbrace{n}_{\#\text{ values of } i} \cdot \underbrace{n}_{\#\text{ values of } j \text{ for each } i} \cdot \underbrace{1}_{L4\text{-}L6} + \underbrace{n}_{\#\text{ values of } i} \cdot \underbrace{1}_{L7} + \underbrace{1}_{L8})$$

$$= O(n^2 + n + 1)$$

$$= O(n^2)$$

Determining the Asymptotic Upper Bound (3.2)

```
1  boolean containsDuplicate (int[] a, int n) {
2    for (int i = 0; i < n; ) {
3      for (int j = 0; j < n; ) {
4        if (i != j && a[i] == a[j]) {
5          return true; }
6        j ++; }
7      i ++; }
8    return false; }
```

$O(10{,}000 \cdot n)$

$= O(n)$

```
1  boolean containsDuplicate (int[] a, int n) {
2    for (int i = 0; i < n; ) {
3      for (int j = 0; j < n; ) {
4        if (i != j && a[i] == a[j]) {
5          return true; }
6        j ++; }
7      i ++; }
8    return false; }
```

$O(1M \cdot 1M)$

$= O(c \cdot n^0)$

$= O(1)$

Determining the Asymptotic Upper Bound (4)

```
1  int sumMaxAndCrossProducts (int[] a, int n) {
2    int max = a[0];
3    for(int i = 1; i < n; i ++) {
4      if (a[i] > max) { max = a[i]; }
5    }
6    int sum = max;
7    for (int j = 0; j < n; j ++) {
8      for (int k = 0; k < n; k ++) {
9        sum += a[j] * a[k]; } }
10   return sum; }
```

1

n

1

n^2

1

$O(1+n+1+n^2+1)$

$= O(n^2+2n+3)$

$= O(n^2)$

Determining the Asymptotic Upper Bound (5)

```
1  int triangularSum (int[] a, int n) {
2    int sum = 0;
3    for (int i = 0; i < n; i ++) {
4      for (int j = i; j < n; j ++) {
5        sum += a[j]; } }
6    return sum; }
```

1 (line 2) · 1 (line 5) → exec for each (i, j) · 1 (line 6)

Pattern of (i, j)

# of times L5 is exec: $n + (n-1) + (n-2) + \cdots + 1$

$= \frac{(n+1) \cdot n}{2}$ is $O(n^2)$

# [0, n-1]	i	j					
n	0	0	1	2	⋯	n-1	[0, n-1] → n
	1		1	2	⋯	n-1	[1, n-1] → n-1
	2			2	⋯	n-1	[2, n-1] → n-2
	⋮			⋮		⋮	
	i						
	n-1					n-1	[n-1, n-1] → 1

$O(\underbrace{1}_{L2} + \underbrace{n^2}_{\#(i,j)} \cdot \underbrace{1}_{L5} + \underbrace{1}_{L6})$

$= O(n^2 + 2) = O(n^2)$

Asymptotic Upper Bound: Arithmetic Sequence/Progression

$$\underbrace{i + 0 \cdot c}_{\text{start term}} + \overbrace{(i + c)}^{\text{common difference}} + (i + 2 \cdot c) + \cdots + \underbrace{(i + (n-1) \cdot c)}_{n \text{ terms}}$$

$$\text{Sum} \quad \frac{(i + (i + (n-1) \cdot c)) \cdot n}{2} = \frac{c \cdot n^2 + (2i - c) \cdot n}{2}$$

is order of $O(n^2)$

Inserting into an Array

object creation: O(1)

[i+1, n] → n-(i+1)+1

insert "e" to index "i" (0 ~ n)

a.length

```
String[] insertAt(String[] a, int n, String e, int i)
1  String[] result = new String[n + 1];                    1
n* for(int j = 0; j <= i - 1; j ++){ result[j] = a[j]; }   j: [0, i-1] → i iterations → n times
1  result[i] = e;
n  for(int j = i + 1; j <= n; j ++){ result[j] = a[j-1]; }  j: [i+1, n] → n-i → n times
1  return result;
```

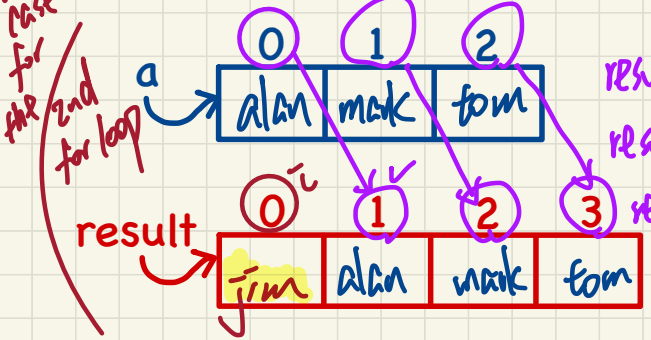
* O(n · 1) → max value of i

O(1+n+1+n+1) = O(n)

Example:

insertAt({alan, mark, tom}, 3, jim, 0)

worst case for the 2nd for loop



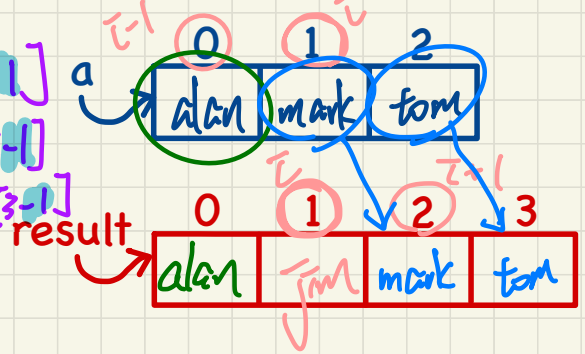
result[1] == a[1-1]

result[2] == a[2-1]

result[3] == a[3-1]

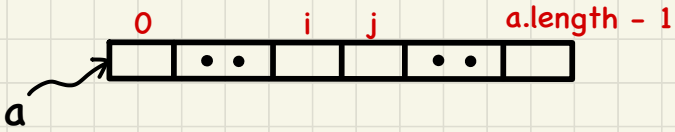
Example:

insertAt({alan, mark, tom}, 3, jim, 1)



Exercise: insertAt({alan, mark, tom}, 3, jim, 3)

Sorting Orders of Arrays



non-descending

$\triangleq \neg(\text{descending})$

$\equiv \neg(a[i] > a[i+1])$

$\equiv a[i] \leq a[i+1]$