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

Determining the Asymptotic Upper Bound (1)

approx.

→ vs. Counting # Bs

```
1  int maxOf (int x, int y) {
2    int max = x;
3    if (y > x) {
4      max = y;
5    }
6    return max;
7  }
```

$O(\underbrace{1}_{L2} + \underbrace{1}_{L3} + \underbrace{1}_{L4} + \underbrace{1}_{L6}) = O(4 \cdot n^0)$

$= O(n^0)$

$O(1)$

Determining the Asymptotic Upper Bound (2)

```
1  int findMax (int[] a, int n) {
2    currentMax = a[0];
3    for (int i = 1; i < n; ) {
4      if (a[i] > currentMax) {
5        currentMax = a[i]; }
6      i ++ }
7    return currentMax; }
```

1 n Iterations

1

1

$$O(1 + n \cdot 1 + 1) = O(n+2)$$

1 2 3 4~6 7 $= O(n)$

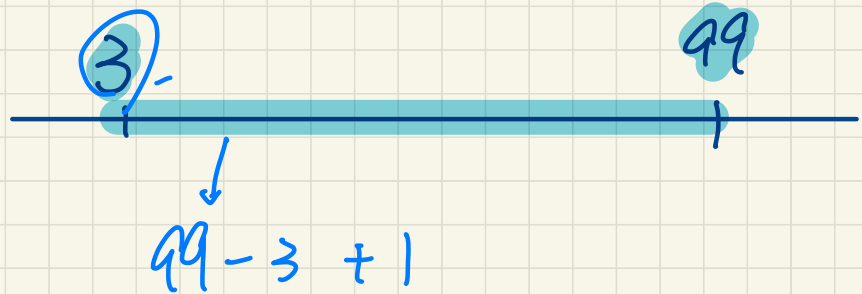
$[a, b]$

↳ $b - a + 1$. integers in the closed interval

$[0, n-1] \rightarrow (n-1) - 0 + 1 = n$.

$[3, 99]$

$99 - 3$



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) ↳ each exec

P2 (line 7) → O(1)

line 8 → O(1)

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

Patterns of Loop Counters

i	j					
0	0	1	2	...	n-1	n its.
1	0	1	2	...	n-1	n its
2	0	1	2	...	n-1	n its
⋮		⋮				
n-1	0	1	2	...	n-1	n its

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

$(2, 1) \rightarrow$ P1 exec once

i values $\cdot$ # j values $= n \cdot n$ cells in the matrix

$$O(\underbrace{n^2}_{\#\ (i,j)\ \text{combinations}} \cdot \underbrace{1}_{\text{L4-L6}} + \underbrace{n}_{\#\ i\ \text{values}} \cdot \underbrace{1}_{\text{L7}} + 1)$$

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

$$= O(n^2)$$

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; }
```

$O(1M \cdot 1M)$

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

$= O(1)$

a.length

b.length

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

$O(n \cdot m) = O(n^2)$

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

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

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

Determining the Asymptotic Upper Bound (5)

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

exec for each (i, j)

Pattern of (i, j)

i	j
0	0 1 2 ⋯ n−1
1	1 2 ⋯ n−1
2	2 ⋯ n−1
⋮	⋮
n−1	n−1

$[0, n-1] \to n$

$[1, n-1] \to n-1$

$[2, n-1] \to n-2$

$[n-1, n-1] \to 1$

of $[0, n-1]$: n

$$\#(i,j) = n + (n-1) + (n-2) + \cdots + 1 = \frac{(n+1) \cdot n}{2} \quad O(n^2)$$

$$O\left(\underbrace{1}_{\sim 2} + \underbrace{n^2}_{\#(i,j)} \cdot \underbrace{1}_{\sim 5} + \underbrace{1}_{\sim 6}\right)$$

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

Asymptotic Upper Bound: Arithmetic Sequence/Progression

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

$$\#\ \text{terms} : n$$

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

$$\hookrightarrow O(n^2)$$

Inserting into an Array

object creation: O(1)

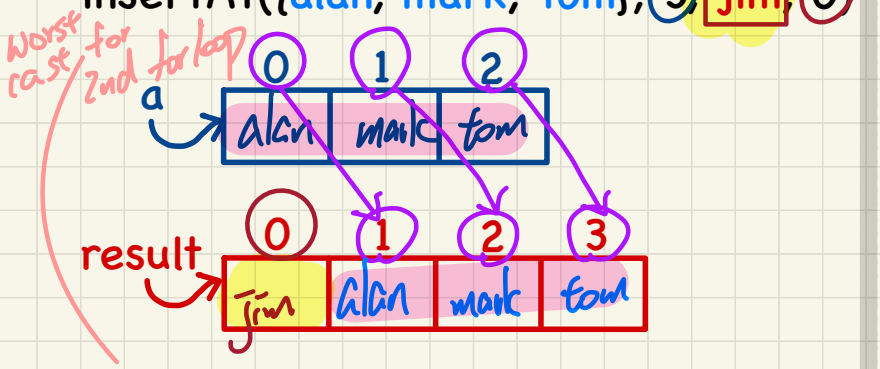
a.length

insert "e" to index "i" → 0 ≤ i ≤ n (insert first / insert last)

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

Example:

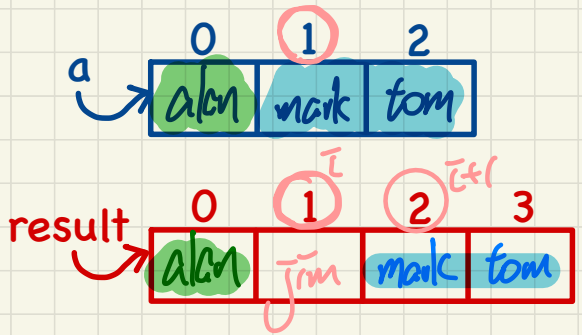
insertAt({alan, mark, tom}, 3, jim, 0)



Example:

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

O(1 + n·1 + 1 + (n-i)·1+1) = O(n)



Exercise: insertAt({alan, mark, tom}, 3, jim, 3) → worst case for 1st for loop