Lecture 7 - Monday, January 30

Announcements

- Written Test 1 guide released
  + EECS account login (for WSC computers)
  + PPY account + Duo Mobile (for eClass)
- Assignment 1 due in a week:
  + Tracing Recursion:
    - Paper: Call Stack vs. Tree
    - Debugger in Eclipse
  + Help: Scheduled Office Hours & TAs

Determining the Asymptotic Upper Bound (3)

$[a, b] = b - a + 1$

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

```java
boolean containsDuplicate (int[] a, int n) {
  for (int i = 0; i < n; ) {
    for (int j = 0; j < n; ) {
      if (i != j && a[i] == a[j]) {
        return true; }
      j ++; }
    i ++; }
  return false; }
```

body of inner loop: 1

p1

p2

* P1 gets executed once for every value of j.
* P2 gets executed once for every value of i.

Pattern of loop Counters

n^2 combinations of i, j

i	j					
0	0	1	2	...	(n-1)	n
1	0	1	2	...	(n-1)	n
2						
⋮	⋮					
n-1	0	1	2	...	(n-1)	n

outer loop runs for n times

inner loop runs for n times

$O(\underbrace{n^2}_{} \cdot \underbrace{1}_{L4 \sim L6} + n \cdot \underbrace{1}_{L7} + \underbrace{1}_{L8})$

iterations of outer loop

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

$= O(n^2)$

Determining the Asymptotic Upper Bound (4)

```
int sumMaxAndCrossProducts (int[] a, int n) {
  int max = a[0];
  for(int i = 1; i < n; i ++) {
    if (a[i] > max) { max = a[i];
  }
  int sum = max;
  for (int j = 0; j < n; j ++) {
    for (int k = 0; k < n; k ++) {
      sum += a[j] * a[k]; } }
  return sum; }
```

1

n

1

n^2

1

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

Determining the Asymptotic Upper Bound (5)

size of [2, n-1] is n-2.

```
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; }
```

pattern of combining (i, j)?

Pattern of (i, j)

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

$[1, n-1] = (n-1) - 1 + 1 = n-1$

$i = 0 \quad j = 0, 1, \dots, n-1 \quad n$

$i = 1 \quad j = 1, \dots, n-1 \quad n-1$

$i = 2 \quad j = 2, \dots, n-1 \quad n-2$

$\vdots$

$i = n-1 \quad j = n-1 \quad 1$

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

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

Sum of Arithmetic Sequence

$1 + 2 + 3 + \cdots + 9 + 10$

$(1+10) \cdot \frac{\cancel{10}\,10}{2} = ?$

$\underbrace{\boxed{I}}_{\text{first term}} \xrightarrow{+C} (I + \boxed{C}) \xrightarrow{+C} (I + 2 \cdot C) + \cdots + (I + (n-1) \cdot C)$

$I + 0 \cdot C$

constant

$$= \frac{[I + (I + (n-1) \cdot C)] \cdot n}{2}$$

n → # of terms

e.g. $\underbrace{1}_{\text{first term}} \xrightarrow{+1} 2 \xrightarrow{+1} 3 + \cdots + n$

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

↳ is $O(n^2)$

Lecture

Arrays vs. Linked Lists

Asymptotic Upper Bounds of Array Operations

$$a[i] = a\boxed{[\underline{i+1}]};$$

Pos: 4 $(= n^0 \cdot 4)$

object creation: O(1)

Inserting into an Array

copy input[0]
⋮
input[i-1] to result

where to insert.

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

1

j = 0..(i-1) 0..0 1

[i+1, n] = n-(i+1)+1 = n-i

copy a[i-1] to result a[i]

$O(i \cdot 1) = O(n)$ — worst case: i == n

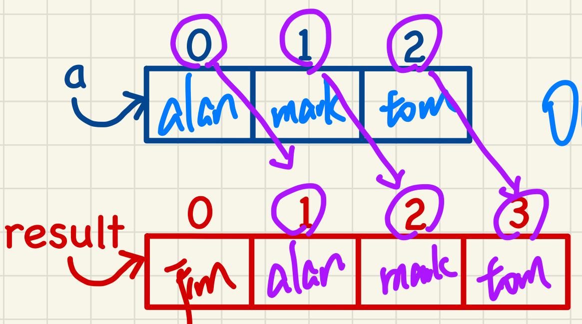
$O((n-i) \cdot 1) = O(n)$ — worst case: i == 0

1

a.length

Example:

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



$O(1+n+1+n+1)$

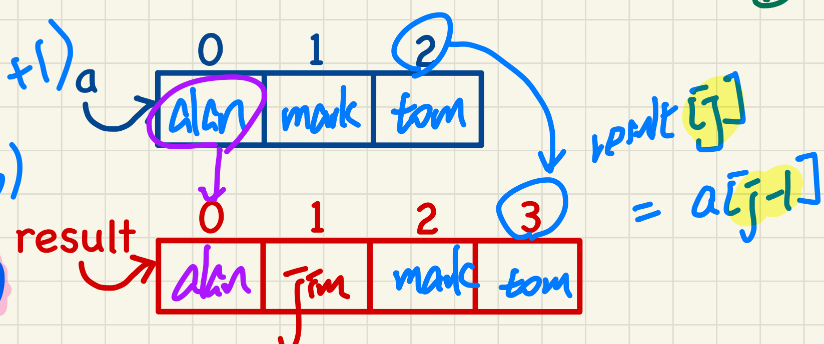
$= O(2n+3)$

$= O(n)$

Example:

result[0] = a[0];
result[1] = jim;
result[2] = a[1]
result[3] = a[2]

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



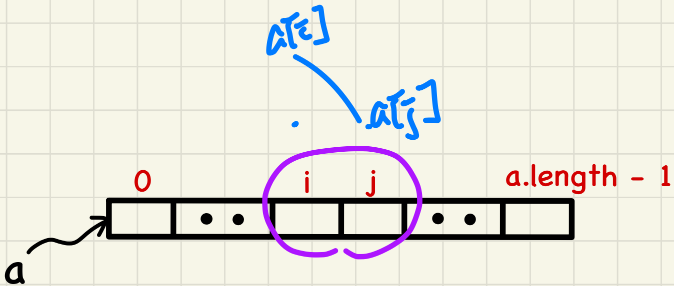
Exercise: insertAt({alan, mark, tom}, 3, jim, 3)

Lecture

Arrays vs. Linked Lists

Selection Sort vs. Insertion Sort

Sorting Orders of Arrays



non-descending

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

$$\neg(a[i] > a[j])$$

$$\equiv a[i] \leq a[j]$$