

Lecture 13

Thursday Oct. 19

Example: Determining Asymptotic Running Time

```
1 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]) {  $O(1)$   
5         return true; }  
6       j++; }  $\bar{i} = n-1$   
7     i++; }  $\bar{j} = 0 \dots n-1 (n)$   
8   return false; }
```

Body of loop takes $O(1)$ $\left[\begin{array}{l} \bar{i} = 0 \\ \bar{j} = 0 \dots n-1 \end{array} \right. (n)$

How many times? n^2 $\left[\begin{array}{l} \bar{i} = 1 \\ \bar{j} = 0 \dots n-1 \end{array} \right. (n)$

RT: $O(1) \neq n^2 \neq O(n^2)$

Example: Determining Asymptotic Running Time

```
1  sumMaxAndCrossProducts (int[] a, int n) {  
2  [int max = a[0];] O(1)  
3  [for(int i = 1; i < n; ) {  
4    [if (a[i] > max) { max = a[i]; } ] O(n)  
5  ]  
6  [int sum = max; ] O(1)  
7  [for (int j = 0; j < n; j ++ ) {  
8    [for (int k = 0; k < n; k ++ ) {  
9      sum += a[j] * a[k]; } } ] O(n2)  
10 [return sum; } ] O(1)
```

RT: $O(n + 1 + \cancel{n^2} + 1) = O(n^2)$.
dominates the RT

Example: Determining Asymptotic Running Time

```
1 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;  
}
```

$$\bar{i} = n-1$$

$$\bar{j} = n-1 \dots n-1 \quad (1)$$

$O(1)$

$$\bar{j} = \bar{i}$$

$$\frac{\bar{i} = 0}{\bar{j} = \frac{\bar{i}}{0}} \dots n \cdot (n)$$

How many iterations?

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

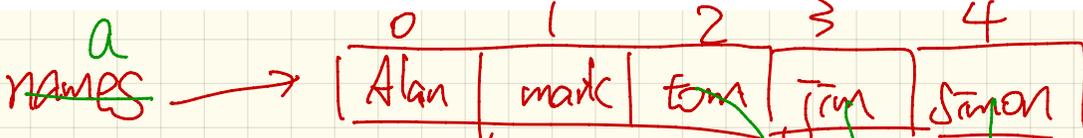
$$\frac{\bar{i} = 1}{\bar{j} = 1 \dots (n-1) \cdot (n-1)}$$

Thinking Time of Inserting into an Array [a, b]

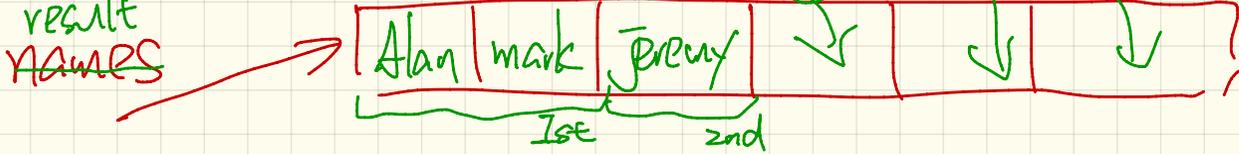
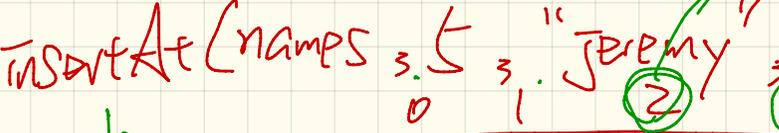
```

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

Annotations: $O(n)$ for the first loop, $O(1)$ for the second, $O(n)$ for the third. The total complexity is $O(n)$.

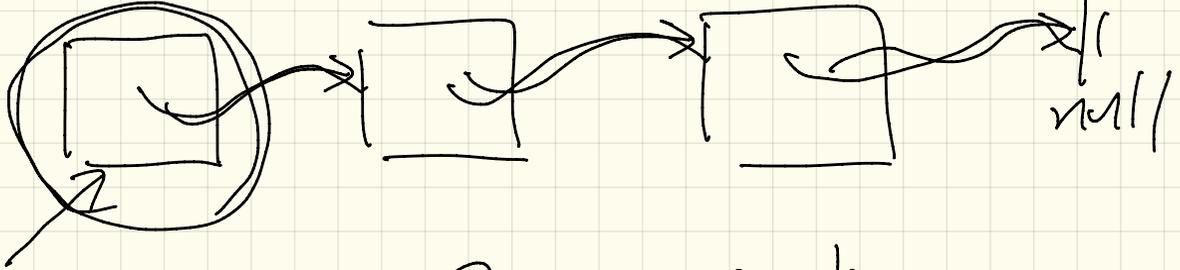
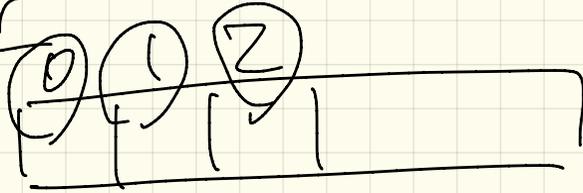


$result[0] = a[0]$

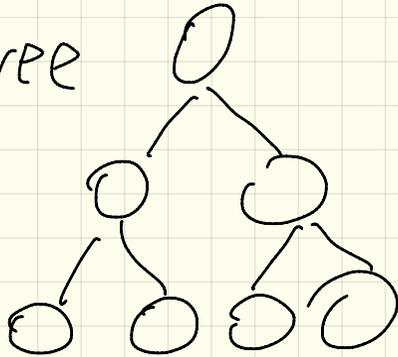


3rd: $j = i + 1$ to $n - 1$.
 Worst case: $i = 0$.
 # of iterations for 3rd step: $n - i - 1$.

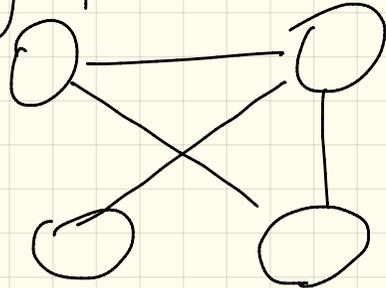
linear

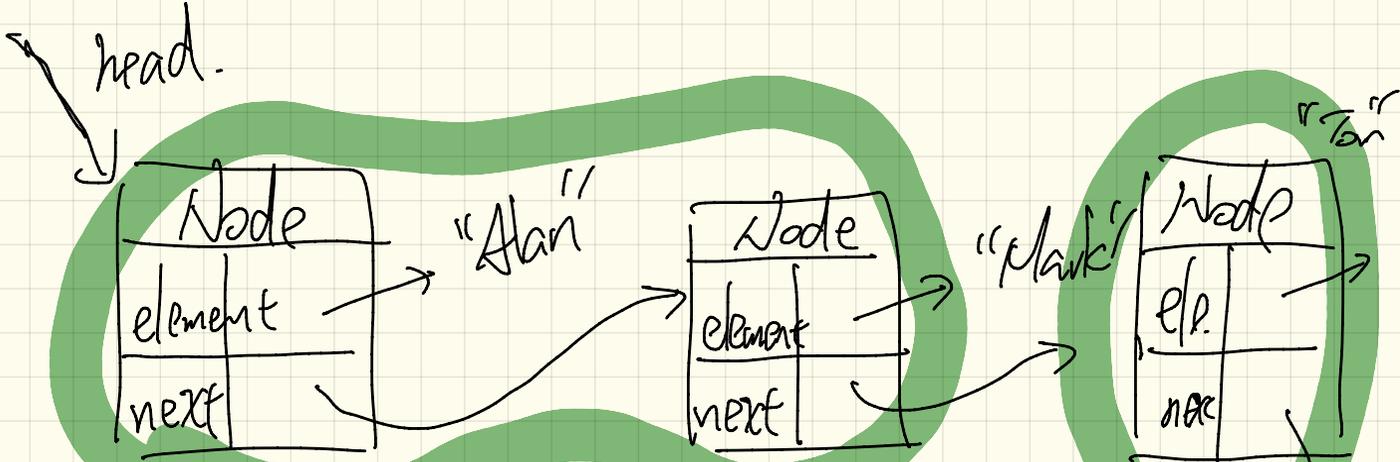


non-linear tree



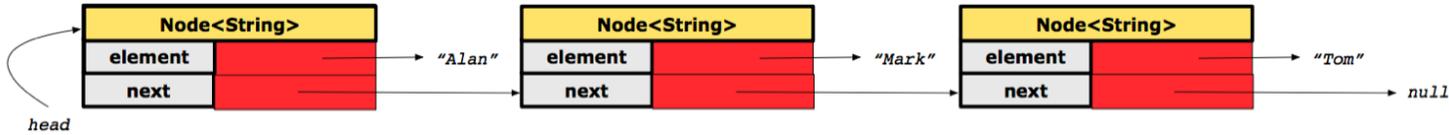
graph





a chain of two linked nodes

last nodes



```

public class Node {
    private String element;
    private Node next;
    public Node(String e, Node n) { element = e; next = n; }
    public String getElement() { return element; }
    public Node getNext() { return next; }
    public void setNext(Node n) { next = n; }
}
  
```

Handwritten annotations: "mark" with an arrow pointing to the `Node n` parameter in the constructor. A green circle highlights `next = n;` in the constructor.

Handwritten: `alan.next == mark`

Handwritten: `Node alan = new Node("Alan", mark)`
 ("Alan's mark")

Approach 1

```

Node tom = new Node("Tom", null);
Node mark = new Node("Mark", tom);
Node alan = new Node("Alan", mark);
  
```

Handwritten annotations: Circled numbers 1 through 5 are placed next to the corresponding lines of code.

