Announcements
  Thursday Extras,
  None next week

CSC 207 Notes
• Supplementary Problem 2—extra credit
  • now posted
  • if completed, must be submitted by 5:00 pm, Fri., Dec. 14
• Revision of Test 2, Problem 1, accepted if submitted by
  Wednesday's class meeting
• Test 2, Part B, deadline changed to start of Wednesday's class

Algorithmic Analysis
• Questions?
• Clicker questions today
Algorithmic Analysis

Suppose arrays a and b have been declared as int arrays as large as needed for the following.

Consider the following code segment:

```c
int i, j;
for (i = 0; i < n; i++) {
    b[i] = 0;
    for (j = 0; j <= i; j++)
        b[i] += a[j];
}
```

Determine Big-O for this code, and briefly justify your answer:

A. O(n) because 1 + 1 + ... + 1 (n times) = n  

B. O(n²), as 1 + 2 + ... + i + ... + n = n(n+1)/2  

C. O(n³), as 1² + 2² + ... + i² + ... + n² = n(n+1)(2n+1)/6  

D. O(n log n), as log 1 + log 2 + ... + log n has O(n log n)  

E. O(n² log n), as 1log 1 + 2log 2 + ... + nlog n has O(n² log n)  

*** ignoring constants and operations performed for lower Big-O
Algorithmic Analysis
Suppose arrays a and b have been declared as int arrays as large as needed for the following.

Consider the following code segment:
```java
int i, j;
for (i = 0; i < n; i++) {
    b[i] = 0;
    for (j = i; j <= i+10; j++)
        b[i] += a[j];
}
```

Determine Big-O for this code, and briefly justify your answer:
A. O(n) because $1 + 1 + \ldots + 1$ (n times) = n
B. O(n^2), as $1 + 2 + \ldots + i + \ldots + n = n(n+1)/2$
C. O(n^3), as $1^2 + 2^2 + \ldots + i^2 + \ldots + n^2 = n(n+1)(2n+1)/6$
D. O(n log n), as log 1 + log 2 + ... + log n has O(n log n)
E. O(n^2 log n), as 1\log 1 + 2\log 2 + ... + n\log n has O(n^2 \log n)

*** ignoring constants and operations performed for lower Big-O
Algorithmic Analysis

Suppose arrays a and b have been declared as int arrays as large as needed for the following.

Consider the following code segment:
```c
int i, j;
for (i = 0; i < n; i++) {
    b[i] = 0;
    for (j = 1; j <= i; j*=2)
        b[i] += a[j];
}
```

Determine Big-O for this code, and briefly justify your answer:

A. O(n²), as $1 + 2 + \ldots + i + \ldots + n = \frac{n(n+1)}{2}$

B. O(n³), as $1^2 + 2^2 + \ldots + i^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$

C. O(n log n), as $\log 1 + \log 2 + \ldots + \log n$ has O(n log n)

D. O(n² log n), as $1\log 1 + 2\log 2 + \ldots + n\log n$ has O(n² log n)

E. O(n) because (when $2^i = n$) $2^0 + 2^1 + \ldots + 2^i = 2n-1$

*** ignoring constants and operations performed for lower Big-O
Algorithmic Analysis
Suppose arrays a and b have been declared as int arrays as large as needed for the following. Consider the following code:

```c
int i, j;
for (i = 0; i < n; i++) {
    b[i] = 0;
    for (j = 1; j <= i; j*=2)
        b[i] += a[j];
    for (j = 1; j <= i; j++)
        b[i] += a[j];
}
```

Determine Big-O for this code, and briefly justify your answer:

A. $O(n^2)$, as $1 + 2 + \ldots + i + \ldots + n = n(n+1)/2$  
B. $O(n^3)$, as $1^2 + 2^2 + \ldots + i^2 + \ldots + n^2 = n(n+1)(2n+1)/6$  
C. $O(n \log n)$, as $\log 1 + \log 2 + \ldots + \log n$ has $O(n \log n)$  
D. $O(n^2 \log n)$, as $1\log 1 + 2\log 2 + \ldots + n\log n$ has $O(n^2 \log n)$  
E. $O(n)$ because (when $2^i = n$) $2^0 + 2^1 + \ldots + 2^i = 2n-1$  

*** ignoring constants and operations performed for lower Big-O
Algorithmic Analysis
Suppose arrays a and b have been declared as int arrays as large as needed for the following. Consider the following code:

```c
int i, j, k;
for (i = 0; i < n; i++) {
    b[i] = 0;
    for (j = 1; j <= i; j*=2)
        for (k = i+j; k <= i+j+17; j++)
            b[i] += a[k];
}
```

Determine Big-O for this code, and briefly justify your answer:

A. $O(n^2)$, as $1 + 2 + ... + i + ... + n = n(n+1)/2$
B. $O(n^3)$, as $1^2 + 2^2 + ... + i^2 + ... + n^2 = n(n+1)(2n+1)/6$
C. $O(n \log n)$, as $\log 1 + \log 2 + ... + \log n$ has $O(n \log n)$
D. $O(n^2 \log n)$, as $1\log 1 + 2\log 2 + ... + n\log n$ has $O(n^2 \log n)$
E. $O(n)$ because (when $2^i = n$) $2^0 + 2^1 + ... + 2^i = 2n-1$

*** ignoring constants and operations performed for lower Big-O
Algorithmic Analysis
Suppose arrays a and b have been declared as int arrays as large as needed for the following. Consider the following code:

```c
int i, j, k;
for (i = 0; i < n; i++) {
    b[i] = 0;
    for (j = 1; j <= i; j*=2)
        for (k = 0; k <= i; k++)
            b[i] += a[k];
}
```

Determine Big-O for this code, and briefly justify your answer:

A. $O(n^2)$, as $1 + 2 + \ldots + i + \ldots + n = n(n+1)/2$

B. $O(n^3)$, as $1^2 + 2^2 + \ldots + i^2 + \ldots + n^2 = n(n+1)(2n+1)/6$

C. $O(n \log n)$, as $\log 1 + \log 2 + \ldots + \log n$ has $O(n \log n)$

D. $O(n^2 \log n)$, as $1\log 1 + 2\log 2 + \ldots + n\log n$ has $O(n^2 \log n)$

E. $O(n)$ because (when $2^i = n$) $2^0 + 2^1 + \ldots + 2^i = 2n-1$

*** ignoring constants and operations performed for lower Big-O