Run Time

What factors affect the speed of the software?

1) CPU Speed
2) Amount of RAM
3) Compiler (C++ is faster than Java)

4) Algothrim used
5) Amount of input

} We control, dependent on application

Big-O Notation

Used by computer scientists to denote fastest growing term in algebraic calculation of exact run time

1a) $O(n^2)$
b) $15n + n\log n = O(n\log n)$ Ignore constants in high term
c) $O(n)$

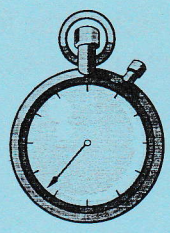
2a) $50n^0 = O(1)$ constant
b) $n \times n = n^2$, $O(n^2)$
c) $n + n = 2n$, $O(n)$
d) $n \times n \times n = n^3$, $O(n^3)$

3a) ~~N/A~~, LS = 8, BS = 4 (2^4)
b) LS = 32, BS = 6 (2^6)
c) LS = 64, BS = 7 (2^7)

Linear Search = LS
Binary Search = BS

Number of Searches
Binary: $\log_2 n$ (2^n)
Linear: $n/2$

Run Time of Popular
Sorting and Searching Algorithms

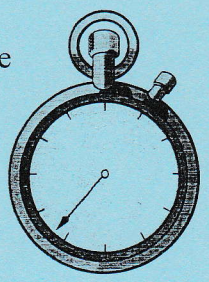


Algorithm	Run Time
Bubble Sort	$O(n^2)$
Selection Sort	$O(n^2)$
Linear Search	$O(n)$
Binary Search	$O(\log n)$

Changes in your input size n can make a BIG difference in the run time of your algorithm!

$\log n$	n	n^2
2	4	16
4	16	256
5	32	1,024
7	128	16,384
9	512	262,144

CSC 205 Worksheet on Run-Time Analysis



1. Suppose by careful measurement we have discovered a function that describes the precise running time of some algorithm. For each of the following such functions, describe the algorithmic running time in *Big-O* notation.

   a. $3n^2 + 3n + 7$
   b. (5 * n) * (3 + log n)
   c. (5n + 4) / 6

2. For each of the following program skeletons, describe the algorithmic execution time as a function of *n* using the *Big-O* Notation.

```
a.  for (int i = 1;  i <= 50; i++) {
        ......
    }

b.  for (int i = 1;  i <= n;  i++) {          n times  (nested, so ×)
      for (int j = 1;  j <= n;  j++)  {       n times
          . . . . . .
      }
    }

c.  for (int i = 1;  i <= n;  i++)  {         n times
        . . . . . .                                    (not nested, so +)
    }
    for (int j = n;  j >= 1;  j--)  {         n times
        . . . . . .
    }

d. for (int i = 1;  i <= n;  i++) {           n times
     for (int j = 1;  j <= n;  j++)  {        n times
       for (int k = 1;  k <= n;  k++) {       n times

       }
     }
   }
```

3. Calculate the average number of checks you would need to make of an array of each of the following sizes using the linear and binary search algorithms :

   a. 16
   b. 64
   c. 128