

Complexity

Goals

- How much storage is used?
 - space complexity
- What is the runtime?
 - runtime complexity

Two types of performance

- This algorithm takes N^2 steps to run
 - We care about this!
- This loop takes 10 instructions but I can rewrite it to take 9!
 - We don't care about this one in this class
 - Later in life, this will be important
 - Many people use this as an excuse to write bad code

Compiler

- `gcc -O# -o bin/runMe bin/driver.o bin/stack.o`
- `#` is zero to 3
 - zero: default, no speed optimization
 - 1, 2, 3 increasing levels of optimization (speed/size)
 - almost no chance of the debugger working

Algorithmic Complexity

- How does the runtime increase as the problem size increases?
- How do we measure the runtime?
 - language, machine independent!
- How do we measure the problem size?

Runtime Complexity

```
const int N = 500;  
int i;  
int aCounts[N];  
  
for (i = 0; i < N; ++ i)  
{  
    aCounts[i] = 0;  
}
```

- What value does the running time depend on?

Runtime Complexity

```
const int N = 500;  
int i, j;  
int aCounts[N][N];  
  
for (i = 0; i < N; ++i)  
{  
    for (j = 0; j < N; ++j)  
    {  
        aCounts[i][j] = 0;  
    }  
}
```

- What value does the running time depend on?

Formal Definition: Big-O

- Computational Complexity
 - number of steps related to some data size, N
 - number of items
 - $O(N)$
 - $O(N^2)$
- Growth rate!
- Algorithm:

Big-O

- Growth, not exact runtime
- Can't (always) tell which algorithm is faster
- Concerned with very large inputs
- Asymptotic algorithm analysis

Big-O notation

- Find a function, $g(n)$, that describes the execution time
- $O(g(n))$
- We only include the highest order terms
 - also ignore constants
- $x^2 + x + 1$
 - Which term dominates this equation (as x gets big)?

N? Complexity?

```
int isSorted (const int aNums[], int howmany)
{
    int bSorted;
    int i;

    bSorted = true;

    for (i = 0; i < (howmany - 1); ++i)
    {
        if (aNums[i] > aNums[i + 1])
        {
            bSorted = false;
        }
    }

    return bSorted;
}
```

Scenarios

- Best Case
- Average Case
- Worst Case

Categories

- $O(1)$ - constant
- $O(\log_2 N)$ - logarithmic
- $O(N)$ - linear
- $O(N \log_2 N)$ - log linear
- $O(N^2)$ - quadratic
- $O(N^3)$ - cubic
- $O(2^N)$ - exponential
- $O(N!)$ - factorial

Growth Rates

N	$\log_2 N$	$N \log_2 N$	N^2	N^3	2^N
2	1	2	4	8	4
4	2	8	16	64	16
8	3	24	64	512	256
16	4	64	256	4096	65536

Identify Big-O

	Average case				Worst case		
	Search	insert	Delete		Search	Insert	Delete
Unordered Array							
Ordered Array							
Singly Linked List							

Identify Big-O

Function	Best	Worst	Average
strlen			
strcmp			
strcat			
strappend			
strreverse			
strclear			
strcpy			

What is N?

```
typedef struct {  
    int length;  
    char data[1024];  
} String;
```

Formally

Function $f(n)$ is $O(g(n))$ iff there exist positive constants c and n_0 such that $f(n) \leq cg(n)$ for all n , where $n \geq n_0$.

```
for (i = 0; i < howmany; ++i)
{
    for (j = i + 1; j < howmany; ++j)
    {
        if(aNums[i] < aNums[j])
        {
            temp = aNums[i];
            aNums[i] = aNums[j];
            aNums[j] = temp;
        }
    }
}
```

// how many times is the if() test executed?