Name: ______________________________

Today's Date: ______________________________

Today's Goals

- Identify when asymptotoic analysis is appropriate.
- Explain what asymptotic analysis does and does not tell us.
- Build familiarity with O, Ω, Θ.
- Differentiate between worst, best, and average case complexity.

Today's Question(s)

How many times does Bessie moo?

```
for (size_t i=0; i<n; ++i) {
  for (size_t j=0; i < p; ++j) {
    bessie.moo()
  }
}
```

Lingering Questions

Learning Goals

- Identify when asymptotic analysis is appropriate.
- Explain what asymptotic analysis does (and doesn't) tell us.

Asymptotic Analysis

Answers an abstract question about an algorithm:

- How do costs *scale* as input sizes become arbitrarily *large*?

Suppose a function with input size n takes $63n$ steps when it runs.

What is the ratio?

$$\frac{\#\text{steps for some input}}{\#\text{steps on some input twice as big}}$$

Asymptotic Analysis

Answers an abstract question about an algorithm:

- How do costs *scale* as input sizes become arbitrarily *large*?

Suppose a function with input size n takes $5n^3$ steps when it runs.

What is the ratio?

$$\frac{\#\text{steps for some input}}{\#\text{steps on some input three times as big}}$$

Asymptotic Analysis

Answers an abstract question about an algorithm:

- How do costs *scale* as input sizes become arbitrarily *large*?

Suppose a function with input size n takes $n^3 + 17$ steps when it runs.

What is the ratio

$$\frac{\#\text{steps for some input}}{\#\text{steps on some input three times as big}}$$

as n gets very large?

Comparing Complexity of Functions

Function 1: T(n) = 7n^2 + n + 2

Function 2: T(n) = 3n^2 - 1

Simplify

1. $O(3n^2 + 2n + 2 + \cos(\pi n))$
2. $O(\log_{10}(n^3))$
3. $O(5n^{1.5} + 2n \log n)$
4. $O(n^2 + 2^n)$

Counting Steps

We can count "steps" rather than "instructions" or "clock cycles"

- Both give the same asymptotic result
- A step can be a lot of work, as long as it's bounded by a constant.

```
const int LINE_LENGTH = 80;
for (char c = 'a'; c <= 'z'; ++c) {
  for(size_t i = 0; i < LINE_LENGTH; ++i){
    std::cout << c << "\n";
  }
}
```

Big-O: Graphical Definition

Big-O: Math Definition

Big-O, Big-Ω, Big-Θ

What does n mean in $O(n)$?

Algorithms A and B each take a vector of nonempty strings as input.

- A prints all the strings to the screen.
- B counts how many strings begin with a capital letter.

Suggest appropriate definitions for “input size” n.

Counting steps

Suppose we have a fixed sorting algorithm for integer arrays, and we all agree on what counts as a "step" in the algorithm.

Let n be the number of integers we are sorting.

Is this enough information to decide exactly how many steps are required to sort the input?

Best Case vs. Worst Case

Worst-case analysis: for each n, pick the hardest input of size n.

Best-case analysis: for each n, pick the easiest input of size n.

Best Case vs. Worst Case

Worst-case analysis: for each n, pick the hardest input of size n.

Best-case analysis: for each n, pick the easiest input of size n.

Important:

- We can talk about worst-case O and Ω
- We can talk about best-case O and Ω

Tracing Insertion Sort

		a[]										
i	j	0	1	2	3	4	5	6	7	8	9	10
		S	O	R	T	E	X	A	M	P	L	E
1	0	O	S	R	T	E	X	A	M	P	L	E
2	1	O	R	S	T	E	X	A	M	P	L	E
3	3	O	R	S	T	E	X	A	M	P	L	E
4	0	E	O	R	S	T	X	A	M	P	L	E
5	5	E	O	R	S	T	X	A	M	P	L	E
6	0	A	E	O	R	S	T	X	M	P	L	E
7	2	A	E	M	O	R	S	T	X	P	L	E
8	4	A	E	M	O	P	R	S	T	X	L	E
9	2	A	E	L	M	O	P	R	S	T	X	E
10	2	A	E	E	L	M	O	P	R	S	T	X
		A	E	E	L	M	O	P	R	S	T	X

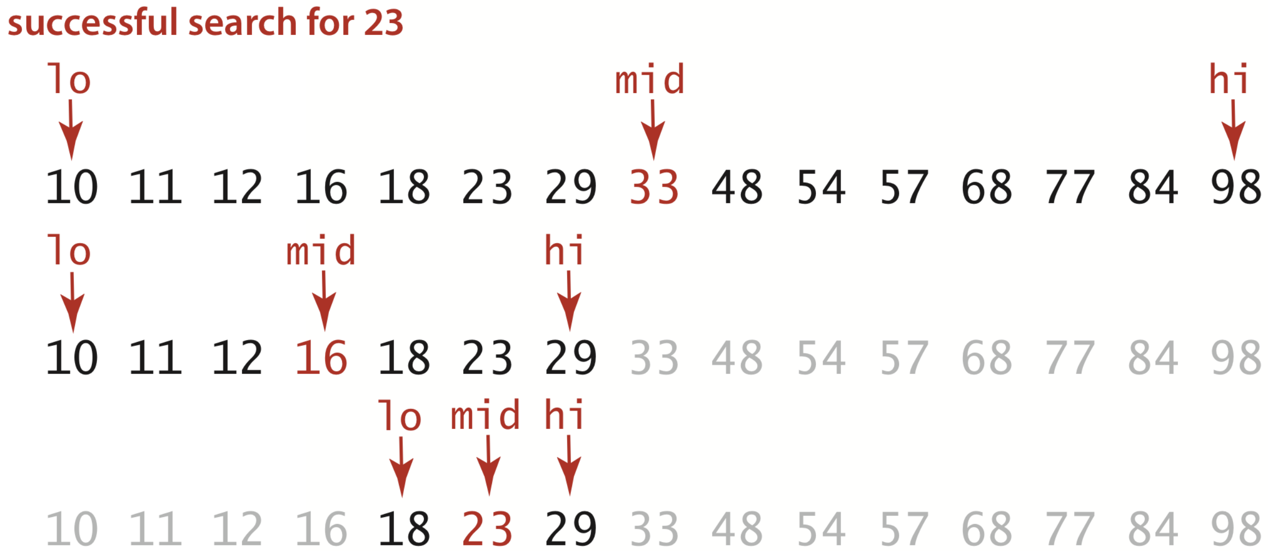
entries in gray do not move

entry in red is a[j]

entries in black moved one position right for insertion

Trace of insertion sort (array contents just after each insertion)

Tracing Binary Search



(from Sedgewick & Wayne, *Algorithms*, 2014)

True or False?

- $\Theta(n \log n)$ means "proportional to n log n".
- $f \in \Theta(n^2)$ means "f is a closed-form polynomial whose highest order term is n^2.
- If two algorithms are $\Theta(n \log n)$, it doesn't matter which we use.
- We should replace $\Theta(n^2)$ algorithms with $\Theta(n \log n)$ algorithms wherever possible.
- Asymptotic analysis is useless.

Name: ____

Today's Date: ____

Today's Goals

- Identify when asymptotoic analysis is appropriate.
- Explain what asymptotic analysis does and does not tell us.
- Build familiarity with O, Ω, Θ.
- Differentiate between worst, best, and average case complexity.

Today's Question(s)

How many times does Bessie moo?

```
for (size_t i=0; i<n; ++i) {
  for (size_t j=0; i < p; ++j) {
    bessie.moo()
  }
}
```

Lingering Questions

Learning Goals

- Identify when asymptotic analysis is appropriate.
- Explain what asymptotic analysis does (and doesn't) tell us.

Asymptotic Analysis

Answers an abstract question about an algorithm:

- How do costs *scale* as input sizes become arbitrarily *large*?

Suppose a function with input size n takes $63n$ steps when it runs.

What is the ratio?

$$\frac{\#\text{steps for some input}}{\#\text{steps on some input twice as big}}$$

Asymptotic Analysis

Answers an abstract question about an algorithm:

- How do costs *scale* as input sizes become arbitrarily *large*?

Suppose a function with input size n takes $5n^3$ steps when it runs.

What is the ratio?

$$\frac{\#\text{steps for some input}}{\#\text{steps on some input three times as big}}$$

Asymptotic Analysis

Answers an abstract question about an algorithm:

- How do costs *scale* as input sizes become arbitrarily *large*?

Suppose a function with input size n takes $n^3 + 17$ steps when it runs.

What is the ratio

$$\frac{\#\text{steps for some input}}{\#\text{steps on some input three times as big}}$$

as n gets very large?

Comparing Complexity of Functions

Function 1: T(n) = 7n^2 + n + 2

Function 2: T(n) = 3n^2 - 1

Simplify

1. $O(3n^2 + 2n + 2 + \cos(\pi n))$
2. $O(\log_{10}(n^3))$
3. $O(5n^{1.5} + 2n \log n)$
4. $O(n^2 + 2^n)$

Counting Steps

We can count “steps” rather than “instructions” or “clock cycles”

- Both give the same asymptotic result
- A step can be a lot of work, as long as it’s bounded by a constant.

```
const int LINE_LENGTH = 80;
for (char c = 'a'; c <= 'z'; ++c) {
  for(size_t i = 0; i < LINE_LENGTH; ++i){
    std::cout << c << "\n";
  }
}
```

Big-O: Graphical Definition

Big-O: Math Definition

Big-O, Big-Ω, Big-Θ

What does n mean in $O(n)$?

Algorithms A and B each take a vector of nonempty strings as input.

- A prints all the strings to the screen.
- B counts how many strings begin with a capital letter.

Suggest appropriate definitions for "input size" n.

Counting steps

Suppose we have a fixed sorting algorithm for integer arrays, and we all agree on what counts as a "step" in the algorithm.

Counting steps

Suppose we have a fixed sorting algorithm for integer arrays, and we all agree on what counts as a "step" in the algorithm.

Let n be the number of integers we are sorting.

Counting steps

Suppose we have a fixed sorting algorithm for integer arrays, and we all agree on what counts as a "step" in the algorithm.

Let n be the number of integers we are sorting.

Is this enough information to decide exactly how many steps are required to sort the input?

Best Case vs. Worst Case

Worst-case analysis: for each n, pick the hardest input of size n.

Best-case analysis: for each n, pick the easiest input of size n.

Best Case vs. Worst Case

Worst-case analysis: for each n, pick the hardest input of size n.

Best-case analysis: for each n, pick the easiest input of size n.

Important:

- We can talk about worst-case O and Ω
- We can talk about best-case O and Ω

Tracing Insertion Sort

a[]

i	j	0	1	2	3	4	5	6	7	8	9	10
		S	O	R	T	E	X	A	M	P	L	E
1	0	O	S	R	T	E	X	A	M	P	L	E
2	1	O	R	S	T	E	X	A	M	P	L	E
3	3	O	R	S	T	E	X	A	M	P	L	E
4	0	E	O	R	S	T	X	A	M	P	L	E
5	5	E	O	R	S	T	X	A	M	P	L	E
6	0	A	E	O	R	S	T	X	M	P	L	E
7	2	A	E	M	O	R	S	T	X	P	L	E
8	4	A	E	M	O	P	R	S	T	X	L	E
9	2	A	E	L	M	O	P	R	S	T	X	E
10	2	A	E	E	L	M	O	P	R	S	T	X
		A	E	E	L	M	O	P	R	S	T	X

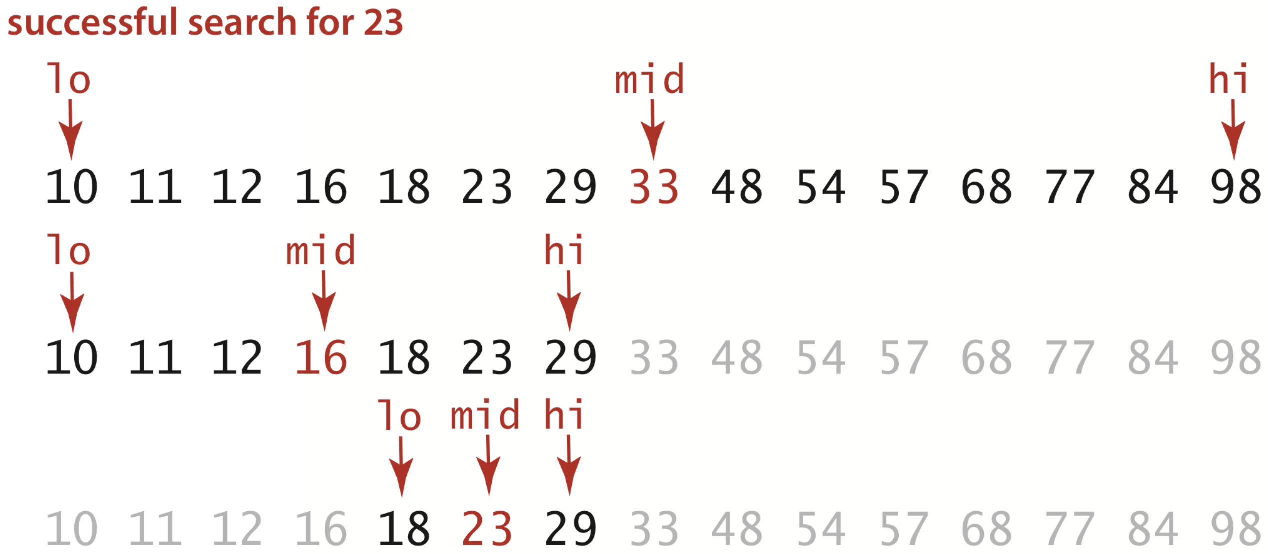
entries in gray do not move

entry in red is a[j]

entries in black moved one position right for insertion

Trace of insertion sort (array contents just after each insertion)

Tracing Binary Search



(from Sedgewick & Wayne, *Algorithms*, 2014)

True or False?

- $\Theta(n \log n)$ means "proportional to n log n".
- $f \in \Theta(n^2)$ means "f is a closed-form polynomial whose highest order term is n^2.
- If two algorithms are $\Theta(n \log n)$, it doesn't matter which we use.
- We should replace $\Theta(n^2)$ algorithms with $\Theta(n \log n)$ algorithms wherever possible.
- Asymptotic analysis is useless.

Exercise 1

```
int lowerBound(const char& x, const vector<char>& a, int first, int last)
{
    while( first < last ) {
        size_t mid = ( first + last ) / 2;

        // Print value of first, last, mid, and (a[mid] < x) here

        if( a[ mid ] < x ) {
            first = mid + 1;
        } else {
            last = mid;
        }
    }
    return last;
}
```

```
lowerbound(P, {A, E, F, H, I, L, N, P}, 0, 8):
Iterating in while. first = 0, last = 8, mid = 4: "I" < "P" => true
Iterating in while. first = 5, last = 8, mid = 6: "N" < "P" => true
Iterating in while. first = 7, last = 8, mid = 7: "P" < "P" => false
lowerBound = 7

lowerbound(A, {A, E, F, H, I, L, N, P}, 0, 8):
Iterating in while. first = 0, last = 8, mid = 4: "I" < "A" => false
Iterating in while. first = 0, last = 4, mid = 2: "F" < "A" => false
Iterating in while. first = 0, last = 2, mid = 1: "E" < "A" => false
Iterating in while. first = 0, last = 1, mid = 0: "A" < "A" => false
lowerBound = 0

lowerbound(O, {A, E, F, H, I, L, N, P}, 0, 8):
Iterating in while. first = 0, last = 8, mid = 4: "I" < "O" => true
Iterating in while. first = 5, last = 8, mid = 6: "N" < "O" => true
Iterating in while. first = 7, last = 8, mid = 7: "P" < "O" => false
lowerBound = 7
```

Exercise 2

```
int lowerBound(const char& x, const vector<char>& a, int first, int last)
{
    if ( first < last ) {
        size_t mid = (first + last) / 2;

        if( a[ mid ] < x ) {




        } else { // a[ mid ] >= x




        }
    } else { // first >= mid
```

```
    }
}
```