Running times

4/13/26

Administrivia

- HW 3 (testing and implementing list methods) due Wednesday (4/15)

- Exam 1 on Friday (4/17)
  - In-class, open-notes, on-paper exam
  - Codingbat-like problems, testing, array-based list methods, linked list methods, running times
  - Similar to, but probably shorter than sample exam

Running times

- We have two different implementations for list
  - Turns out that they are each faster for some operations
  - Need a way to compare them (and other decisions that affect running time performance)

Running times
(basic version)

- Constant time
  - Method takes the same amount of time no matter how much is in the List

- Linear time
  - Method takes time proportional to the number of items in the List

(For both, assume code goes through longest path)

What is the running time of the array-based implementation of get?

A. Constant time
B. Linear time

What is the running time of the array-based implementation of get?

A. Constant time
B. Linear time

What is the running time of the array-based implementation of remove(i), i.e. removing the value at index i?

A. Constant time
B. Linear time

What is the running time of the array-based implementation of remove(i), i.e. removing the value at index i?

A. Constant time
B. Linear time

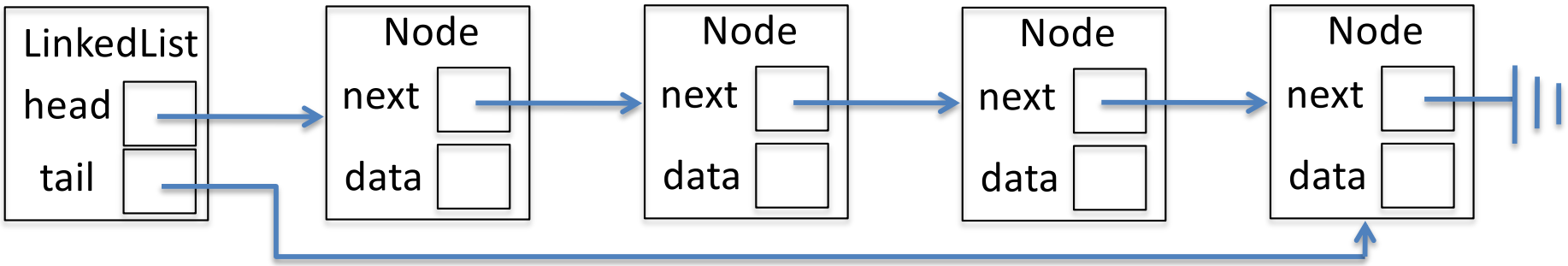
What is the running time of the linked-memory implementation of add for List?

A. Constant time
B. Linear time

What is the running time of the linked-memory implementation of add for List?

A. Constant time
B. Linear time

Adding a tail pointer



"Tail pointer" (tail reference) refers to last Node in the list

Helps adding to the end (now constant time)

Doesn't help removing from the end (still linear)

Doubly-linked lists

- Each node also stores a reference to the previous Node
- Allows insertion of new nodes before or after any node to which you have a reference

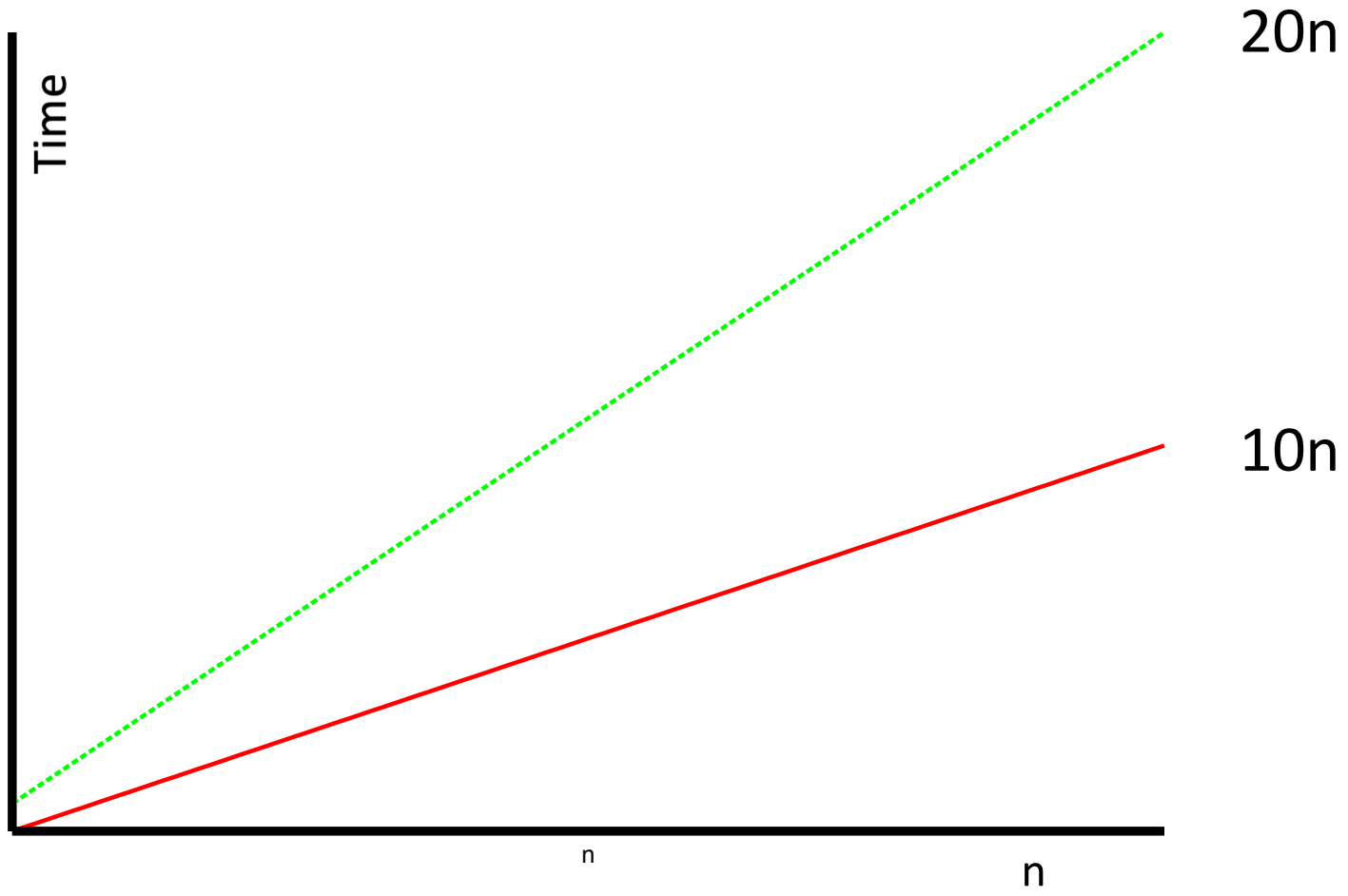
Doubly-linked list

```
public class Node {
    T val;
    Node next;
    Node prev;
}
```

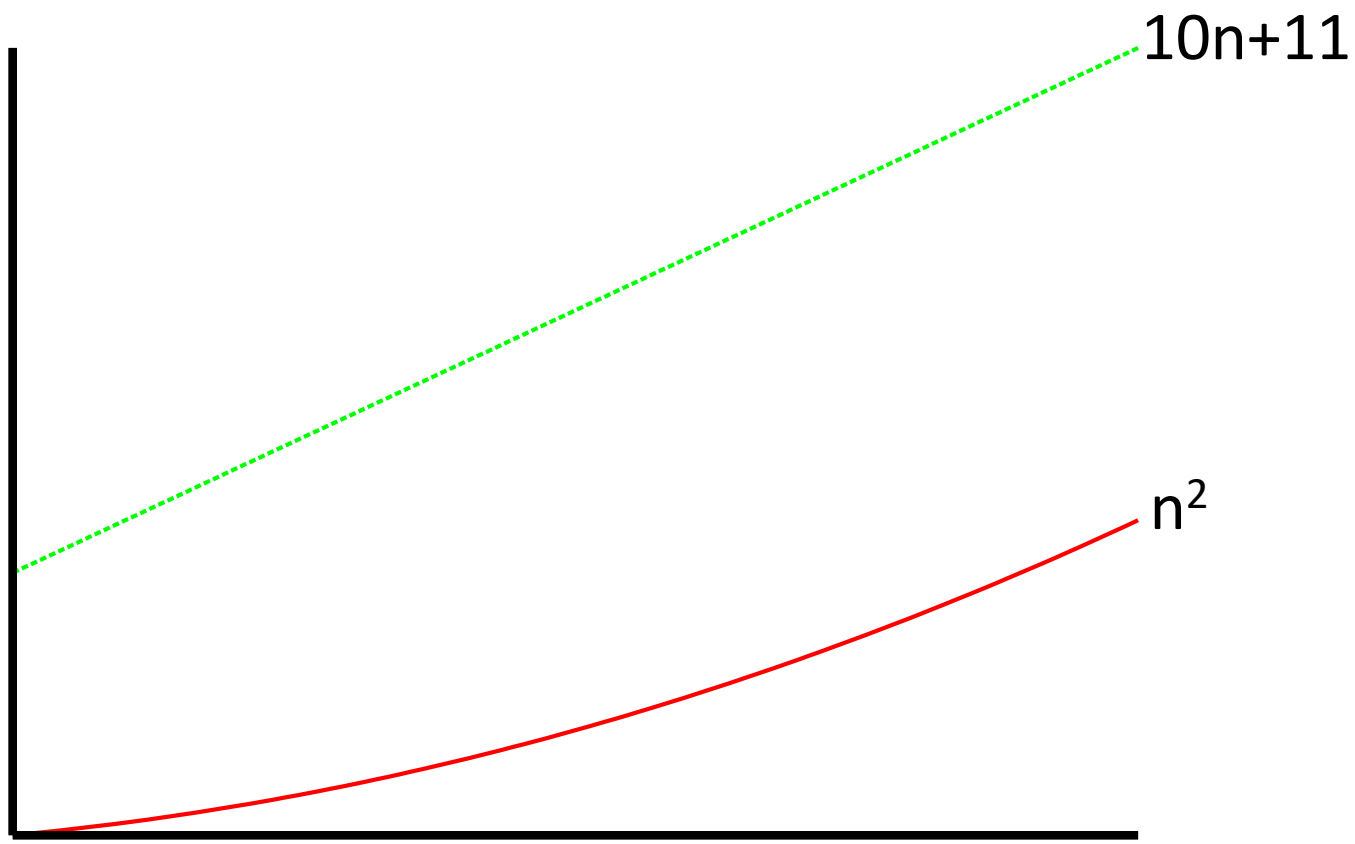
Advanced running times: How many lines of code are executed?

	Statements	Cost
1	float findAvg (int []grades){	
2	float sum = 0;	1
3	int count = 0;	1
4	while (count < grades.length) {	$n + 1$
5	sum += grades[count];	n
6	count++;	n
7	}	
8	if (grades.length > 0)	1
9	return sum / grades.length;	
10	else	1
11	return 0.0f;	
12	}	
ALL		**3n+5**

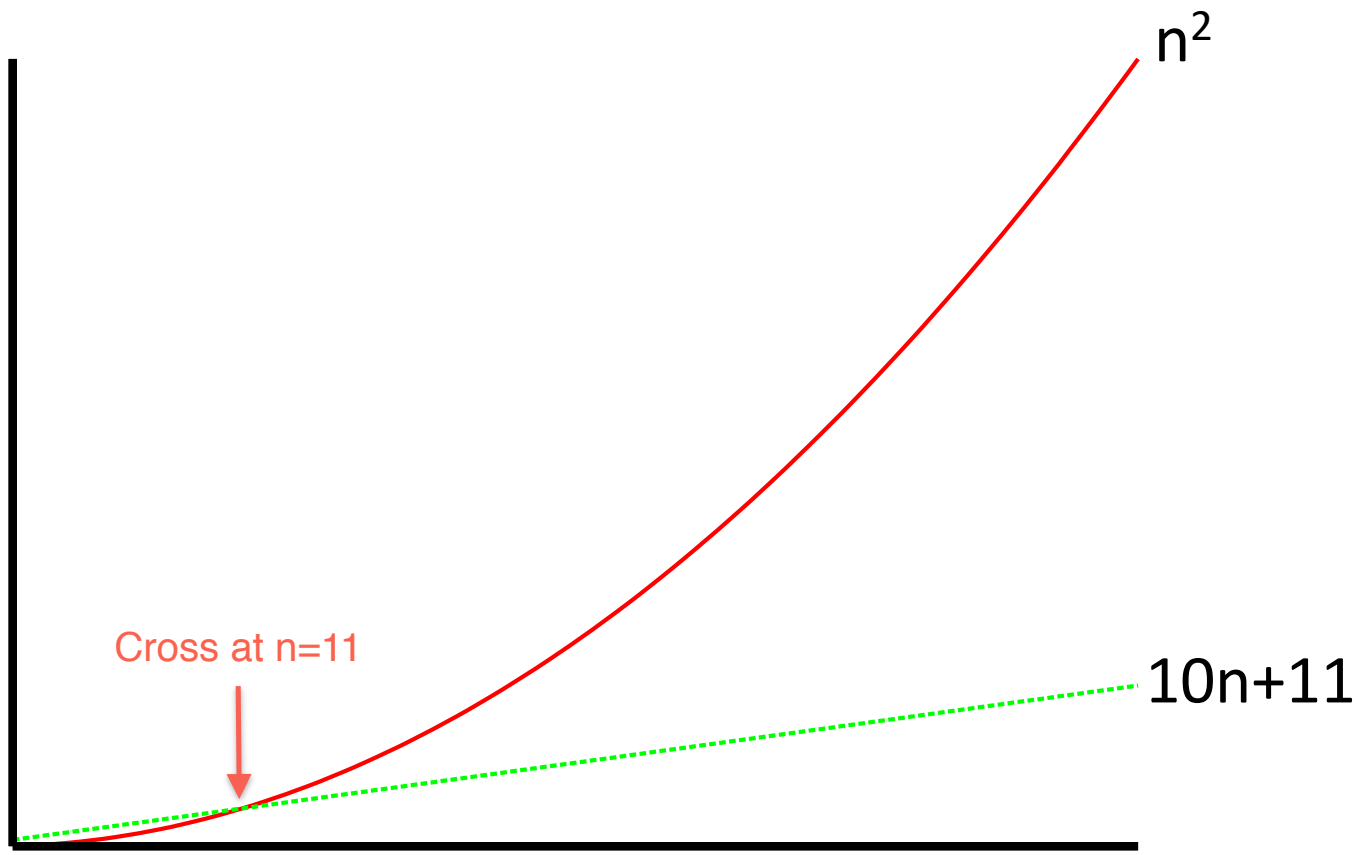
Is 10n or 20n bigger?



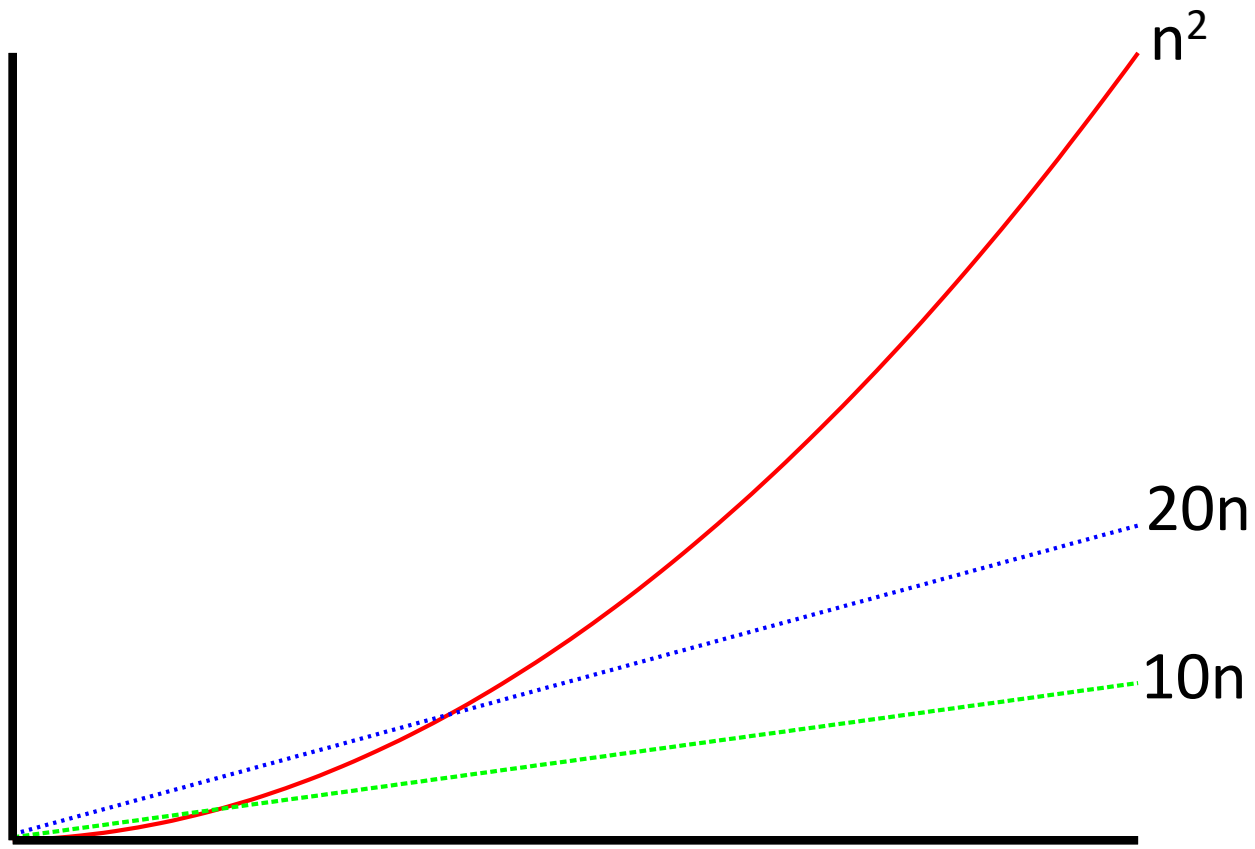
Is n^2 or 10n+11 bigger?



n^2 vs. 10n+11 revisited



10n vs. 20n revisited (with n^2)



Numbers of operations for different rates of growth

1,000,000	n	n^2	n^3	2^n
1,000,000	1	1	1	2
1,000,000	10	100	1,000	1,024

Numbers of operations for different rates of growth

1,000,000	n	n^2	n^3	2^n
1,000,000	1	1	1	2
1,000,000	10	100	1,000	1,024
1,000,000	100	10,000	1,000,000	~10^{30}

Numbers of operations for different rates of growth

1,000,000	n	n^2	n^3	2^n
1,000,000	1	1	1	2
1,000,000	10	100	1,000	1,024
1,000,000	100	10,000	1,000,000	$\sim 10^{30}$
1,000,000	1,000	1,000,000	1,000,000,000	$\sim 10^{300}$
1,000,000	10,000	100,000,000	10^{12}	Not really worth calculating...
1,000,000	100,000	10,000,000,000	10^{15}	
1,000,000	1,000,000	10^{12}	10^{18}	
1,000,000	10,000,000	10^{14}	10^{21}	
1,000,000	100,000,000	10^{16}	10^{24}	
1,000,000	1,000,000,000	10^{18}	10^{27}	
1,000,000	10,000,000,000	10^{20}	10^{30}	

Want to ignore

a) multiplicative constants

b) behavior at “small” n

Want to ignore

a) multiplicative constants

b) behavior at “small” n

Big-O notation:

f(n) = O(g(n)) if there exist constants n_0 and c such that $f(n) \leq cg(n)$ for all $n \geq n_0$

Which O( ) most accurately characterizes the growth rate of the following function's running time?

```
int maxDifference(int[] arr){
    max = 0;
    for (int i=0; i<arr.length;  i++) {
        for (int j=0; j<arr.length; j++) {
            if (arr[i] - arr[j] > max)
                max = arr[i] - arr[j];
        }
    }
    return max;
}
```

A. $f(n) = O(2^n)$
B. $f(n) = O(n^2)$
C. $f(n) = O(n)$
D. $f(n) = O(n^3)$
E. Not exactly one

(assume n = arr.length)

Which O( ) most accurately characterizes the growth rate of the following function's running time?

```
int maxDifference(int[] arr){
    max = 0;
    for (int i=0; i<arr.length;  i++) {
        for (int j=0; j<arr.length; j++) {
            if (arr[i] - arr[j] > max)
                max = arr[i] - arr[j];
        }
    }
    return max;
}
```

A. $f(n) = O(2^n)$
B. $f(n) = O(n^2)$
C. $f(n) = O(n)$
D. $f(n) = O(n^3)$
E. Not exactly one

(assume n = arr.length)

Which O( ) most accurately characterizes the growth rate of the following function's running time?

```
public int within10dups(int[] arr) {
    int cnt = 0;
    for(int i=0; i < arr.length; i++)
        for(int j=1; j <= 10; j++)
            if((j+i < arr.length) && (arr[i] == arr[i+j]))
                cnt++;
    return cnt;
}
```

A. $f(n) = O(2^n)$
B. $f(n) = O(n^2)$
C. $f(n) = O(n)$
D. $f(n) = O(n^3)$
E. Not exactly one of the above

(assume n = arr.length)

Which O( ) most accurately characterizes the growth rate of the following function's running time?

```
public int within10dups(int[] arr) {
    int cnt = 0;
    for(int i=0; i < arr.length; i++)
        for(int j=1; j <= 10; j++)
            if((j+i < arr.length) && (arr[i] == arr[i+j]))
                cnt++;
    return cnt;
}
```

A. $f(n) = O(2^n)$
B. $f(n) = O(n^2)$
C. $f(n) = O(n)$
D. $f(n) = O(n^3)$
E. Not exactly one of the above

(assume n = arr.length)

Which O( ) most accurately characterizes the growth rate of the following function's running time?

```
public int something_fun(int[] arr) {
    int cnt = 0;
    for(int i=0; i < arr.length; i=i+5)
        for(int j=arr.length-1; j>=0; j--)
            if(arr[i] == arr[j])
                cnt++;
    return cnt;
}
```

A. $f(n) = O(2^n)$
B. $f(n) = O(n^2)$
C. $f(n) = O(n)$
D. $f(n) = O(n^3)$
E. Not exactly one of the above

(assume n = arr.length)

Which O( ) most accurately characterizes the growth rate of the following function's running time?

```
public int something_fun(int[] arr) {
    int cnt = 0;
    for(int i=0; i < arr.length; i=i+5)
        for(int j=arr.length-1; j>=0; j--)
            if(arr[i] == arr[j])
                cnt++;
    return cnt;
}
```

A. $f(n) = O(2^n)$
B. $f(n) = O(n^2)$
C. $f(n) = O(n)$
D. $f(n) = O(n^3)$
E. Not exactly one of the above

(assume n = arr.length)

Let $f(n) = 2^n + 14n^2 + 4n^3$

Which of the following is true?

A. $f(n) = O(2^n)$
B. $f(n) = O(n^2)$
C. $f(n) = O(n)$
D. $f(n) = O(n^3)$
E. Not exactly one of the above

Let $f(n) = 2^n + 14n^2 + 4n^3$

Which of the following is true?

A. $f(n) = O(2^n)$
B. $f(n) = O(n^2)$
C. $f(n) = O(n)$
D. $f(n) = O(n^3)$
E. Not exactly one of the above

Let $f(n) = 100$

Which of the following is true?

A. $f(n) = O(2^n)$
B. $f(n) = O(n^2)$
C. $f(n) = O(n)$
D. More than one of the above
E. None of the above

Let $f(n) = 100$

Which of the following is true?

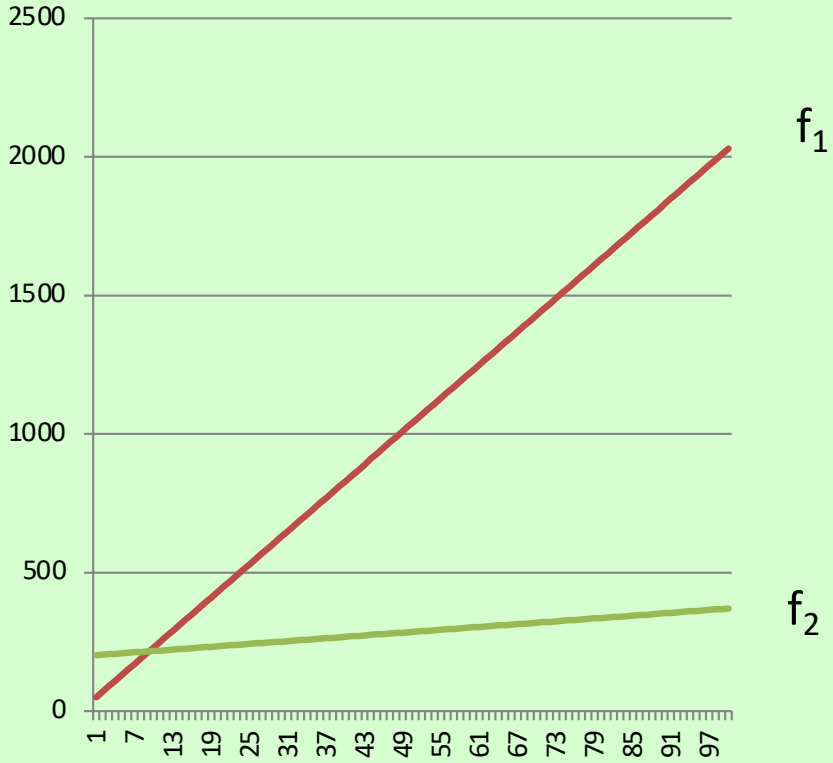
A. $f(n) = O(2^n)$
B. $f(n) = O(n^2)$
C. $f(n) = O(n)$
D. More than one of the above (all of them)
E. None of the above

f_2 is $O(f_1)$

A. TRUE

B. FALSE

Why or why not?

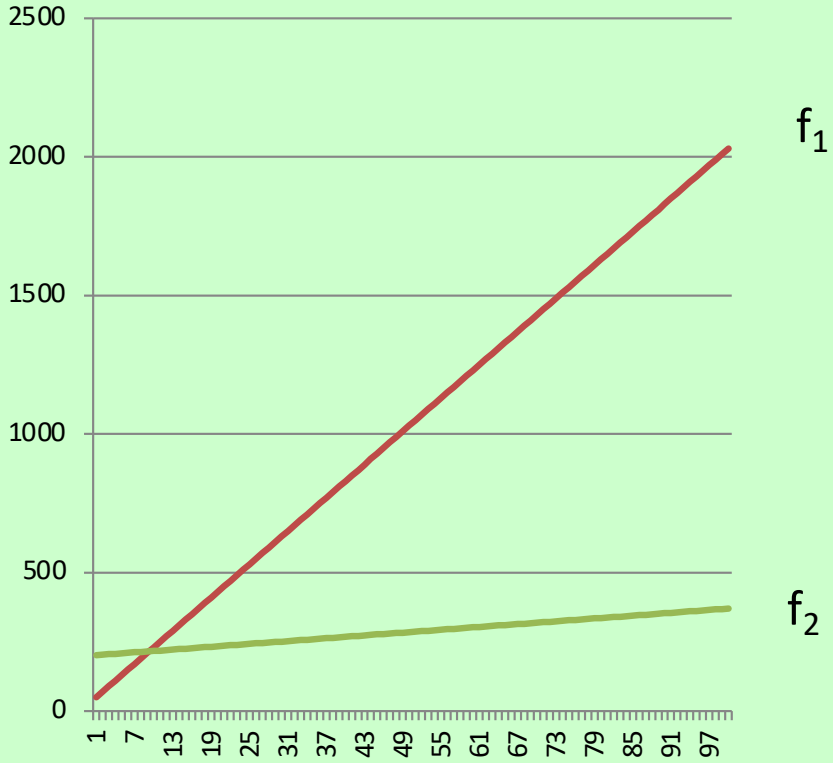


f_2 is $O(f_1)$

A. TRUE

B. FALSE

Why or why not?

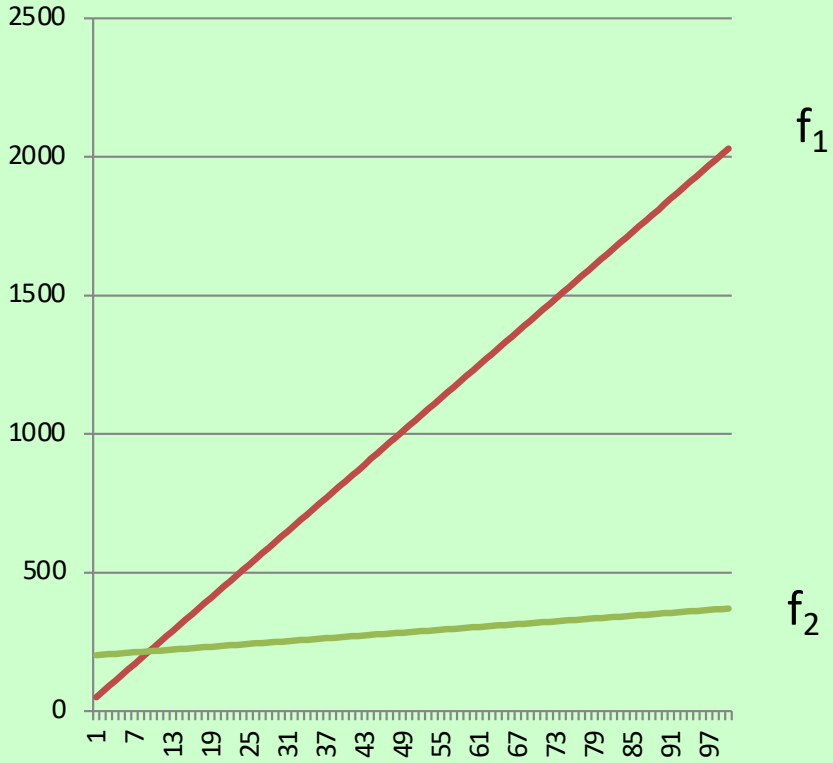


f_1 is $O(f_2)$

A. TRUE

B. FALSE

Why or why not?



f_1 is $O(f_2)$

A. TRUE

B. FALSE

Why or why not?