Lecture 04: `List` Performance

Announcements

1. Gradescope HW02 Link Later Today
2. Attendance and COVID
   - College masking policies strictly enforced
     - no food/drink in class
   - sessions recorded, notes posted
   - **do not come to class if sick/exposed**
   - risk = shared risk

Overview

1. Examining `ArraySimpleList`
2. Testing `ArraySimpleList` Performance
3. Modifying `ArraySimpleList`
4. `LinkedSimpleList` Implementation → linked list imp.
5. Other List-like ADTs
   - Double-ended Queues (Deques)
   - Queues
   - Stacks

Interfaces: Specification — what actions must be performed to implement an interface

arr :

0	1	2	3	4	5		
1	2	3	4	5	6	7	8

Remove(arr)

SimpleList interface

ArraySimpleList

LinkedSimpleList

SimpleList<Integer> list = new ??

Examining `ArraySimpleList`

Last Time

`List` ADT to represent lists of elements

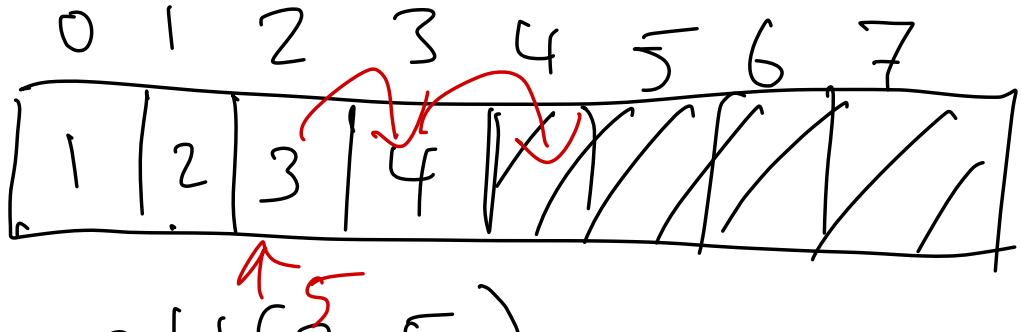
- `size()` — # elements in list
- `isEmpty()` — is size 0?
- `get(i)` — return ith element
- `set(i, y)` — sets val of ith elt
- `add(i, y)` — inserts at index
- `remove(i)` — remove and return elt. @ i

Discussed implementing `List` using an array to store contents

- `ArraySimpleList.java`

"1, 2, 3, 4"

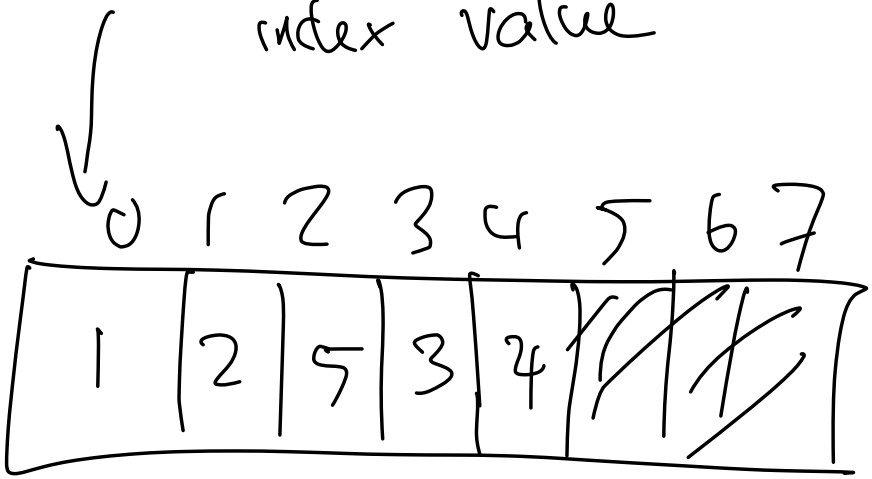
Representation of `List` as Array



(size = ~~4~~ 5)

How to add(2, 5)

index value



size = 5

Challenges

- arrays have fixed size
  - ADT requires operations to succeed without pre-specifying size
  - must “resize” the array (create new array and copy contents)

- arrays do not natively support insert/remove
  - must implement ourselves

Adding and Removing?

Starting from 1, 2, 4, 5, how to…

…`add(2, 3)`?

…`remove(0)`?

Look at the code!

- `ArraySimpleList.java`
- `SimpleList.java`
- `SimpleListTester.java`

Java “Feature”: No Generic Arrays

Does not compile:

```
E[] contents;
```

Compiles, but compiler complains

```
Object[] contents
...
E element = (E) contents[i];
```

Compiles, no complaints

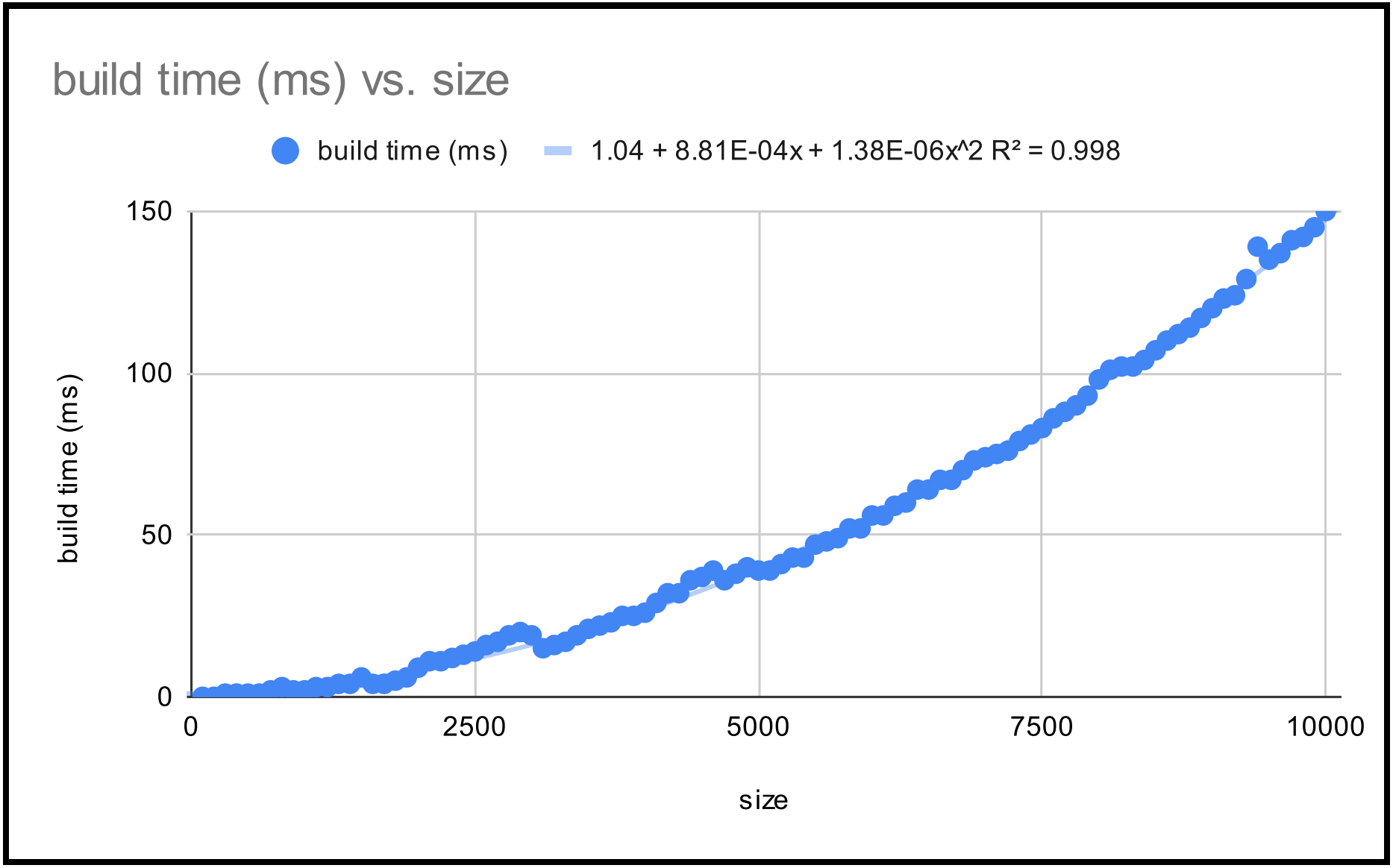
```
Object[] contents
...
@SuppressWarnings("unchecked")
E element = (E) contents[i];
```

Testing `ArraySimpleList` Performance

Test Time to Build a Large List

- `ListBuildTimer.java`
- `RunTimer.java`
- `CSVWriter.java`

Running Times



Question

Are we happy with this performance?

Modifying `ArraySimpleList`

An Issue

Every time we add an element, we copy entire array contents!

Can we do better?

Another Approach

Double the array size whenever we increase capacity

Why does this help? How much does it help?

Modified Code

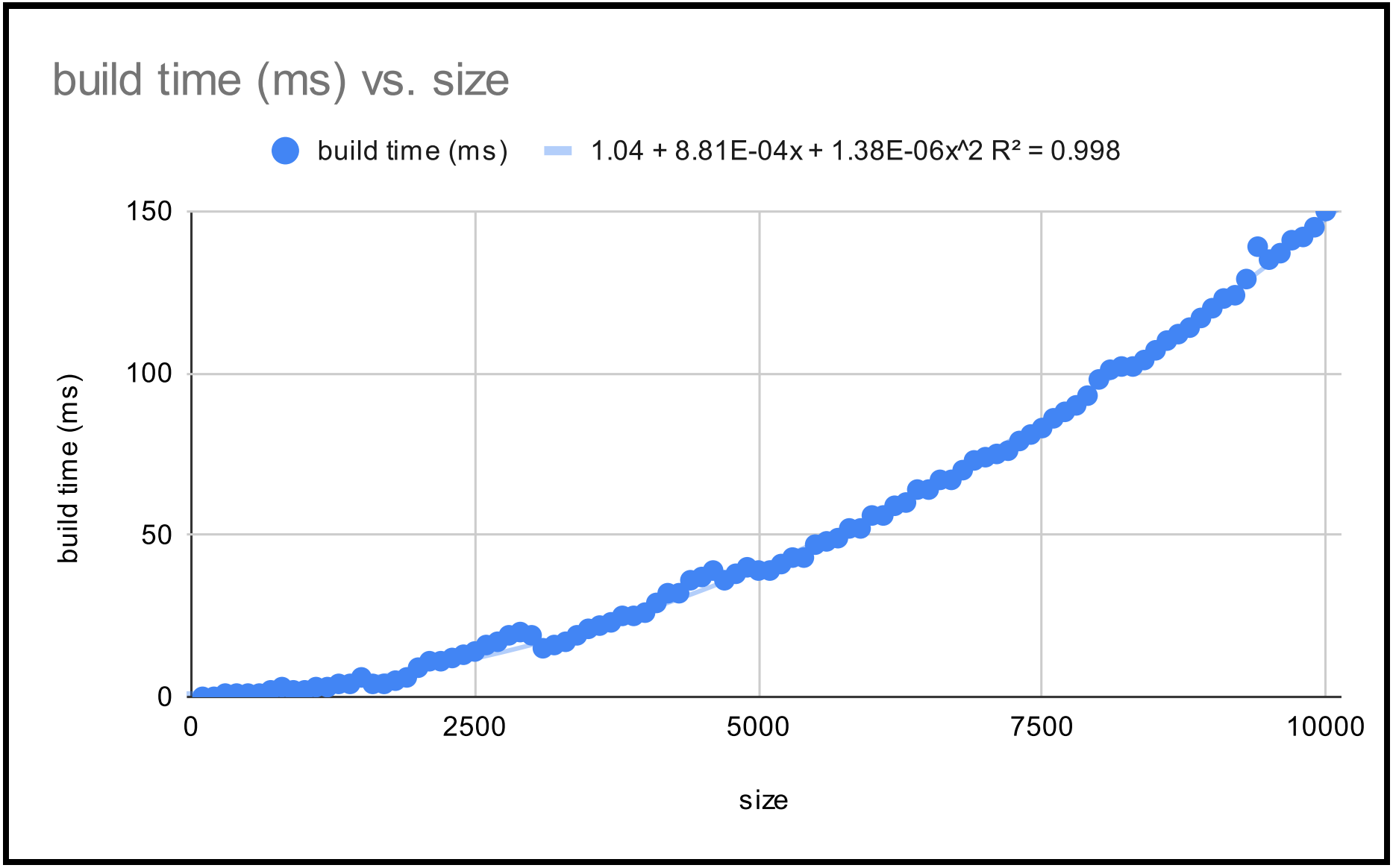
Before:

```
private void increaseCapacity() {
    Object[] bigContents = new Object[capacity + 1];
    for (int i = 0; i < capacity; ++i) {
        bigContents[i] = contents[i];
    }
    contents = bigContents;
    capacity = capacity + 1;
}
```

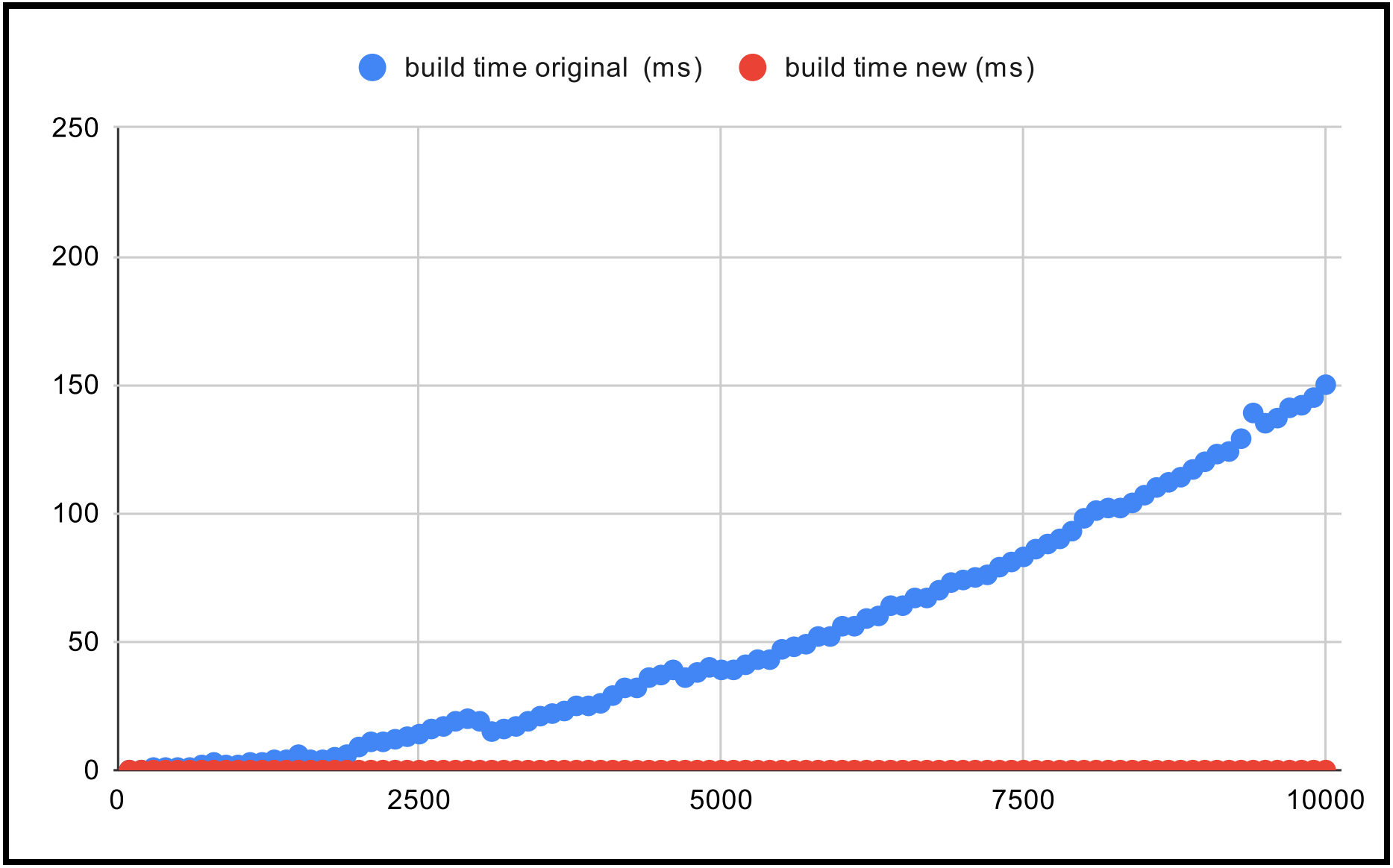
After:

```
private void increaseCapacity() {
    Object[] bigContents = new Object[2 * capacity];
    for (int i = 0; i < capacity; ++i) {
        bigContents[i] = contents[i];
    }
    contents = bigContents;
    capacity = 2 * capacity;
}
```

Performance Before



Performance After



Performance After Again

Astonishment

Second implementation built a list of size 5 million faster than first implementation built a list of size 10 thousand!

Question

Why is performance *so much* better?