CSE 250: Induction

Lecture 12

Sept 25, 2023

Reminders

- WA2 due Sun, Oct 1 at 11:59 PM
  - Released today.
- Midterm 1 on Oct 2 in-class
  - Covers: Asymptotics, Sequences/Lists, Arrays, Linked Lists, Recursion
  - Bounds: Tight Upper/Lower, Unqualified vs Amortized

Sorted Lists

Finding an item

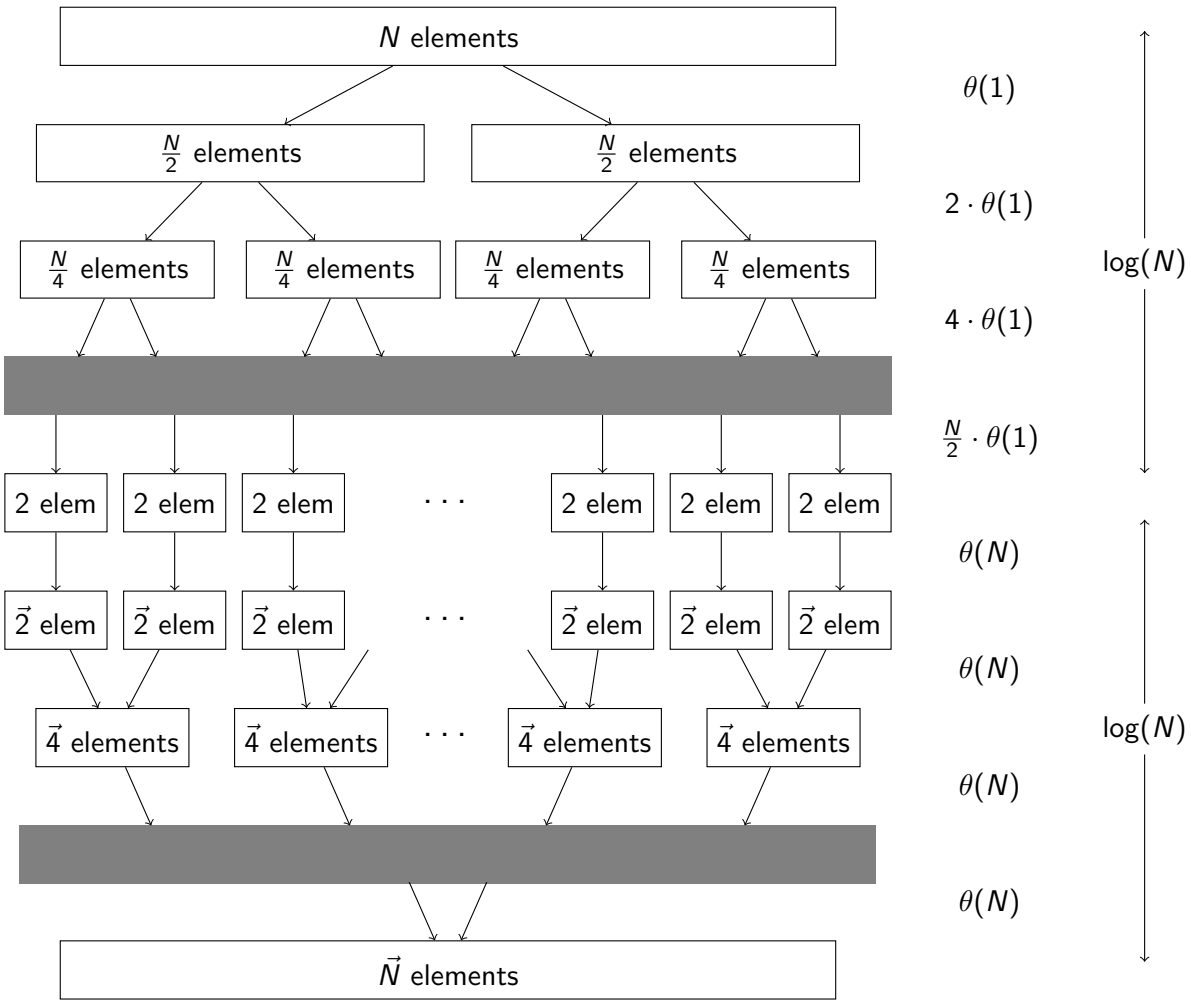
- **Regular List**: Sequential Scan $O(N)$
- **Sorted List**: Binary Search $\theta(log(N))$ calls to `get()`

So how do we get a sorted list?

Bubble Sort

$O(N^2)$

Merge Sort



Merge Sort

$$\left(\sum_{i=0}^{\log(N)-1} 2^i \cdot \theta(1)\right) + \left(\sum_{i=1}^{\log(N)-1} \theta(N)\right)$$

$$\left(2^{\log(N)}\theta(1)\right) + (\log(N)\theta(N))$$

$$\theta(N) + \theta(N\log(N))$$

Merge Sort: $\theta(N\log(N))$

Bubble Sort: $\theta(N^2)$

Induction

Can we frame this proof in terms of the code?

Merge Sort

```
1   public ArrayList<E> mergeSort<E>(ArrayList<E> list)
2   {
3     if(list.size() > 2){
4       int splitIndex = input.size()/2;
5       ArrayList<E> left =
6         mergeSort(list.subList(0, splitIndex));
7       ArrayList<E> right =
8         mergeSort(list.subList(splitIndex, list.size));
9       return merge(left, right);
10    } else {
11      if((list.size() == 2) && (list.get(0) > list.get(1))){
12        E tmp = list.get(0);
13        list.set(0, list.get(1));
14        list.set(1, tmp);
15      }
16      return list;
17    }
18  }
```

Merge Sort

- If $N > 2$
  - Split $O(1)$
  - $2\times$ `mergeSort(`$\frac{N}{2}$`)` ???
  - `merge(`N`)` $O(N)$
- Otherwise
  - Sort 2 elements $O(1)$

$$T_{merge}(N) = \begin{cases} O(N) + 2 \cdot ??? T_{merge}(\frac{N}{2}) & \textbf{if } N > 2 \\ O(1) & \textbf{otherwise} \end{cases}$$

How do we solve for $T_{merge}(N)$?

Recursive Complexity

Let's start with a simpler problem.

Factorial

$$439! = 439 \cdot 438 \cdot 437 \cdot 436 \cdot 435 \cdot 434 \cdot 433 \cdot 432 \cdot \ldots$$
$$= 439 \cdot 438!$$

$$438! = 438 \cdot 437 \cdot 436 \cdot 435 \cdot 434 \cdot 433 \cdot 432 \cdot \ldots$$

Factorial

$$N! = N \cdot (N-1)!$$

$$\underbrace{N!}_{\text{big problem}} = N \cdot \underbrace{(N-1)!}_{\text{smaller (same) problem}}$$

```
1    public long factorial(long N)
2    {
3      return N * factorial(N-1);
4    }
```

StackOverflowError

Factorial

- $1! = 1$ — Base Case
- $N! = N \cdot (N-1)!$ — Recursive Case

```
1  public long factorial(long N)
2  {
3    if(N <= 1){ return 1; }
4    else { return N * factorial(N-1); }
5  }
```

Factorial

What is the complexity of Factorial?

```
1    public long factorial(long N)
2    {
3      if(N <= 1){ return 1; }
4      else { return N * factorial(N-1); }
5    }
```

$$T_{factorial}(N) = \begin{cases} \theta(1) & \textbf{if } N \leq 1 \\ \theta(1) + ??? T_{factorial}(N-1) & \textbf{otherwise} \end{cases}$$

Factorial

$$T_{factorial}(N) = \begin{cases} \theta(1) & \textbf{if } N \leq 1 \\ \theta(1) + T_{factorial}(N-1) & \textbf{otherwise} \end{cases}$$

Solve for $T_{factorial}(N)$.

Induction

Solve for $T_{factorial}(N)$.

Induction

1. Generate a hypothesis.
2. Prove the hypothesis for the base case.
3. Prove the hypothesis inductively.

Generate a Hypothesis

Hypothesis: Solve for increasing values of N

N	1	2	3	4	5	6
$T(N)$	$1 \cdot \theta(1)$	$2 \cdot \theta(1)$	$3 \cdot \theta(1)$	$4 \cdot \theta(1)$	$5 \cdot \theta(1)$	$6 \cdot \theta(1)$

What's the pattern? $(N \cdot \theta(1))$

Hypothesis: $T(N) \in \theta(N)$

- There is some $c_{high} > 0$ such that $T(n) \leq c_{high} \cdot n$ ←
- There is some $c_{low} > 0$ such that $T(n) \geq c_{low} \cdot n$

Algebra with θ

Remember, $\theta(N)$ used in a math equation is shorthand for: $f(N)$ where $f(N) \in \theta(N)$

So $\theta(1)$ is shorthand for some constant c.

Factorial

$$T_{factorial}(N) = \begin{cases} \theta(1)c_1 & \textbf{if } N \leq 1 \\ \theta(1)c_2 + T_{factorial}(N-1) & \textbf{otherwise} \end{cases}$$

Goal: Show that $T_{factorial}(N) \leq c \cdot N$ for some $c > 0$ $(N > N_0)$

Factorial

$$T_{factorial}(N) = \begin{cases} c_1 & \textbf{if } N \leq 1 \\ c_2 + T_{factorial}(N-1) & \textbf{otherwise} \end{cases}$$

Goal: Show that $T_{factorial}(N) \leq c \cdot N$ for some $c > 0$ $(N > N_0)$

$$T_{factorial}(1) \overset{?}{\leq} 1 \cdot c$$

$$c_1 \overset{?}{\leq} 1 \cdot c \checkmark$$

Factorial

$$T_{factorial}(N) = \begin{cases} c_1 & \textbf{if } N \leq 1 \\ c_2 + T_{factorial}(N-1) & \textbf{otherwise} \end{cases}$$

Goal: Show that $T_{factorial}(N) \leq c \cdot N$ for some $c > 0$ $(N > N_0)$

$$T_{factorial}(2) \overset{?}{\leq} 2 \cdot c$$

$$c_2 + T_{factorial}(1) \overset{?}{\leq} 2 \cdot c$$

$$c_2 + c_1 \overset{?}{\leq} 2 \cdot c \checkmark$$

Factorial

$$T_{factorial}(N) = \begin{cases} c_1 & \textbf{if } N \leq 1 \\ c_2 + T_{factorial}(N-1) & \textbf{otherwise} \end{cases}$$

Goal: Show that $T_{factorial}(N) \leq c \cdot N$ for some $c > 0$ $(N > N_0)$

$$\begin{aligned} T_{factorial}(2) &\overset{?}{\leq} 2 \cdot c \\ c_2 + T_{factorial}(1) &\overset{?}{\leq} 2 \cdot c \\ c_2 + T_{factorial}(1) &\overset{?}{\leq} c + c \\ c_2 &\overset{?}{\leq} c \checkmark \end{aligned}$$

Factorial

$$T_{factorial}(N) = \begin{cases} c_1 & \textbf{if } N \leq 1 \\ c_2 + T_{factorial}(N-1) & \textbf{otherwise} \end{cases}$$

Goal: Show that $T_{factorial}(N) \leq c \cdot N$ for some $c > 0$ $(N > N_0)$

$$\begin{aligned} T_{factorial}(3) &\overset{?}{\leq} 3 \cdot c \\ c_2 + T_{factorial}(2) &\overset{?}{\leq} 3 \cdot c \\ c_2 + T_{factorial}(2) &\overset{?}{\leq} c + 2c \\ c_2 &\overset{?}{\leq} c \checkmark \end{aligned}$$

Factorial

This is boring!

Can't I automate the proof?

Induction

1. Generate a hypothesis. $T(N) \in O(N)$
2. Prove the hypothesis for the base case. $\exists c : T(N) \leq c \cdot N$
3. Prove the hypothesis inductively. ...
   - Assume you've proved it for case $N-1$
   - Prove that it holds for case N

Factorial

$$T_{factorial}(N) = \begin{cases} c_1 & \textbf{if } N \leq 1 \\ c_2 + T_{factorial}(N-1) & \textbf{otherwise} \end{cases}$$

Goal: Show that $T_{factorial}(N) \leq c \cdot N$ for some $c > 0$ $(N > N_0)$

Assume: $T_{factorial}(N-1) \leq c \cdot N - 1$

$$T_{factorial}(N) \stackrel{?}{\leq} N \cdot c$$

$$c_2 + T_{factorial}(N-1) \stackrel{?}{\leq} N \cdot c$$

$$c_2 + T_{factorial}(N-1) \stackrel{?}{\leq} c + (N-1)c$$

$$c_2 \stackrel{?}{\leq} c \checkmark$$

Induction

We showed there exists a c such that...

- $T(1) \leq 1 \cdot c$ (Base Case)
- if $T(N-1) \leq (N-1) \cdot c$ then $T(N) \leq N \cdot c$ (Inductive Proof)

So...

1. $T(1) \leq 1 \cdot c$ (Base Case)
2. $T(2) \leq 2 \cdot c$ (#1 + Inductive Proof)
3. $T(3) \leq 3 \cdot c$ (#2 + Inductive Proof)
4. $T(4) \leq 4 \cdot c$ (#3 + Inductive Proof)
5. $T(5) \leq 5 \cdot c$ (#4 + Inductive Proof)
6. $T(6) \leq 6 \cdot c$ (#5 + Inductive Proof)
7. ...

The proof holds for any $N \geq 1 \rightarrow T(N) \in O(N)$ ✓

Factorial

What is the complexity of Factorial?

```
1    public long factorial(long N)
2    {
3      if(N <= 1){ return 1; }
4      else { return N * factorial(N-1); }
5    }
```

Answer: $O(N)$[1]

How much memory does it use?

[1] Technically it's $\theta(N)$, but we haven't proven $T(N) \in \Omega(N)$