CSE 326 Quiz Section:
Analyzing Analysis
April 18, 2002

“Plain” Algorithmic Analysis
• Algorithm $Foo$ is $O(f(n))$
  – For all inputs, $Foo$ takes at most $cf(n)$ steps.
  – Upper-bound is $f(n)$
• Algorithm $Foo$ is $\Omega(f(n))$
  – For all inputs, $Foo$ takes at least $cf(n)$ steps.
  – Lower-bound is $f(n)$
• Algorithm $Foo$ is $\Theta(f(n))$
  – For all inputs, $Foo$ takes approximately $cf(n)$ steps.
  – Lower-bound and upper-bound are both $f(n)$

Visual Representation: Plain

Worst, Best, Average… Oh My!

WORST $O(f(n))$

AVERAGE $\Theta(f(n))$

BEST $\Omega(f(n))$

Worst-case Analysis
• Idea: What is the most work algorithm $Foo$ will ever have to do?
• What the bounds mean for worst-case analysis:
  – $O(f(n))$: For any input, $Foo$ takes at most $cf(n)$ time
  – $\Omega(f(n))$: There exists an input of length $n$ such that $Foo$ takes at least $cf(n)$ time
  – $\Theta(f(n))$: There exists an input of length $n$ such that $Foo$ takes at least $cf(n)$ time and no other input of length $n$ takes more than $df(n)$ time.
Visual Representation: Worst

Best-case Analysis
- Idea: What is the least work algorithm $Foo$ will ever have to do?
- What the bounds mean for worst-case analysis:
  - $O(f(n))$: There exists an input of length $n$ such that $Foo$ takes at most $c f(n)$ time
  - $\Omega(f(n))$: For any input, $Foo$ takes at least $cf(n)$ time
  - $\Theta(f(n))$: There exists an input of length $n$ such that $Foo$ takes at most $cf(n)$ time and no other input of length $n$ takes less than $df(n)$ time.

Visual Representation: Best

Average-case Analysis
- The book calls this “expected analysis”
- IDEA: On average, how much work will $Foo$ do?
- The method: for a fixed input size $n$
  compute $T(n)$ for all inputs
  take the average of all these $T(n)$
- $O(f(n))$ and $\Omega(f(n))$ act just like mathematical upper and lower bounds.

Real-World Expected Analysis
- In a purely mathematical sense, which is more likely of an input to sort?
  1,2,4,6,8,7,11,10,12,13
  or
  3,7,5,1,9,12,8,6,4,10
- What about more likely in the real world?

Real-World Expected Analysis
IDEA: Reflect the actual probability of inputs in the cost analysis
1. Fix input size $n$
2. For each input $i$ of size $n$, assign a probability of it occurring
3. Compute
   $$\sum_{\text{input } i} P(i) \cdot (\text{Time cost of } i)$$
Expected Analysis

- Expected analysis usually refers to analyzing the performance of a randomized algorithm
- Randomized algorithms involve random choices in their operations, meaning the amount of time spent for one input can vary from run to run
- Idea for the analysis:
  average time for a randomized algorithm over different random seeds for any input

Now into some reality

<table>
<thead>
<tr>
<th></th>
<th>Sorted Linked List</th>
<th>Unsorted Array</th>
<th>Sorted Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Insert</td>
<td>$O(n)$</td>
<td>$O(1)$ or $O(n)$?</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Delete w/</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Find</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Delete w/o</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Find</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

Stretchy Array Implementation

```c
int * data;  // Best case insert = O(1)
int maxsize, end;  // Worst case insert = O(n)

insert(e) {
  if (end == maxsize) {
    temp = new int[2*maxsize];
    for (i=0; i<maxsize; i++)
      temp[i] = data[i];
    delete data;
    data = temp;
    maxsize = 2*maxsize;
  }
  data[++end] = e;
}
```

Inserting in an Unsorted Array

- Inserting is usually $O(1)$ time
- Stretching the array takes $O(n)$ time
- Does inserting always take linear time?

Amortized Analysis

- Consider any sequence of operations applied to a data structure
  – your worst enemy could choose the sequence!
- Some operations may be fast, others slow
- Goal: show that the average time per operation is still good

```
total time for n operations = \frac{\text{total time for n operations}}{n}
```

Stretchy Array Amortized Analysis

- Consider sequence of $n$ operations
  `insert(3); insert(19); insert(2); …`
- What is the max number of stretches?
- What is the total time?
  – let’s say a regular insert takes time $a$, and stretching an array contain $k$ elements takes time $bk$.

- Amortized time =
Stretchy Array Amortized Analysis

• Consider sequence of n operations
  insert(3); insert(19); insert(2); …
• What is the max number of stretches? \log n
• What is the total time?
  – let’s say a regular insert takes time a, and stretching an
    array contain k elements takes time bk.
    \[ an + b(1 + 2 + 4 + 8 + \ldots + n) = an + b \sum_{i=0}^{\log n} 2^i \]
• Amortized time =

\[
\begin{align*}
\text{Stretchy Array Amortized Analysis} \\
\text{• Consider sequence of n operations} \\
\text{insert(3); insert(19); insert(2); …} \\
\text{• What is the max number of stretches? } \log n \\
\text{• What is the total time?} \\
\text{– let’s say a regular insert takes time a, and stretching an} \\
\text{array contain k elements takes time bk.} \\
\text{\[ an + b(1 + 2 + 4 + 8 + \ldots + n) = an + b \sum_{i=0}^{\log n} 2^i \]} \\
\text{\[ = an + b(2n - 1) \]} \\
\text{• Amortized time = } (an + b(2n - 1))/n = O(1) 
\end{align*}
\]