Logarithms

- $f(n) = \log_{10} n$ and $g(n) = \log_2 n$. How do they relate?
  \[
  \log_{10} n = c \cdot \log_2 n
  \]
  \[
  \log_{10} n \text{ is } \Theta (\log_2 n)
  \]
- $f(n) = \log n$. What base do I mean?
Ω and Θ

- $\mathcal{O}$: upper bound on growth rate of function.
- What are Ω and Θ used to describe?

---

A reminder

There are two steps for analysis:

1. What am I analyzing?
   (worst-case? best-case? etc)

2. What bound am I providing?
   (upper, lower, etc)
Running Time of DFS? \( n = |V| \quad m = |E| \)

**DFS-iterative\( (s) \)**
- \( \forall v \) discovered\( [v] = \text{false} \) \( \{ O(n) \} \)
- \( S \leftarrow \text{empty stack} \) \( \{ O(1) \} \)
- push \( s \) to \( S \) \( \{ O(1) \} \)
- \text{while } \text{S is not empty do} \)
  - \( u \leftarrow \text{pop}(S) \)
  - if discovered\( [u] = \text{false} \) then \( \{ O(1) \} \)
    - if discovered\( [u] = \text{true} \) \( \{ O(m) \} \)
    - for all edges \( (u, v) \) do \( \{ O(\delta(u)) \} \)
    - push\( (S, v) \) \( \{ O(1) \} \)
- \( O(n + m) \)

**Totals**
- \( O(n) \)
- \( O(m) \)
- \( O(n) \)
- \( O(m) \)
- \( O(m + n) \)

A new procedure for Stacks/Queues

When push to a “full” array-based stack or queue:
- Create new Object *, twice size of current
- Copy everything from elements to new array.
- Delete the old elements
- Set the array pointer to point to this new one.

- Is this \( O(n) \)? \( \Theta(n) \)?

\[ \text{NO} \]
Step One: Gather Data

- Some operations cost more than others
- Many cheap operations, then an expensive
- What about a sequence of operations?

Goal at this step: develop a hypothesis.

**Question:** Total work after $k$ pushes?

<table>
<thead>
<tr>
<th>Push #</th>
<th>Work here</th>
<th>Total so far</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>1 each</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>21</td>
</tr>
<tr>
<td>12-20</td>
<td>1 each</td>
<td>30</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
<td>51</td>
</tr>
<tr>
<td>22-40</td>
<td>19</td>
<td>70</td>
</tr>
<tr>
<td>41</td>
<td>41</td>
<td>111</td>
</tr>
</tbody>
</table>

Total Work Done?

![Graph showing total work done over pushes]
Running time of new procedure

- Assume elements began at small size.
  - A power of two
- Only double in size when needed.
- A push takes $O(1)$ unless we are doubling.
- The doubling takes $O(n)$, but happens rarely.
- I want to show: $k$ pushes takes $O(k)$

$$
\sum_{i=0}^{k} \Theta(i) + \sum_{i=0}^{k} \Theta(1) \quad \text{then simplify}
$$

- **Amortizes**

Running time of new procedure

- Assume elements began at small size.
- Only double in size when needed.
- A push takes $O(1)$ unless we are doubling.
- The doubling takes $O(n)$, but happens rarely.
- I want to show: $k$ pushes takes $O(k)$

There’s a better way: “credit argument”

Charge each push $\$1+\$2=\$3$

If $O(i)$ push, I pay $\$1$ and save rest
If $O(n)$ push, need $\$n$ in savings

How many pushes since last linear push?

$$\frac{n}{2} \quad \text{Charge each } \$2\text{ extra}$$
Another View

Warm Up for Wednesday

- Linked list is sorted
- Linked list has $-\infty$ and $\infty$
- Want largest key whose value is at most $k$

find($k$)