I & C SCI 46 Winter 2023
Lecture 6: Analysis II

2

Logarithms

- $f(n) = \log_{10} n$ and $g(n) = \log_2 n$.
  How do they relate?

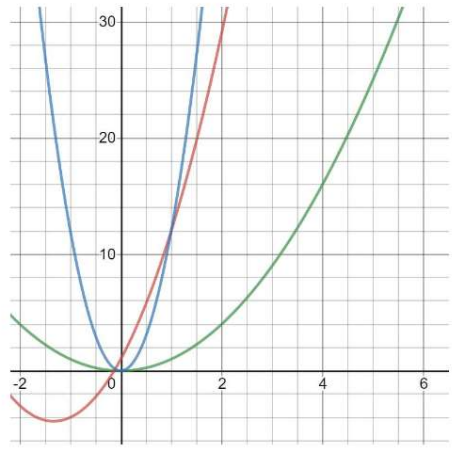
  $\log_{10} n = c \cdot \log_2 n$

  $\log_{10} n$ is $O(\log_2 n)$
- $f(n) = \log n$. What base do I mean?

3

Ω and Θ

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



4

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)

5

Running Time of DFS?

$n = |V|$ $m = |E|$

Totals

```
DFS-iterative(s)
  ∀v discovered[v] = false
  S ← empty stack
  push s to S
  while S is not empty do
    u ← pop(S)
    if discovered[u] = false then
      discovered[u] = true
      for all edges (u, v) do
        push(S, v)
```

$O(n)$

$O(1)$

$O(1)$ each time $\rightarrow$ $O(n)$ total

TOTAL of $O(n)$

$O(\delta(u))$ each time

$O(\sum \delta(u))$ total

$= O(m)$

$O(n)$

$O(m)$

$O(n)$

$O(m)$

$O(m+n)$

6

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 $O(n)$? ✓ $\Theta(n)$? NO

7

Step One: Gather Data

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

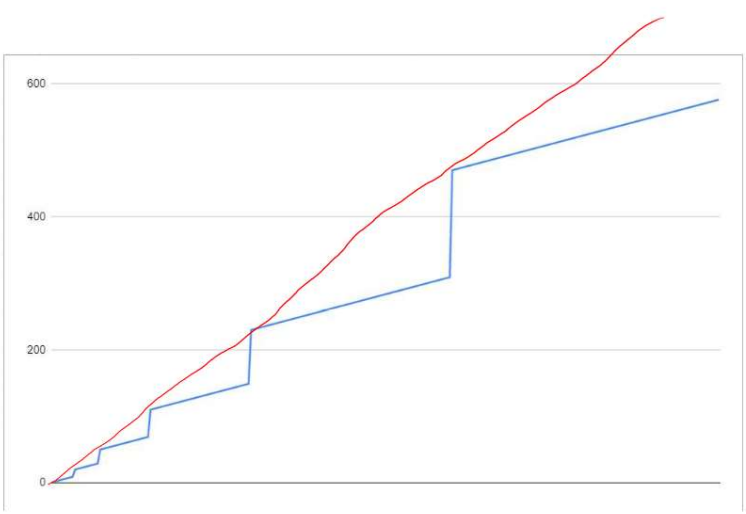
Goal at this step: develop a hypothesis.

Question: Total work after k pushes?

push #	work here	total so far
1-10	1 each	10
11	11	21
12-20	1 each	30
21	21	51
22-40	19	70
41	41	111

8

Total Work Done?



9

Running time of new procedure

amortizes

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

$$\sum_{\substack{i=0; \\ i \text{ pows of } 2}}^{k} \Theta(i) + \sum_{\substack{i=0; \\ \text{not pows of } 2}}^{k} \Theta(1)$$

then simplify

9

Running time of new procedure

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

There's a better way: "credit argument"

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

If $O(1)$ push, I pay \$1 and save rest

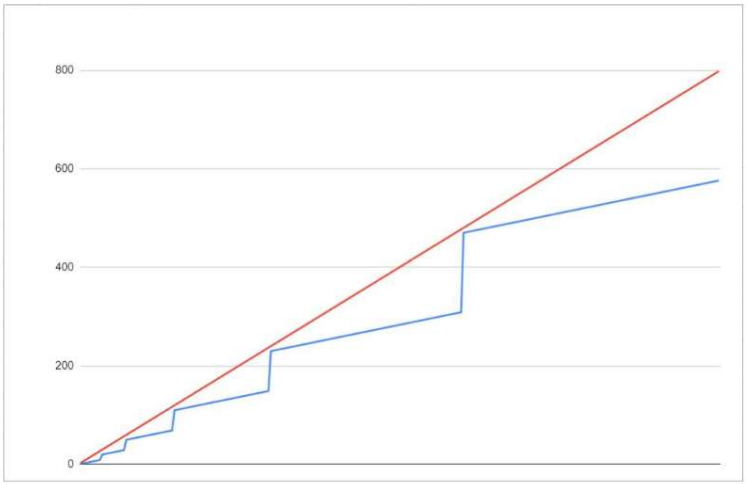
If $O(n)$ push, need \$n in savings

How many pushes since last linear push?

$\frac{n}{2}$. Charge each \$2 extra

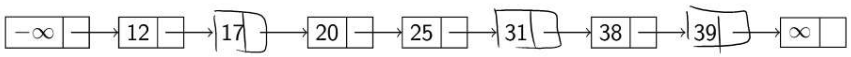
10

Another View



11

Warm Up for [redacted] Wednesday



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

`find(`k`)`