CS61B DISCUSSION 9

TA: SHERDIL NIYAZ

Administrivia

- Spring break is here! Have some fun (…do something besides CS.)
- Midterm 2 is the **Thursday after we get back.**
- Please make sure to start studying around now (this will give you a good two weeks to get ready). The exam will be of typical difficulty for 61B (no "nerfing" the exam because of Midterm 1)

How to study for MT2

- Common trend with students who didn’t do well on MT1: didn’t do many practice tests.
- HKN and TBP have resources. Make sure to check them out.
- My recommendation: you should have done 7-8 practice midterms by the time that Midterm 2 rolls around.
- Don’t just bash through them though. Do a practice test, see what you missed, study that (re-do discussion, rematch lecture, etc…). Rinse, lather, repeat.
- As always, feel free to email me if you’re feeling really stressed about 61B :) We’re here for you!

Old data structures

- Trees, lists, etc.
- Great for certain uses, but had an issue: not all operations were constant time.
- Wouldn't it be great if we could design a data structure that (on average) had constant runtime for **all** operations?

Enter the HashMap!

- Idea: Have data come in <key, value> pairs. This is an implementation of the **Map** ADT you saw in an earlier discussion.

- Idea: You put in a key value pair:

  put(“foo”, 3)

  and can then recover the value for a particular key:

  get(“foo”) <—- should return 3!!

How does it work?

- Blackboard time.

Pretty Picture

- ERMAGERD TECHNOLOGY
- (By the way, this only shows the **keys** for the hashed item. Don’t let that confuse you!)
- https://www.cs.usfca.edu/~galles/visualization/OpenHash.html

How did that work?

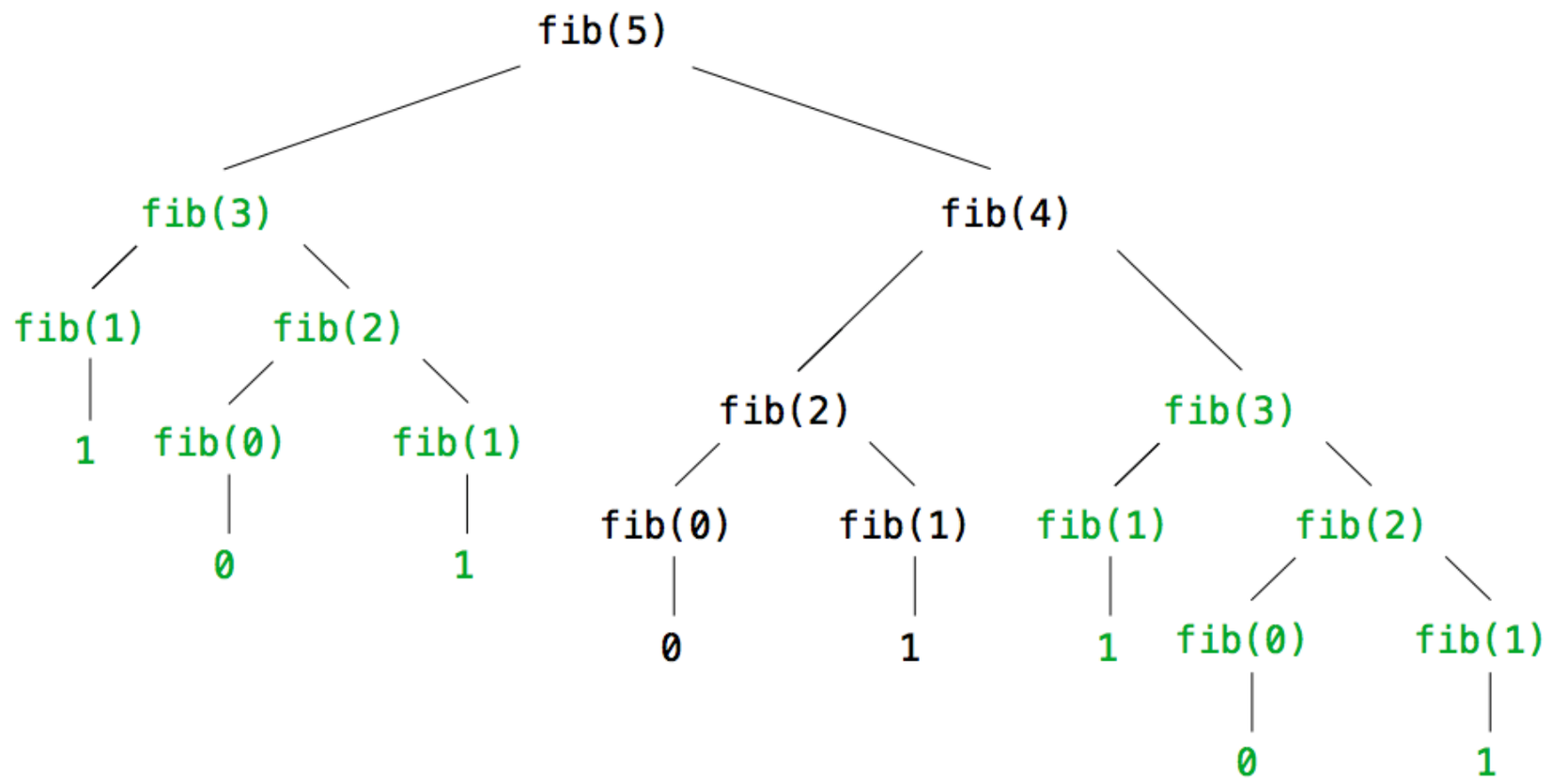
- Which has function looked better? (Strings or ints?)
- Remember: hash function should **look** random (it actually isn't, but it should appear that way for all intents and purposes).
- Bad hash function: reallllllly bad run time. If we have a pseudo-random function, the runtime of our hash map is O(1) on average.

QUESTIONS

61A Good Times

Repetition in Tree-Recursive Computation

This process is highly repetitive; fib is called on the same argument multiple times



(We will speed up this computation dramatically in a few weeks by remembering results)