CS 361
Data Structures & Algs
Lecture 8

Prof. Tom Hayes
University of New Mexico
09-16-2010

Today

More about Big O.

Online Hiring and Selling.

Sorting and Searching.

next: Priority Queues

Reminders

- Reading: finish Chapter 2 (sec 2.5 on PQs)
- Written Assignment 2: problems
  1.8, 2.1, 2.2, 2.3, 2.4, 2.5*, 2.6, 2.7, 2.8+
  *:tricky. +:challenging
  Quiz: 2 next Thursday
- Work together!

Test your understanding

True or False:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = C$$ implies f = O(g)

True. Existence of this limit implies that, for large n, f(n)/g(n) is arbitrarily close to C. In particular, f(n)/g(n) is between 0 and 2C. But this implies $f(n) \leq 2C\ g(n)$, so f = O(g).

Same proof shows f = Ω(g). Hence f=Θ(g)

Setting up a proof

Given: f = O(g). f, g are positive.

Prove: $f^2 = O(g^2)$

Proof:

Setting up a proof

Given: f = O(g). f, g are positive.

Prove: $f^2 = O(g^2)$

Proof: From the hypothesis, there exists C such that, for every n, $f(n) \leq C\ g(n)$.

...

Therefore, for every n, $f^2(n) \leq C'\ g^2(n)$, where C' = ... Thus $f^2 = O(g^2)$.

Setting up a proof

Given: f = O(g). f, g are positive.

Prove: $f^2 = O(g^2)$

Proof: From the hypothesis, there exists C such that, for every n, $f(n) \leq C\ g(n)$.

Square both sides. $f^2(n) \leq C^2\ g^2(n)$.

Therefore, for every n, $f^2(n) \leq C'\ g^2(n)$, where $C' = C^2$. Thus $f^2 = O(g^2)$.

Sorting Functions

Problems 2.1, 2.2, 2.3

Useful principles:

(1) $2^f = O(2^g)$ means "2^f does not exceed 2^g by more than a constant factor". Same as "f does not exceed g by more than an additive constant".

Why? 8 times 2^g equals 2^{g+3}
C times 2^g equals $2^{g + \log(C)}$

Sorting Functions

(2) Look at the ratio, f/g. f = O(g) means the ratio f/g is bounded. Try to simplify this ratio and understand it.

(3) How do changes to the input affect the output? Suppose f(n+1) = 2f(n)-1 and g(n+1) = g(n) + sqrt(g(n)). Then f will eventually grow faster than g.

(4) How fast does f double? Similar to (3). The function that doubles faster grows faster.

Online Algorithms

Sometimes, you have to make decisions before you know the whole input.

Examples:

Take local bus or wait for express?

Buy bread or wait for a sale?

Buy 1 loaf or stock up with 5?

Accept McDonalds job or keep hunting?

Price shareware at $10 or $20?

Online Algorithms

Sometimes, you have to make decisions before you know the whole input.

Reasons:

Information is not available, or

No time to read it. For example, a one-pass algorithm (linear time).

“Secretary problem”

100 applicants are coming to audition for secretary job at CIA. After talking to each, you must decide: hire, or shoot!

Goal: Hire the best of the 100. Once you hire, send the rest away with no interview.

Difficulty: You don’t know how good the ones you don’t interview are.

“Secretary problem”

100 applicants are coming to audition for secretary job at CIA. After talking to each, you must decide: hire, or shoot!

Goal: Hire the best of the 100. Once you hire, send the rest away with no interview.

Difficulty: You don’t know how good the ones you don’t interview are.

Solution: Interview, then shoot, the first 37. Then hire the next better one.

Selling widgets online

You have 100 widgets to sell. 1000 customers come by, and each offers you some money. Suppose the offers are independent random values between $1 and $100. Which offers should you accept?

Selling widgets online

You have 100 widgets to sell. 1000 customers come by, and each offers you some money. Suppose the offers are independent random values between $1 and $100. Which offers should you accept?

Accepting all offers above $90 is good, but not best possible. Why not?

Selling widgets online

You have 100 widgets to sell. 1000 customers come by, and each offers you some money. Now, suppose we don't know what the distribution of offers will be. What should we do?

Selling widgets online

You have 100 widgets to sell. 1000 customers come by, and each offers you some money. Now, suppose we don't know what the distribution of offers will be. What should we do?

"Statistical Inference": use the first few offers to infer a guess as to how the remaining offers will be distributed.

Selling widgets online

You have 100 widgets to sell. 1000 customers come by, and each offers you some money. Now, suppose we don't know what the distribution of offers will be. What should we do?

"Statistical Inference": use the first few offers to infer a guess as to how the remaining offers will be distributed.

Then try to sell to roughly the top 10% of remaining customers.

Combinatorial auction

In practice, one may have a more complicated setting, where customers offer money for “bundles” of several different kinds of widgets. For example, the active TopCoder contest on CuttingFigures.

Such settings may be very difficult to solve optimally, as they combine several different difficult aspects.

For instance, the FCC’s recent sale of radio station bandwidth.

Sorting and Searching

Sorting: Given an array of N numbers. Output the same array in increasing order. Running time?

Search: Maintain a set of N numbers, so that you can quickly find the i'th biggest one. How quickly can this be done?

Sorting and Searching

Sorting: Given an array of N numbers. Output the same array in increasing order. Running time?

Search: Maintain a set of N numbers, so that you can quickly find the i'th biggest one. How quickly can this be done?

Sorting

Sorting an Array can be done in time O(N log N). Several ways to do it:

1) MergeSort

2) QuickSort

3) with a Priority Queue (HeapSort)

4) other ways (e.g. CountingSort)