

CS 329e - Algorithms for Bioinformatics

Jan 24, 2008

Tandy Warnow

Topics for today

- Running time
- Big-oh analysis (and proving them)
- Recurrences (defining them, and solving them)

Analyzing the running time of FindMax

- **Index**:=1
- **Max**:=A[1]
- For i=2 up to n DO:
 - If A[i]>**Max**, then {set **Max**:=A[i] and set **Index**:=i}
- Return **Index**

Running time: $n+3$ operations for the initialization of variables (array A, Index, max, and i), $n-1$ comparisons of array entries to Max, at most $3(n-1)$ changes of variables, and then one operation for returning Index.

Total: At most $5n+1$

SimpleSort

- Input: array $A[1,2,\dots,n]$ of integers
- Output: the same elements but in sorted order, from smallest to largest

Algorithm

- If A has only one element, return A .
- Else let $i = \text{FindMax}[A]$, and swap $A[i]$ and $A[n]$.
- $\text{SimpleSort}(A[1\dots n-1])$

Running time analysis

- If $T(n)$ denotes the time used by SimpleSort on an array of n elements, then
- $T(1) = C_1$ (for some constant C_1)
- $T(n) \leq T(n-1) + C_2n + C_3$ (for some constants C_2 and C_3)

MergeSort

Same input as for SimpleSort (array of n integers), same output

Technique:

1. Divide array into two equal sized arrays.
2. Sort each of the two arrays, recursively.
3. Finally, put the two sorted arrays together (picking the smallest entry off the top of each array, in turn) to get the complete sort.

Running time of MergeSort

- $T(1) = C_1$
- $T(n) \leq 2T(n/2) + C_2n$ (let $n/2$ denote the ceiling of $n/2$)

Algorithmic running times

- Some algorithms are described as iterations -- these are generally easy to analyze (e.g., dynamic programming, bubble sort, insertion sort, find the max, etc.)
- Algorithms that are described recursively (e.g., divide-and-conquer) have running times $T(n)$ that are described using recurrence relations:
 - $T(n) = 3T(n/2) + n, T(1)=1.$
 - $T(n) = 2T(n/2) + 2n, T(1)=1$

Big-oh notation

- A function $f(n)$ is $O(g(n))$ if there is some pair of constants c and c' such that

$$f(n) \leq cg(n) \text{ whenever } n > c'$$

Equivalently, if $\lim_{n \rightarrow \infty} f(n)/g(n)$ exists, then

$$\lim_{n \rightarrow \infty} f(n)/g(n) \leq c$$

for some constant c

Proof techniques

- To see if $f(n)$ is $O(g(n))$ compute
$$\lim_{n \rightarrow \infty} f(n)/g(n)$$
- If this limit exists, then it must be that
$$\lim_{n \rightarrow \infty} f(n)/g(n) \leq C \text{ (for some constant } C)$$
- Examples:
 - $3n^2+5$ is $O(n^2)$
 - $87n^2+5000$ is $O(n^2)$
 - $5n^2$ is $O(n^3)$
 - But n^3 is *not* $O(n^2)$

General stuff about big-oh

Big-oh is an upper bound, not an exact statement

If $f(n)$ and $g(n)$ are polynomials and

- $\text{degree}(f) = \text{degree}(g)$ then $f(n)$ is $O(g(n))$ and $g(n)$ is $O(f(n))$
- $\text{degree}(f) < \text{degree}(g)$ then $f(n)$ is $O(g(n))$ but $g(n)$ is **not** $O(f(n))$

- Examples:
 - $3n^2$ is $O(n^2)$
 - $87n^2+5000$ is $O(n^2)$
 - $5n^2$ is $O(n^3)$
 - But n^3 is *not* $O(n^2)$
 - n^3 is $O(5n^3)$
 - Anything that is $O(5n^3)$ is $O(2n^3)$ and $O(3n^3+100n^2)$, etc.
- The leading constants (and lower degree terms) do not matter!

Proof 1.

How do we prove $3n^2$ is $O(n^2)$?

Find constants c and c' so that $3n^2 \leq cn^2$
for all $n > c'$.

Let $c=3$ and $c'=0$.

Proof 2.

How do we prove $5+3n^2$ is $O(n^2)$?

Find constants c and c' so that
 $5+3n^2 \leq cn^2$ for all $n > c'$.

Letting $c = 3$ doesn't work! So pick a bigger value (say, $c = 4$) and solve for c' .

Proof 2, cont.

We want to find constant c' so that $5+3n^2 \leq 4n^2$ for all $n > c'$.

Solving for n , we find that c' must satisfy that $n^2 \geq 5$ whenever $n > c'$. So let $c'=3$.

Together: **$5+3n^2 \leq 4n^2$ for all $n > 3$ and therefore $5+3n^2$ is $O(n^2)$**

Proof 3.

- Prove that $3n^2$ is $O(n^3)$.
- We want $3n^2 \leq cn^3$ for all $n > c'$.
- Set $c = 3$ and $c'=0$.

Proof 4.

- Prove that $5+3n^2$ is $O(n^3)$.
- We want $5+3n^2 \leq cn^3$ for all $n > c'$.
- Set $c = 4$ and solve for c' .
- $5+3n^2 \leq 4n^3$ for all $n > c'$. Solving for n we get a cubic (not easy to solve).

Proof 4, cont.

- If $n > c'$ satisfies $5 + 3n^2 \leq 4n^3$, then $4n^3 - 3n^2 \geq 5$ for $n > c'$.
- Note, $n^3 \geq n^2$, so $4n^3 \geq 4n^2$. Hence, $4n^3 - 3n^2 \geq n^2$. Hence, we can just solve $n^2 \geq 5$.
- Obvious: all $n > 3$ satisfy $n^2 \geq 5$ (so set $c' = 3$).
- Thus, we have shown:
 $5 + 3n^2 \leq 4n^3$ for all $n > 3$, and so $5 + 3n^2$ is $O(n^3)$

Proof 5

- Proof (by contradiction) that n^3 is not $O(n^2)$:
- Suppose otherwise. Then there are constants c and c' so that $n^3 \leq cn^2$ for all $n > c'$.
- Solving we find **$n \leq c$ for all $n > c'$** . But this is not true! (Set $C^* = \max\{c, c'\} + 1$. Then $C^* > c'$, and also $C^* > c$! And so we have shown that the statement above is never true, for any constants c, c').

True/False Big-Oh statements

1. $87n^2+5000$ is $O(n^2)$
2. $5n^2$ is $O(n^3)$
3. n^3 is $O(n^2)$
4. n^3 is $O(5n^3)$
5. $\ln n$ is $O(n)$
6. $(\ln n)^6$ is $O(n)$
7. n^3 is $O(n!)$
8. n^3 is $O(2^n)$
9. $n!$ is $O(2^{2n})$
10. 2^n is $O(n^n)$

Techniques for evaluating Big-Oh statements

- Find C so that $\lim_{n \rightarrow \infty} f(n)/g(n) < C$. Then pick C' for this C . (You probably will need to use L'Hopital's Rule here!)
- Compare logarithms of $f(n)$ and $g(n)$.

Using logs

- Suppose $f(n)$ is $O(g(n))$.
Then there are constants C, C' s.t.
 $f(n) \leq Cg(n)$ for all $n > C'$.
- Hence $\lg(f(n)) \leq \lg(Cg(n)) = \lg(C) + \lg(g(n))$ for all $n > C'$.
- So, one way to test whether $f(n)$ is $O(g(n))$ is to compare $\lg(f(n))$ to $\lg(g(n))$.

Using logs

- Let $f(n)=5n^3$ and $g(n)=n^3$. Then $\lg(f(n))=\lg 5+ 3 \lg(n)$, and $\lg(g(n))=3 \lg n$.
- Note that $\lg(f(n)) > \lg(g(n))$.
- However, $\lg(f(n)) \leq \lg(g(n)) + C^*$ for some constant C^* . Hence, $f(n)$ is $O(g(n))$.

Using logarithms

- Use logarithms
 - Compare $\ln(f(n))$ to $\ln(g(n))$
 - If $\ln(f(n)) \leq \ln(g(n)) + A$ (for all large enough n , and for some constant A), then $f(n)$ is $O(g(n))$
- Example: $f(n) = n^4$ and $g(n) = n^3$
 - $\ln(f(n)) = 4 \ln(n)$, $\ln(g(n)) = 3 \ln(n)$, but $4 \ln(n) > 3 \ln(n) + A$ (for all constants A , once n is large enough), and so $f(n)$ is not $O(g(n))$.
 - But $g(n)$ is $O(f(n))$ by the same analysis!

- Example: $f(n)=n^3$, and $g(n)=2^n$.
 - $\ln(f(n))=3 \ln(n)$
 - $\ln(g(n))=n \ln(2)$
- Is $3 \ln(n) \leq n \ln(2) + C$, for all large enough n and some constant C ?
- To find out, we compute
 - $\lim_{n \rightarrow \infty} [3 \ln(n)] / [n \ln(2) + C]$.

This should be at most 1 for $f(n)$ to be $O(g(n))$

Using logs to determine if n^3 is $O(2^n)$, cont.

We evaluate

$$\lim_{n \rightarrow \infty} [3 \ln(n)] / [n \ln(2) + C]$$

We use L'Hôpital's Rule:

$$\lim_{n \rightarrow \infty} [3 \ln(n)] / [n \ln(2) + C] =$$

$$\lim_{n \rightarrow \infty} [3 \ln(n)]' / [n \ln(2) + C]' =$$

$$\lim_{n \rightarrow \infty} [3/n] / [\ln(2)] =$$

$$\lim_{n \rightarrow \infty} 3 / [n \ln(2)] = 0$$

Hence, $\ln(n^3) \leq \ln(2^n)$ for all large enough n , and n^3 is $O(2^n)$