

- $\log^2 n^5 = (\log n^5)^2 = 25(\log n)^2 = 25\log^2 n$
- $\log 5n = \log 5 + \log n$

E

• Can we do better?

_____ Interview Question from before _____

_____ Beyond Big-O _____

MaxSeq2 (int arr[], int n) • Both MaxSeq1 and MaxSeq2 have same best case and worst int max = 0;for (int i = 1;i<=n;i++)</pre> case behavior int sum = 0;• In a sense, we can say more about them than big-O time for (int j=i;j<=n;j++)</pre> • I.e. we can say more than that their run time is approx "≤" sum += arr[j]; some amount • Want a way of saying "assymptotically equal to" if (sum > max) • In general, want assymptotic analogues of \leq , \geq , =, etc. max = sum; //and store i and j if desired return max; 6 9 Formal Defn of Big-O Analysis of MaxSeq2 _____ • Let *f* be the number of operations this algorithm performs. The:

$$f = \sum_{i=1}^{n} \sum_{j=i}^{n} 1$$
 (1)

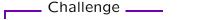
$$= \sum_{i=1}^{n} (n-i+1)$$
 (2)

$$= \sum_{i=1}^{n} i$$
 (3)
= $(n+1)(n/2)$ (4)
= $O(n^2)$ (5)

Before we go beyond big-O, what precisely does it *mean*?
It has a precise, mathematical definition:
A function f(x) is O(x(x)) if there exist positive constants

Example _____

 A function f(n) is O(g(n)) if there exist positive constants c and n₀ such that f(n) ≤ cg(n) for all n ≥ n₀



- MaxSeq2 is much better than MaxSeq1 $(O(n^2) \text{ vs } O(n^3))$
- But it's still not great, can you do better?
- We'll come back to this when we do recurrences

- Let's try to show that f(n) = 10n + 100 is O(g(n)) where g(n) = n
- We need to give constants c and n_0 such that $f(n) \leq cg(n)$ for all $n \geq n_0$
- In other words, we need constants c and n_0 such that $10n+100 \leq cn$ for all $n \geq n_0$

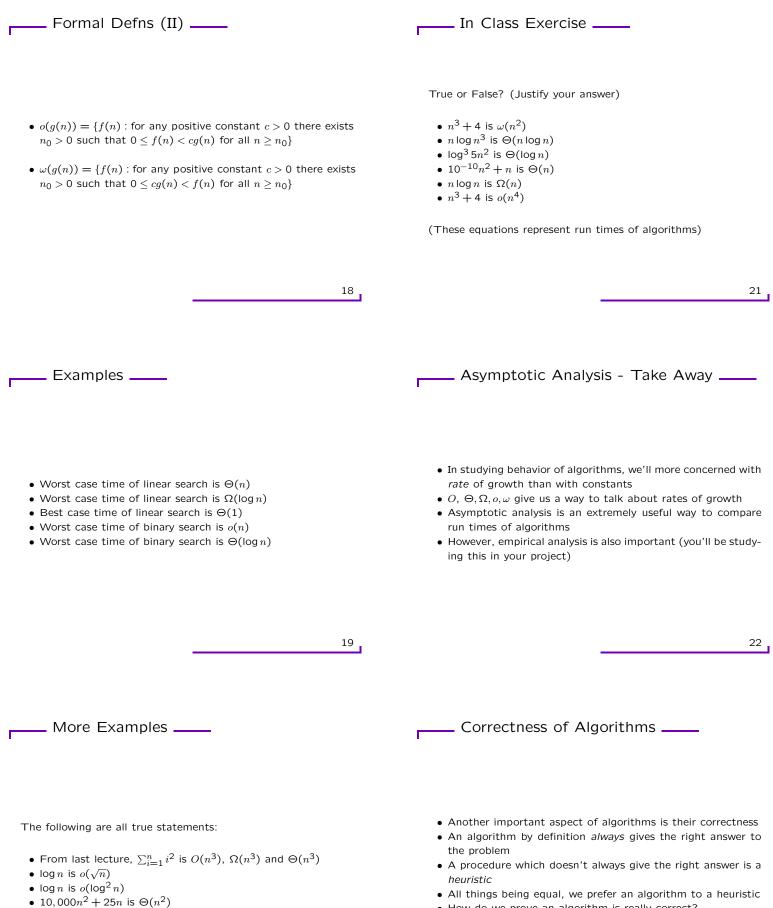
7

10

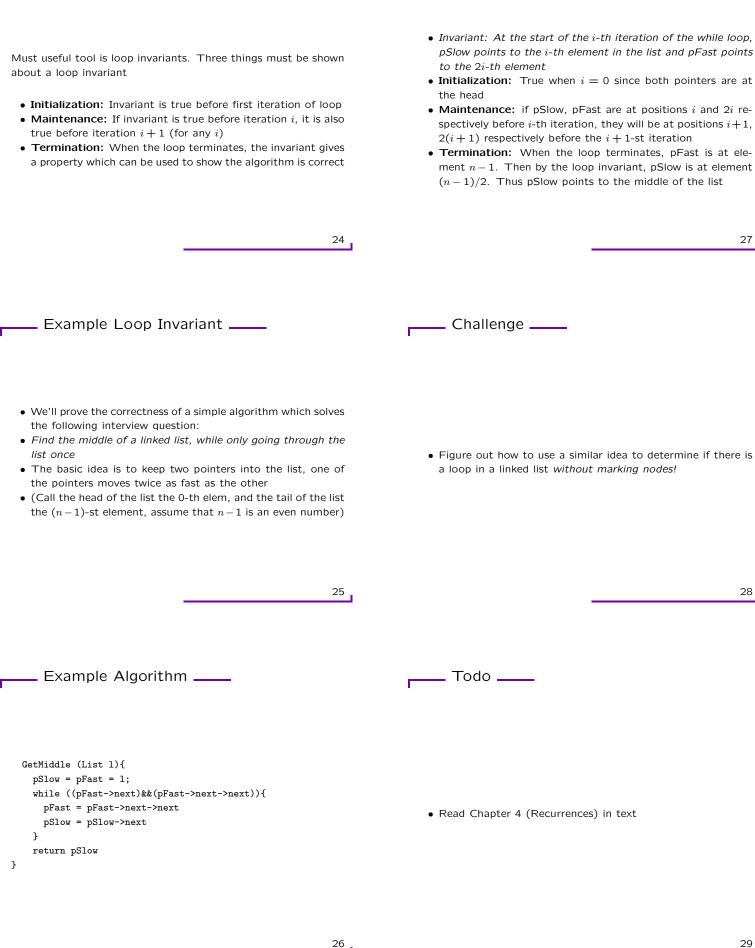
Following are some of the relatives of big-O: • We can solve for appropriate constants: 10n + 100 < cn(6) 10 + 100/n < c(7)Ο "<" Θ "=' • So if n > 1, then c need be greater than 110. Ω ">" • In other words, for all $n>1,\;10n+100\leq110n$ "<" 0 • So 10n + 100 is O(n)12 15 — Relatives of big-O —— Another Example _____ When would you use each of these? Examples: • Let's try to show that $f(n) = n^2 + 100n$ is O(g(n)) where $g(n) = n^2$ "<" This algorithm is $O(n^2)$ (i.e. worst case is $\Theta(n^2)$) Ο • We need to give constants c and n_0 such that $f(n) \leq cg(n)$ "=" Θ This algorithm is $\Theta(n)$ (best and worst case are $\Theta(n)$) for all $n \ge n_0$ Ω ">" Any algorithm for sorting is worst case $\Omega(n \log n)$ • In other words, we need constants c and n_0 such that $n^2 +$ "<" Can you write an algorithm for sorting that is $o(n^2)$? 0 $100n \le cn^2$ for all $n \ge n_0$ ">" This algorithm is not linear, it can take time $\omega(n)$ ω

- So if n > 1, then c need be greater than 101.
- In other words, for all $n>1,\;n^2+100n\leq 101n^2$

• $\Omega(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0$ such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0\}$



• How do we prove an algorithm is really correct?



Loop Invariants _____

_____ Example Loop Invariant _____