Questions from last time

CS 361, Lecture 2

Jared Saia University of New Mexico Express the following in O notation

- $n^3/1000 100n^2 100n + 3$
- $\log n + 100$
- $10 * \log^2 n + 100$
- $\sum_{i=1}^{n} i$

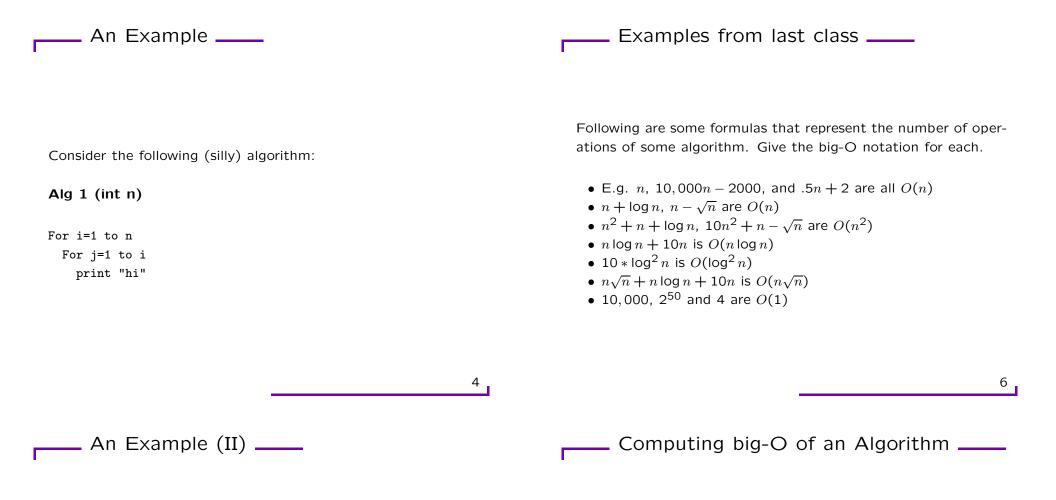
_ Today's Outline _____

• Asymptotic Analysis

Computing big-O of an Algorithm _____

- Write down a formula, f(n), which gives the number of elementary operations performed by the algorithm as a function of the input size, n
- Compute the big-O value for f(n)

2



- First we write down the formula f giving the number of basic operations the algorithm performs: $f = \sum_{i=1}^{n} i = (n+1)n/2$
- Next we compute the big-O value for f: (n+1)n/2 is $O(n^2)$

We can then say that Alg1 takes $O(n^2)$ time. Or, for short, we just say Alg1 is $O(n^2)$

Following is a shorter way to compute big-O for an algorithm:"Atomic operations"Constant timeConsecutive statementsSum of timesConditionalsLarger branch time plus test timeLoopsSum of iterationsFunction CallsTime of function bodyRecursive FunctionsSolve Recurrence Relation

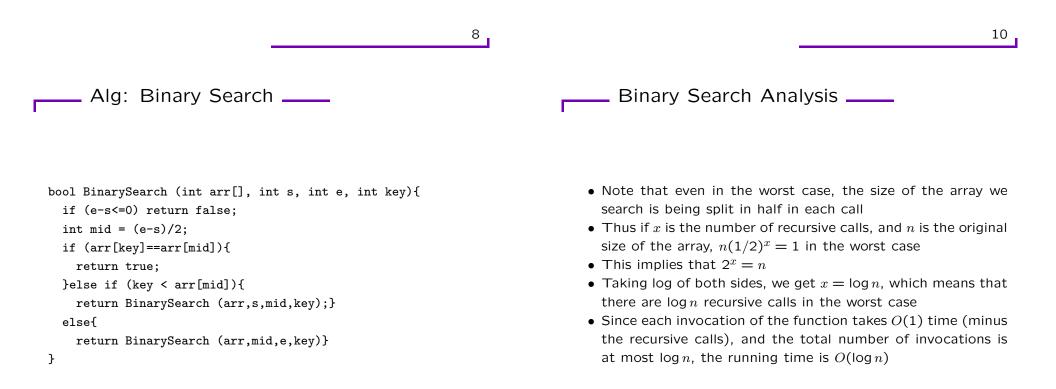
Alg: Linear Search

bool LinearSearch (int arr[], int n, int key){
 for (int i=0;i<n;i++){
 if (arr[i]==key)
 return true;
 }
}</pre>

```
return false;
```

}

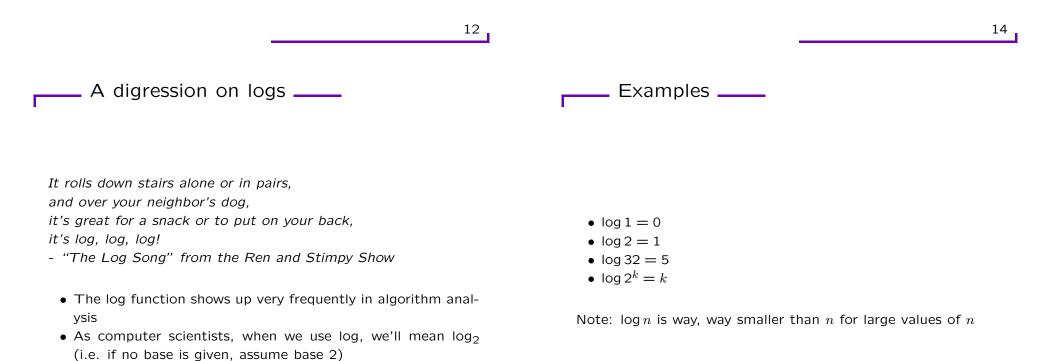
- To analyze the linear search algorithm, we consider the worst case
- The worst case occurs when the key is the very last element in the array
- In this case, the algorithm takes O(n) time
- Thus we say that the run time of Linear Search is O(n)
- (Note that the average time of Linear Search is also O(n))



____ Definition _____

- Linear Search is O(n) time
- Binary Search is $O(\log n)$ time
- Binary Search is a *much* faster algorithm, particularly for large input sizes

- $\bullet \, \log_x y$ is by definition the value z such that $x^z = y$
- $x^{\log_x y} = y$ by definition



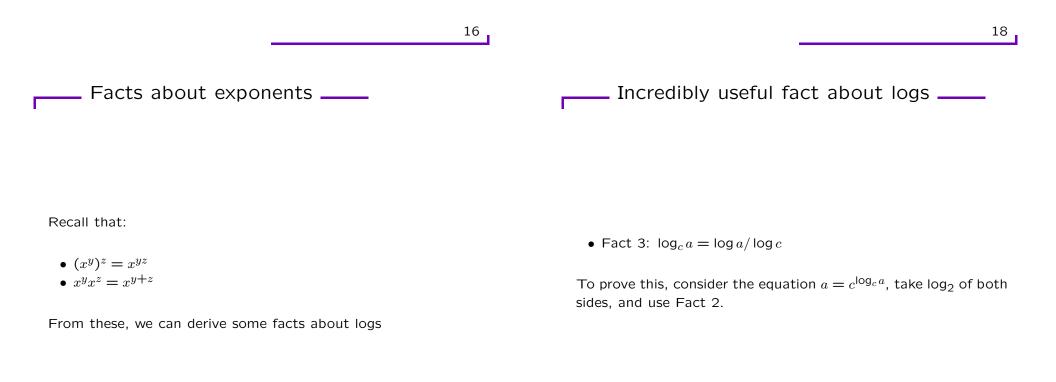
Examples _____

— Facts about logs ——

- $\log_3 9 = 2$
- $\log_5 125 = 3$
- $\log_4 16 = 2$
- $\log_{24} 24^{100} = 100$

To prove both equations, raise both sides to the power of 2, and use facts about exponents

- Fact 1: $\log(xy) = \log x + \log y$
- Fact 2: $\log a^c = c \log a$



Log facts to memorize _____

_ Take Away ____

Memorize these facts

- Fact 1: $\log(xy) = \log x + \log y$
- Fact 2: $\log a^c = c \log a$
- Fact 3: $\log_c a = \log a / \log c$

These facts are sufficient for all your logarithm needs. (You just need to figure out how to use them)

- All log functions of form $k_1 \log_{k_2} k_3 * n^{k_4}$ for constants k_1 , k_2 , k_3 and k_4 are $O(\log n)$
- For this reason, we don't really "care" about the base of the log function when we do asymptotic notation
- Thus, binary search, ternary search and k-ary search all take $O(\log n)$ time



- Note that $\log_8 n = \log n / \log 8$.
- Note that $\log_{600} n^{200} = 200 * \log n / \log 600$.
- Note that $\log_{100000} 30 * n^2 = 2 * \log n / \log 100000 + \log 30 / \log 100000$.
- Thus, $\log_8 n$, $\log_{600} n^{600}$, and $\log_{100000} 30*n^2$ are all $O(\log n)$
- In general, for any constants k_1 and k_2 , $\log_{k_1}n^{k_2}=k_2\log n/\log k_1$, which is just $O(\log n)$

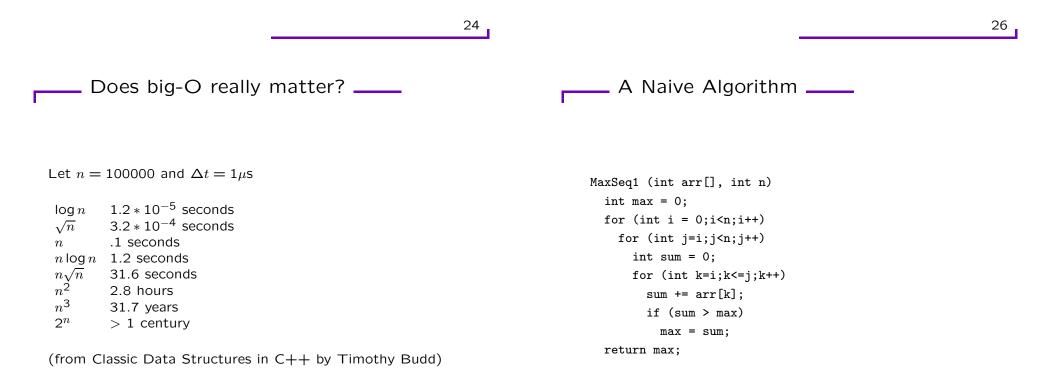
- $\log^2 n = (\log n)^2$
- $\log^2 n$ is $O(\log^2 n)$, not $O(\log n)$
- This is true since $\log^2 n$ grows asymptotically faster than $\log n$
- All log functions of form $k_1 \log_{k_3}^{k_2} k_4 * n^{k_5}$ for constants k_1 , k_2 , k_3, k_4 and k_5 are $O(\log^{k_2} n)$

In-Class Exercise

Simplify and give *O* notation for the following functions. In the big-O notation, write all logs base 2:

- $\log 10n^2$
- $\log_5(n/4)$
- $\log^2 n^4$
- 2^{log₄ n}
- $\log \log \sqrt{n}$

- The Question: Design an algorithm to return the largest sum of contiguous integers in an array of ints
- Example: if the input is (-10,2,3,-2,0,5,-15), the largest sum is 8, which we get from (2,3,-2,0,5).



Analysis _____

- Need to count the total number of operations of MaxSeq1
- Might as well assume time to do the inner loop is 1 (since it's a constant and therefore O(1))
- Let f(n) be the runtime for an array of size n

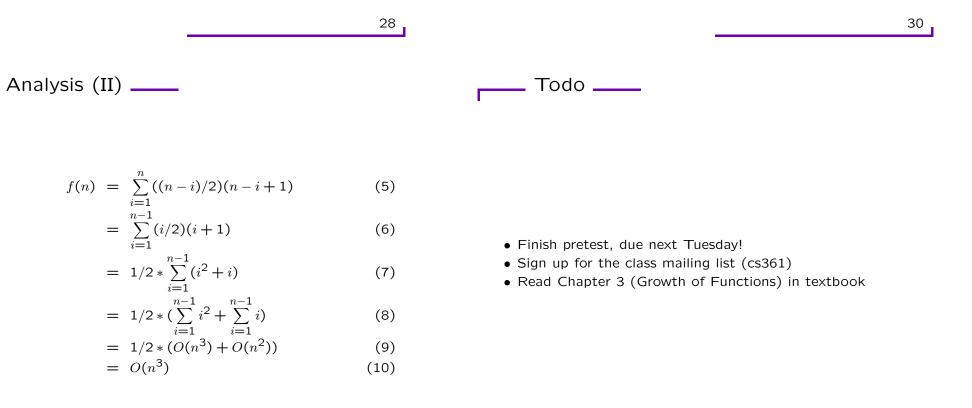
$$f(n) = \sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=i}^{j} 1$$
 (1)

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

$$= \sum_{i=1}^{n} \sum_{j=1}^{n-i} j$$
(3)

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

- MaxSeq1 is very slow
- This kind of algorithm won't impress an interviewer
- Can you do better?



31