

Discrete Mathematical Structures CS 2233 Lecture Fifteen

Prof. William Winsborough
March 17, 2009

Business

- **Recall:**
 - **Homework 7**, due Thursday 3/19
 - 3.2: 2, 4, 6, 8, 14, 20, 22
 - 3.3: 2
 - Read Sections 3.2 and 3.3
- IDEA Survey to be held
 - Who would be willing to volunteer to administer it?
- Questions???

17 March 2009

Winsborough CS 2233 Lecture 15

2

Big-Omega and Big-Theta

- Big-O provides an upper bound on function growth
- Big-Omega gives a lower bound
 - Def: $f(x)$ is $\Omega(g(x))$ if there are positive constants C and k such that $|f(x)| \geq C|g(x)|$ whenever $x > k$
- Big-Theta gives both
 - Def: $f(x)$ is $\Theta(g(x))$ if $f(x)$ is $O(g(x))$ and $f(x)$ is $\Omega(g(x))$
 - In this case we say $f(x)$ is *of order* $g(x)$

17 March 2009

Winsborough CS 2233 Lecture 15

3

Computational Complexity of Algorithms

- This is where Big-O, Big-Omega, and Big-Theta get used
- Complexity is measured principally in terms of two resources
 - Time Complexity
 - Space Complexity
 - Discussed more in course on data structures
- Worst-case complexity vs. Average-case

17 March 2009

Winsborough CS 2233 Lecture 15

4

Worst- versus Average-case Complexity

- Possible inputs are partitioned into cases that elicit the same behavior (*input cases*)
- *Worst-case complexity* analyzes the input case that maximizes cost of executing the algorithm
- *Average-case complexity* considers all input cases
 - The *average-case complexity* of executing an algorithm is:
 - The sum over all input cases of the cost of each case times its probability
 - Can be difficult to calculate
 - Assumes you know the probability of each input case
 - Also called *expected-case complexity* or, less formally, *expected cost*

17 March 2009

Winsborough CS 2233 Lecture 15

5

Linear Search Expected Cost

```
proc linear search(x:int; a1,a2,...,an: distinct ints)
  i := 1
  while (i ≤ n and x ≠ ai)
    i := i + 1
  if i ≤ n then location := i else location := 0
```

- Each case has x appearing in a different location
- When x is in position i , $1 \leq i \leq n$, $2i + 1$ comparisons are performed
- If we assume x appears somewhere in the list and that each position is equally probable,
 - The probability of each case is $1/n$
 - The expected cost is $\sum_{1 \leq i \leq n} (2i+1)(1/n)$

17 March 2009

Winsborough CS 2233 Lecture 15

6

Simplification of Sum

$$\begin{aligned}
 \sum_{1 \leq i \leq n} (2i+1)(1/n) &= (1/n) \sum_{1 \leq i \leq n} (2i+1) \\
 &= (1/n) (n + \sum_{1 \leq i \leq n} 2i) \\
 &= (1/n) (n + 2 \sum_{1 \leq i \leq n} i) \\
 &= (1/n) (n + 2((n(n+1))/2)) \\
 &= (1/n) (n + n^2 + n) \\
 &= 2 + n \quad \text{which is } \Theta(n)
 \end{aligned}$$

- Note that we could also use the theorems concerning products and sums
- The theorem about sums only applies to sums of constant size!

17 March 2009

Winsborough CS 2233 Lecture 15

7

| Complexity | Terminology | Example Algorithm |
|-------------------------|--|--|
| O(1) | Constant complexity | Find max value in first 100 elements |
| O(log n) | Logarithmic complexity | Binary search |
| O(n) | Linear | Linear search |
| O(n log n) | n log n | Merge sort** |
| O(n ^b) | Polynomial | Bubble sort |
| O(b ⁿ), b>1 | Exponential in number of prop. variables | Evaluate logical formulas via truth tables |
| O(n!) | Factorial | Traveling salesman** |

* Considered *tractable* ** Will not be covered in this class
17 March 2009 Winsborough CS 2233 Lecture 15

8

Theory versus Practice*

- Some algorithms that have high complexity in the worst case usually execute much faster than algorithms that have better worst-case complexity
- If the expensive cases are rare enough, such algorithms may be preferable
 - E.g., quicksort (discussed in Algorithms)

* They say that the difference between theory and practice is less in theory than it is in practice.

17 March 2009

Winsborough CS 2233 Lecture 15

9

Problems versus Algorithms

- The complexity of a problem is characterized by the complexity of the best algorithm for solving it
- It is always possible to give a sub-optimal algorithm
 - In Chapter 3, we will see that the sorting problem is actually n log n, even though the algorithms we have discussed so far are quadratic (n²)

17 March 2009

Winsborough CS 2233 Lecture 15

10

Undecidable Problems

- There are problems that cannot be solved by any algorithm
 - Such problems are called *undecidable*
 - The book calls them *unsolvable*
- Alan Turing first proved that the *halting problem* is undecidable
 - Given an algorithm and an input, determine whether the algorithm halts when executed on the input

17 March 2009

Winsborough CS 2233 Lecture 15

11

Problem Classes: P versus NP

- Problems in P (polynomial time) have polynomial-time algorithms
- Problems in NP (nondeterministic polynomial time) have no **known** polynomial-time algorithms, but for which a solution can be checked in polynomial time
 - While it remains open whether P = NP, most computer scientists consider this unlikely
- There is a sub-class of NP called NP-complete for which, if any such problem can be solved in polynomial time, they all can
 - Satisfiability of propositional formulas is an NP-complete problem
 - Given a truth assignment that makes the formula true, we can verify this in polynomial time
 - There is no known algorithm to find such assignments in polynomial time
 - If one were discovered, it would prove P = NP

17 March 2009

Winsborough CS 2233 Lecture 15

12