ECE 2574: Data Structures and Algorithms - Big-O Notation

C. L. Wyatt

Today we will look at a common way algorithms are described, as an upper-bound on their growth rate functions.

These are used to categorize algorithms roughly and can be used as a guide to algorithm selection.

- Properties of growth rates
- Order of growth
- Best, Worst, and Average cases
- Tractible algorithms and class P problems
- class NP problems
- Examples
- Project 3

Properties of growth rates

Lower order terms can be ignored.
Example:
Algorithm A has complexity n^2-3n +10
means
Algorithm A has complexity of order O(n^2)
Why? For some k and n > 0, k*n^2> n^2-3n + 10

Properties of growth rates

Using composition you can combine orders of growth.
O(O(f(n)) + O(g(n))) = O(f(n) + g(n))
Example:

- Algorithm A has complexity O(nˆ2)
- Algorithm B has complexity O(n)
- Algorithm C uses algorithm A and B sequentially.

means
Algorithm C has complexity of order O(nˆ2)

Properties of growth rates

Multiplicative constants can be ignored.
Example:
Algorithm A has complexity 1000nˆ2
means
Algorithm A has complexity of order O(nˆ2)
Note that the size of the constant may make the algorithm impractical, but the order is the same.

Polynomial closure

One reason polynomial complexity is a convenient measure is the closure property
Composition of polynomials of polynomials is a polynomial.
$f(n) = g(h(n))$ is polynomial if g and h are.
sum, differences, products of polynomials are polynomial.

best, worse, and average O(f(n))

Note that the order can be different for the best, worse, or average cases.
Example: linked list with head pointer only implementation of list retrieve

- is O(1) in the best case (index=1)
- is O(n) in the worst case (index = last)
- is O(n) in the average case

Tractable Algorithms

An algorithm is said to be tractable if it has a worst case polynomial complexity.
Such tractable algorithms solve problems that are in the class of P.
Example:

- the problem “retrieval from a list” is in the class P
- the problem “solve towers of Hanoi” is not in the class P

The input set can make a big difference.

Consider the problem of testing whether a given integer is prime (this is an example of a decision problem).
If n is the number we want to test it would appear the complexity would be at worst sqrt(n)
However, a more realistic measure of the input is the number of digits m in the number with respect to a base b $>=$ 2.
Thus, the worst case is O(bˆ(m/2)) (it is not known if prime testing is in class P)

The class NP

Consider a nondeterministic algorithm, one that is allowed to guess a solution.
This solution is then verified by another algorithm.
If the guess and the verification can be done with polynomial complexity, the algorithm is in the class NP (nondeterministic polynomial).
Note that P is a subset of NP.

Example of NP problems: sum of subsets

Given a set of input integers {s1, s2, ... sn} and a target sum C, is there a subset of integers whose sum is C?
The guess can just randomly select a subset.
The verification is just to sum the integers in the subset which has complexity O(n) in the worst case.
Does the combination always give a correct answer? if so then the problem is of the class NP.

Does P = NP?

A fundamental question in algorithm analysis is: are the set of problems in P the same as the set of problems in NP?
In other words does the ability to guess help?

When does complexity matter?

“Premature optimization is the root of all evil.” - Donald Knuth
So where do you focus your efforts?

- What is the frequency of the operation? Seldom used but critical operations must be efficient.
- Generally if n is small (how small?) then complexity matters less
- Time versus space.
- Consider the clarity of the code.

Sorting Algorithms

In order to get more familiar with complexity theory, we will use it to study several different sorting algorithms.

- Selection sort
- Bubble sort (next time)
- Insertion sort (next time)
- Merge sort (next time)
- Quick sort (next time)

Sorting algorithms are used often.

Searching, (e.g. binary search) often requires having the items sorted.
Tables and spread sheets often arrange rows/columns based on a key sort.
Consider sorting your University record based on your student ID number.

Internal versus External Sort

Sorting algorithms which do not require any auxiliary storage, that is they fit entirely in memory are called internal sorting algorithms. Sorting algorithms which do not fit into memory and must be accessed from a disk or external resource are called external sorting algorithms.

Order preserving sort

A sorting algorithm that preserves the original order of entries with keys that have the same value is called order preserving (stable).

Common Sorting Algorithms

Algorithm	Worst Case	Best Case
Selection Sort		
Bubble Sort		
Insertion Sort		
Merge Sort		
Quick Sort		

Selection sort works by iteratively finding the largest element in a list.

Given a list of keys {k1, k2, k3, ... kn}
First, locate the largest item in the list.
Next swap that key and the last key
Repeat on the list {k1, k2, k3, ... kn-1}
Stop when the list has size 1.

Selection Sort Example

```
12 5 237 64 39 78
12 5 78  64 39 237
12 5 39  64 78 237
12 5 39  64 78 237
12 5 39  64 78 237
5 12 39 64 78 237
```

Selection sort pseudocode (1-based indexing)

```
SelectionSort( in list:List )
n = list.length
for last = n down to 1
    // find the largest entry
    max = 1
    for i = 2 to last
        if (list[i] > list[max])
            max = i
    endfor

    // swap the entries
    temp = list[last]
    list[last] = list[max]
    list[max] = temp
endfor
```

Selection Sort: Number of compares and moves

The loop to find the maximum does a single compare(n-1) + (n-2) + (n-3) + ... 1 = n*(n-1)/2 times total. The loop does 3 moves to swap each element (n-1) times total. Together the number of comparison and moves is n*(n-1)/2 + 3*(n-1) or O(nˆ2)
Does it differ in the best or worse case ?

Common Sorting Algorithms

Algorithm	Worst Case	Best Case
Selection Sort	n^2	n^2
Bubble Sort		
Insertion Sort		
Merge Sort		
Quick Sort		

Next Actions and Reminders

- Read CH pp. 305-312 on basic sorting
- Project 3 is relased, due 10/31 by 11:55 pm.
- Midterm is Friday, in-class