CS 61B Data Structures and Programming Methodology

July 14, 2008

David Sun

Announcement

- Project 1 Due Tomorrow at 11:00a.m.
- Due to technical problems, for HW1 everyone will get credit.

Today

- More on asymptotic analysis

Analysis Example 1

- Problem 1
  - Given a set of ***p*** points, find the pair closest to each other.
- Algorithm 1:
  - Calculate the distance between each pair; return the minimum.

```
double minDistance = point[0].distance(point[1]);

/* Visit a pair (i, j) of points. */
for (int i = 0; i < point.length; i++) {
   for (int j = i + 1; j < point.length; j++) {
        double thisDistance = point[i].distance(point[j]);
        if (thisDistance < minDistance) {
               minDistance = thisDistance;
        }
   }
}
```

Iterates *p* times

Iterates *p/2* times on average

- There are *p (p - 1) / 2* pairs, and each pair takes constant time to examine. Therefore, worst- and best-case running times are in $\Theta(p^2)$.

Analysis Example 2

- Problem 2:
  - remove consecutive duplicates from an ints array of length ***N***
- Algorithm 2:

```
int i = 0, j = 0;
while (i < ints.length) {
     ints[j] = ints[i];
     do {
             i++;
     } while ((i < ints.length) && (ints[i] == ints[j]));
     j++;
}
```

Iterates up to *p* times,

Iterates up to *p* times

- Although we have a nest loop the running-time is **not** $\Theta(N^2)$. Why?
- The outer loop can iterate up to *ints.length* times, and so can the inner loop. But the index *i* advances on every iteration of the inner loop. It can't advance more than *ints.length* times before both loops end.
- So the worst-case running time of this algorithm is $\Theta(N)$ time.

Analysis Example 3

- Problem 3
  - Given 2 strings, tests if the second string is a substring of the first.
- Algorithm 3:

```
boolean occurs (String S, String X) {
    if (S.equals (X)) return true;
    if (S.length () <= X.length ()) return false;
    return
          occurs (S.substring (1), X) ||
          occurs (S.substring (0, S.length ()-1), X);
}
```

- What's the best case?
- What's the worst case?
- What's the complexity of the worst case?
- Consider a fixed size of *N*, N_0 Let *C(N)* be the worst-case cost of the algorithm.

$$C(N) = \begin{cases} 1, & \text{if } N \leq N_0, \\ 2C(N-1) & \text{if } N > N_0 \end{cases}$$

- *C(N)* grows exponentially

$$C(N) = 2C(N-1) = 2 \cdot 2C(N-2) = \ldots = \underbrace{2 \cdot 2 \cdots 2}_{N-N_0} \cdot 1 = 2^{N-N_0} \in \Theta(2^N)$$

Algorithm 4

- Problem 4
  - Finds if a String is in a sorted array of Strings.
- Algorithm 3:

```
boolean isIn (String X, String[] S, int L, int U) {
    if (L > U) return false;
    int M = (L+U )/2;
    int direct = X.compareTo (S[M]);
    if (direct < 0) return isIn (X, S, L, M-1);
    else if (direct > 0) return isIn (X, S, M+1, U);
    else return true;
}
```

- Consider a fixed size of *D*. Let *C(D)* be the worst-case cost of the algorithm
- The problem size is cut by half each time.

$$
\begin{aligned}
C(D) &= \begin{cases} 0, & \text{if } D \le 0, \\ 1 + C((D-1)/2), & \text{if } D > 0. \end{cases} \\
&= \underbrace{1+1+\ldots+1}_{k}+0 \\
&= k = \lceil \lg D \rceil \in \Theta(\lg D)
\end{aligned}
$$

Functions of Several Variables

Analysis Example 5

- Problem 5:
  - A matchmaking program for ***w*** women and ***m*** men.
- Algorithm 5:
  - Compare each woman with each man. Decide if they're compatible.
- Suppose each comparison takes *constant time* then the running time, $T(w, m)$, is in $\Theta(wm)$.
  - There exist constants c, d, W, and M, such that**:** $d\,wm \le T(w, m) \le c\,wm$ for every $w \ge W$ and $m \ge M$.
- $T(w, m)$ is NOT in $O(w^2)$, nor in $O(m^2)$, nor in $\Omega(w^2)$, nor in $\Omega(m^2)$.
- Every one of these possibilities is eliminated either by choosing $w >> m$ or $m >> w$. Conversely, w^2 is in neither $O(wm)$ nor $\Omega(wm)$.

Analysis Example 6

- Problem 6:
  - Suppose you have an array containing ***n*** music albums, sorted by title. You request a list of all albums whose titles begin with "The Best of"; suppose there are ***k*** such albums.
- Algorithm 6:

  Search for the first matching album with binary search. *log n*

  Walk (in both directions) to find the other matching albums. *k*
- Worst case:
  - Binary search: *log n* steps.
  - The complete list of k matching albums is found, each in constant time. Thus, the worst-case running time is in ***Θ(log n + k)*.**
- Can we simplify?
  - Because *k* can be as large as *n*, it is not dominated by the *log n* term.
  - Because *k* can be as small as 0 , it does not dominate the *log n* term.
  - Hence, there is no simpler expression.
  - The algorithm is *output-sensitive*, because the running time depends partly on the size *k* of the output.
- Best case:
  - Finds a match right away, *Θ (1+ k) = Θ(k).*

Analysis Example 7

- Problem 7: Find the ***k-th*** item in an ***n*-node** doubly-linked list.
- Algorithm 7:

  If k < 1 or k > n, report an error and return.

  Otherwise, compare k with n-k.

  If k <= n-k

  start at the beginning of the list and walk forward k-1 nodes.

  Otherwise

  start at the end of the list and walk backward n-k nodes.

- If $1 \leq k \leq n$, this algorithm takes $\Theta(\min\{k, n-k\})$ time (in all cases)
- This expression cannot be simplified: without knowing *k* and *n*, we cannot say that *k* dominates *n-k* or that *n-k* dominates *k*.

Some Intuition

- How big a problem can you solve in a given time?

Time (μsec) for problem size N	Max N Possible in 1 second	1 hour	1 month	1 century
$\lg N$	10^{300000}	$10^{1000000000}$	$10^{8 \cdot 10^{11}}$	$10^{9 \cdot 10^{14}}$
N	10^6	$3.6 \cdot 10^9$	$2.7 \cdot 10^{12}$	$3.2 \cdot 10^{15}$
$N \lg N$	63000	$1.3 \cdot 10^8$	$7.4 \cdot 10^{10}$	$6.9 \cdot 10^{13}$
N^2	1000	60000	$1.6 \cdot 10^6$	$5.6 \cdot 10^7$
N^3	100	1500	14000	150000
2^N	20	32	41	51