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.

```java
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;
        }
    }
}
```

- 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:

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

- 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
  
  $C(N) = 2C(N - 1) = 2 \cdot 2C(N - 2) = \ldots = \frac{2 \cdot 2 \cdots 2 \cdot 1}{N-N_0} = 2^{N-N_0} \in \Theta(2^N)$
Algorithm 4

• Problem 4
  – Finds if a String is in a sorted array of Strings.

• Algorithm 3:
  
  ```java
  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.

  \[
  C(D) = \begin{cases} 
  0, & \text{if } D \leq 0, \\
  1 + C((D-1)/2), & \text{if } D > 0.
  \end{cases}
  \]

  \[
  = \underbrace{1+1+\ldots+1}_{k}+0
  \]

  \[
  = k = \lceil \lg D \rceil \in \Theta(\lg D)
  \]
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:
    \[ d \, wm \leq T(w, m) \leq c \, wm \text{ for every } w \geq W \text{ and } m \geq M. \]

• \( T(w, m) \) 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 \( \Theta(\log n + k) \).

• 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, \( \Theta(1 + k) = \Theta(k) \).
Analysis Example 7

• Problem 7: Find the \textbf{k-th} item in an \textbf{n-node} doubly-linked list.

• Algorithm 7:
  \textit{If } k < 1 \textit{ or } k > n, \textit{ report an error and return.}
  \textit{Otherwise, compare } k \textit{ with } n-k.
  \textit{If } k \leq n-k
    \textit{start at the beginning of the list and walk forward } k-1 \textit{ nodes.}
  \textit{Otherwise}
    \textit{start at the end of the list and walk backward } n-k \textit{ 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 \textit{k} and \textit{n}, we cannot say that \textit{k} dominates \textit{n-k} or that \textit{n-k} dominates \textit{k}.
Some Intuition

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

<table>
<thead>
<tr>
<th>Time (μsec) for problem size $N$</th>
<th>1 second</th>
<th>Max $N$ Possible in 1 hour</th>
<th>Max $N$ Possible in 1 month</th>
<th>Max $N$ Possible in 1 century</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lg N$</td>
<td>$10^{300000}$</td>
<td>$10^{1000000000}$</td>
<td>$10^{8\cdot10^{11}}$</td>
<td>$10^{9\cdot10^{14}}$</td>
</tr>
<tr>
<td>$N$</td>
<td>$10^6$</td>
<td>$3.6 \cdot 10^9$</td>
<td>$2.7 \cdot 10^{12}$</td>
<td>$3.2 \cdot 10^{15}$</td>
</tr>
<tr>
<td>$N \lg N$</td>
<td>63000</td>
<td>$1.3 \cdot 10^8$</td>
<td>$7.4 \cdot 10^{10}$</td>
<td>$6.9 \cdot 10^{13}$</td>
</tr>
<tr>
<td>$N^2$</td>
<td>1000</td>
<td>60000</td>
<td>$1.6 \cdot 10^6$</td>
<td>$5.6 \cdot 10^7$</td>
</tr>
<tr>
<td>$N^3$</td>
<td>100</td>
<td>1500</td>
<td>14000</td>
<td>150000</td>
</tr>
<tr>
<td>$2^N$</td>
<td>20</td>
<td>32</td>
<td>41</td>
<td>51</td>
</tr>
</tbody>
</table>