Algorithm Analysis

CPE 103

Theresa Migler-VonDollen

Running Time

For any algorithm there are three important things to prove:

- Is it correct?
- How fast is it?
- How much space does it take?

Example:
Compare $f(x) = 5x + 100$ with $g(x) = x^2$

- If these functions represent running times, which is faster?
- We need to formalize what we mean by “faster”.
  - “Big O” notation.

Big O Notation

- Big O notation describes the limiting behavior of a function when the input gets very large.
- Big O notation characterizes functions according to their growth rates:
  - Functions with the same growth rate may be represented using the same O notation.
- When we make statements such as, $f(x) = 2x^3 - 7x + 14 = O(x^3)$:
  - The first equals sign really means equality.
  - The second equals sign represents set inclusion.

Big O Notation

Practical tricks:

- If $f(x)$ is a sum of several terms, then the one with the largest growth rate is kept, and all others omitted.
- If $f(x)$ is a product of several factors, any constants are omitted.

Example

For the following examples determine if $f = O(g)$, $f = \Omega(g)$, or both ($f = \Theta(g)$):

1. $f(x) = 4x^2$ and $g(x) = 1,000x^2 + 12x + 7$
2. $f(x) = x^2 3^x$ and $g(x) = 4^x$
3. $f(x) = 2^x$ and $g(x) = x!$
4. $f(x) = 10 \log x$ and $g(x) = \log x^2$

Running Time Calculations

General Rules:

- *for* loops:
  - The running time for a *for* loop is at most the running time of the statements inside the *for* loop times the number of iterations.
- Nested loops:
  - Analyze inside out.
- Consecutive statements:
  - Add.
- *if/else* statements:
  - Bounded by the running time of the test plus the running time of the larger (if/else) operation.

Running Time Calculations

Example:
What is the running time for the following pseudocode?

$x = 7$
for $i = 1$ to N **do**
    $x = 2 \times x + i$
return x

- $O(2 \times N + 2) = O(N)$

Running Time Calculations

Example:
What is the running time for the following pseudocode?

```
x = 12
for i = 1 to N do
    for j = 1 to N do
        x = x + i − j
return x
```

- $O(N \times 2N + 2) = O(2N^2) = O(N^2)$

Running Time Calculations

Example:
What is the running time for the following pseudocode?

```
x = 12
for i = 1 to N do
    for j = 37 to N do
        x = x + i - j
return x
```

- $O(N \times 2(N - 36) + 2) = O(2N^2 - 72N + 2) = O(N^2)$

Running Time Calculations

Example:
What is the running time for the following pseudocode?

```
x = 0
for i = 1 to N do
    x = x + 2 × i
y = 0
for j = 1 to N do
    y = j
return x + y
```

- $O(2 \times N + N + 4) = O(N)$

Running Time Calculations

Example:
What is the running time for the following pseudocode?

```
if x = 4 then
    for i = 1 to N do
        x = x + i
else
    for i = 1 to N do
        for j = 0 to N do
            x = x + j
```

- $\max\{O(N), O(N^2)\} + O(1) = O(N^2)$

Maximum Subsequence Sum

We will consider the following problem and analyze 3 different algorithms to solve it.

Maximum Subsequence Sum

Input: A list of integers, $a_1, a_2, a_3, \ldots a_n$.
Goal: Find the maximum value of $\sum_{k=i}^{j} a_k$.

Example:
Input: 5, -17, 12, 5, -10, 6, 4, 8, -5, -10, -17, 22, 1
The output should be 25 (a_3 through a_8).

Maximum Subsequence Sum - Algorithm

Cubic MSS

Input: A list of integers, $a_1, a_2, a_3, \ldots a_n$.
Output: The maximum value of $\sum_{k=i}^{j} a_k$.

```
maxSum = 0
for i = 1 to n do
    for j = i to n do
        thisSum = 0
        for k = i to j do
            thisSum = thisSum + a_k
        if thisSum > maxSum then
            maxSum = thisSum
return maxSum
```

Cubic MSS

- Is the algorithm correct?
- What is the running time?
  - $O(n^3)$

Maximum Subsequence Sum - Algorithm

Quadratic MSS

Input: A list of integers, $a_1, a_2, a_3, \ldots a_n$.
Output: The maximum value of $\sum_{k=i}^{j} a_k$.

```
maxSum = 0
for i = 1 to n do
    thisSum = 0
    for j = i to n do
        thisSum = thisSum + a_j
        if thisSum > maxSum then
            maxSum = thisSum
return maxSum
```

Quadratic MSS

- Is the algorithm correct?
- What is the running time?
  - $O(n^2)$

Maximum Subsequence Sum - Algorithm

Linear MSS

Input: A list of integers, $a_1, a_2, a_3, \ldots a_n$.
Output: The maximum value of $\sum_{k=i}^{j} a_k$.

$maxSum = 0$
$thisSum = 0$
for $i = 1$ to n **do**
    $thisSum = thisSum + a_i$
    if $thisSum > maxSum$ **then**
        $maxSum = thisSum$
    else if $thisSum < 0$ **then**
        $thisSum = 0$
return $maxSum$

Linear MSS

- Is the algorithm correct?
- What is the running time?
  - $O(n)$