Algorithms

Introduction

- Definition: A set of steps to accomplish a task

    to get the school from your home

    to find an item in a supermarket

- In CS, an algorithm is a set of instructions for a computer program to accomplish a task

    Google map uses a route finding alg to give you
    a route from your current location
    to a destination point.

Introduction

- design an algorithm to find the maximum number of a finite sequence of numbers (not sorted)
  - set a temporary variable, *temp*
  - set *temp* as the first element of the sequence
  - compare the second element of the sequence with *temp*: if the second is bigger than *temp*, set *temp* as the second; if not, do nothing; pass to the third one
  - compare the third element of the sequence with *temp*: if the third is bigger than *temp*, set *temp* as the third; if not, do nothing; pass to the fourth one
  - continue in this way till there is no more element in the sequence, and output *temp*

Introduction

MAX-INTEGER

input : $\{a_1, a_2, ..., a_n\}$
output: max of $\{a_1, a_2, ..., a_n\}$

```
max = a_1
for i = 2 to n
    if max < a_i
        max = a_i
return max
```

Introduction

Basic goals for an algorithm

- always correct
- always terminates
- has good performance
  - performance often draws line between what is possible and what is impossible

Introduction

How do we evaluate efficiency?

- using asymptotic analysis, we can evaluate the efficiency of an algorithm independent of the software and the hardware

Introduction

- Running time
  - depends on input (it's easy to search an element in a sorted sequence)
  - parameterized by the input size
  - It's desired an upper bound to guarantee the performance
- Two kinds of analysis for the running time
  - Worst case analysis (usually), maximum time on any input of size n
  - Average case analysis(sometimes), expected time over all inputs of size n

Big-O Notation

- used to estimate the number of operations the algorithm uses in terms of the size of its input

- enables us to determine whether it is practical to use the corresponding algorithm to solve the given problem, and to compare two algorithms in order to decide which one is more efficient

Definition : Let $f, g : \mathbb{Z}^{+} \to \mathbb{R}$ be two functions. If there are constants C and k such that $|f(x)| \leq C.|g(x)|$ for all $x \in \mathbb{Z}$ where $x \geq k$, we say that g dominates f (or f is big-O of g),

$$f(x) = O(g(x))$$

Big-O Notation

- $f, g : \mathbb{Z}^+ \to \mathbb{R}$, $f(x) = 5x$ and $g(x) = x^2$.
  - $f(1) = 5, f(2) = 10, f(3) = 15, f(4) = 20, f(5) = 25, \ldots$
    $g(1) = 1, g(2) = 4, \; g(3) = 9, \;\; g(4) = 16, g(5) = 25, \ldots$
  - for $n \geq 5$, $n^2 \geq 5n \to |f(x)| \leq |g(x)|$
  - for $C = 1$ and $k = 5$,

    $|f(x)| \leq C.|g(x)|$ for all $x \geq k$. Thus, $f(x) = O(g(x))$.
  - C and k don't have to be unique

Big-O Notation

- $f, g : \mathbb{Z}^+ \to \mathbb{R}$, $f(x) = 5x^2 + 3x + 1$ and $g(x) = x^2$.

$|f(x)| = |5x^2 + 3x + 1| = 5x^2 + 3x + 1$

$\leq 5x^2 + 3x^2 + x^2 = 9x^2 = 9|g(x)|$

for $C = 9$ and $k = 1$,

$|f(x)| \leq C.|g(x)|$ for all $x \geq k$. Thus, $f(x) = O(g(x))$.

$|g(x)| = |x^2| = x^2 \leq 5x^2 \leq 5x^2 + 3x + 1 = |f(x)|$

for $C = 1$ and $k = 1$,

$|g(x)| \leq C.|f(x)|$ for all $x \geq k$. Thus, $g(x) = O(f(x))$.

Big-O Notation

- $f, g : \mathbb{Z}^{+} \to \mathbb{R}$, $f(x) = 7x^2$ and $g(x) = x^3$.

  $|f(x)| = |7x^2| = 7x^2 \leq 7x^3 = 7|g(x)|$

  for $C = 7$ and $k = 1$,

  $|f(x)| \leq C.|g(x)|$ for all $x \geq k$. Thus, $f(x) = O(g(x))$.

  $|g(x)| = |x^3| = x^3 \leq C.7.x^2 = C.|f(x)| \to x \leq C.7$ for all $x \geq k$

  there cannot be any C and k that satisfy this inequality.

Big-O Notation

- $f, g : \mathbb{Z}^+ \to \mathbb{R}$, $f(x) = 4x^3 - 9x^2 + 3x + 2$ and $g(x) = x^3$.

$$
\begin{aligned}
|f(x)| = |4x^3 - 9x^2 + 3x + 2| &\le |4x^3| + |-9x^2| + |3x| + |2| \\
&\le 4x^3 + 9x^3 + 3x^3 + 2x^3 \\
&= 18x^3 = 18|g(x)|
\end{aligned}
$$

for $C = 18$ and $k = 1$,

$|f(x)| \le C.|g(x)|$ for all $x \ge k$. Thus, $f(x) = O(g(x))$.

$|g(x)| = |x^3| \le C.|4x^3 - 9x^2 + 3x + 2| = C.|f(x)|$.

Assume $C = 1$, then $|x^3| \le |4x^3 - 9x^2 + 3x + 2|$

$$|x^3| \le |x^3 + 3x^3 - 9x^2 + 3x + 2|$$

$3x^3 - 9x^2 \ge 0 \to x^2(3x - 9) \ge 0$ for all $x \ge 3$

for $C = 1$ and $k = 3$,

$|g(x)| \le C.|f(x)|$ for all $x \ge k$. Thus, $g(x) = O(f(x))$.

Big-O Notation

- $f : \mathbb{Z}^+ \to \mathbb{R}$, $f(x) = a_t x^t + a_{t-1} x^{t-1} + \ldots + a_1 x + a_0$

$$
\begin{aligned}
|f(x)| = |a_t x^t + a_{t-1} x^{t-1} + \ldots + a_1 x + a_0| &\leq |a_t x^t| + \ldots + |a_1 x| + |a_0| \\
&= |a_t| . x^t + \ldots + |a_1| . x + |a_0| \\
&\leq |a_t| . x^t + \ldots + |a_1| . x^t + |a_0| . x^t \\
&\leq (|a_t| + \ldots + |a_1| + |a_0|) . x^t = C . |x^t|
\end{aligned}
$$

for $C = |a_t| + \ldots + |a_1| + |a_0|$ and $k = 1$,

$|f(x)| \leq C . |x^t|$ for all $x \geq k$. Thus, $f(x) = O(x^t)$

Big-O Notation

- $f : \mathbb{Z}^+ \to \mathbb{R}$, $f(x) = 1 + 2 + \ldots + x$

  $|f(x)| = |1 + 2 + \ldots + x\,| = 1 + 2 + \ldots + x \leq x + x + \ldots + x = |x^2|$
  for $C = 1$ and $k = 1$,
  $|f(x)| \leq C.|x^2|$ for all $x \geq k$. Thus, $f(x) = O(x^2)$

- $f : \mathbb{Z}^+ \to \mathbb{R}$, $f(x) = 1^2 + 2^2 + \ldots + x^2$

  $|f(x)| = |1^2 + 2^2 + \ldots + x^2| = 1^2 + 2^2 + \ldots + x^2 \leq x^2 + x^2 + \ldots + x^2 = |x^3|$
  for $C = 1$ and $k = 1$,
  $|f(x)| \leq C.|x^3|$ for all $x \geq k$. Thus, $f(x) = O(x^3)$

- $f : \mathbb{Z}^+ \to \mathbb{R}$, $f(x) = 1^t + 2^t + \ldots + x^t$

  $|f(x)| = |1^t + 2^t + \ldots + x^t| = 1^t + 2^t + \ldots + x^t \leq x^t + x^t + \ldots + x^t = |x^{t+1}|$
  for $C = 1$ and $k = 1$,
  $|f(x)| \leq C.|x^{t+1}|$ for all $x \geq k$. Thus, $f(x) = O(x^{t+1})$

Big-O Notation

- $f : \mathbb{Z}^+ \to \mathbb{R}$, $f(x) = 1.2.\ldots.x = x!$

  $|f(x)| = |1.2.\ldots.x| = 1.2.\ldots.x \leq x.x.\ldots.x = |x^x|$

  for $C = 1$ and $k = 1$,

  $|f(x)| \leq C.|x^x|$ for all $x \geq k$. Thus, $f(x) = O(x^x)$

- $f : \mathbb{Z}^+ \to \mathbb{R}$, $f(x) = (\log x)!$

  $|f(x)| = |1.2.\ldots.\log x| = 1.2.\ldots.\log x \leq \log x \ldots.\log x = \left|\log x^{\log x}\right|$

  for $C = 1$ and $k = 1$,

  $|f(x)| \leq C.|\log x.\log x|$ for all $x \geq k$. Thus, $f(x) = O(\log^2 x)$

- use smallest possible function for big-O notation

1	$\log n$	n	$n \log n$	n^2	n^t	2^n	$n!$
constant	logarithmic	linear		quadratic	polynomial	exponential	factorial

Big-O Notation

- $f_1(n) = O(g_1(n))$ and $f_2(n) = O\left(g_2(n)\right)$

$$
\begin{aligned}
|f_1(n) + f_2(n)| &\leq |f_1(n)| + |f_2(n)| \\
&\leq C_1|g_1(n)| + C_2|g_2(n)| \\
&\leq C_1|g(n)| + C_2|g(n)| \quad \text{where} \quad g(n) = max\,\{g_1(n), g_2(n)\} \\
&= (C_1 + C_2)|g(n)|
\end{aligned}
$$

$$f_1(n) + f_2(n) = O(max\,\{g_1(n), g_2(n)\})$$

$$f_1(n).f_2(n) = O(g_1(n).g_2(n))$$

- $f(n) = (n+1)\log(n^2+1) + 3n^2$

$$O(n) \qquad O(\log n) \qquad O(n^2)$$

$$
\begin{aligned}
\log(n^2+1) &\leq \log(2n^2) \\
&= \log 2 + \log n^2 \\
&= \log 2 + 2\log n \\
&\leq 3\log n
\end{aligned}
$$

$$f(n) = O(n^2)$$

Worst-Case Analysis

5 op

MAX-INTEGER

input : $\{a_1, a_2, ..., a_n\}$
output: max of $\{a_1, a_2, ..., a_n\}$

max = a_1 —— 1 op
for i = 2 to n —— n-1 times
    if max < a_i
        max = a_i —— 2 op
return max

$$f(n) = 2(n-1) + 1 = 2n - 1$$
$$f(n) = O(n)$$

2, 5, 11, 20, 24, 37, 38, 45

max = 2

i = 2

max < 5
max = 5

i = 3

max < 11
max = 11

Worst-Case Analysis

LINEAR-SEARCH

input : $\{a_1, a_2, \ldots, a_n; x\}$
output: location

$k = 1$ 2 op
$loc = 0$
while $k \leq n$ n times
  if $x = a_k$ 3 op
    $loc = k$
  $k = k + 1$
return k

$f(n) = 3n + 2$ (or $3n + 3$)
$f(n) = O(n)$

LINEAR-SEARCH

input : $\{a_1, a_2, \ldots, a_n; x\}$
output: location

$loc = 0$ 1 op
for i = 1 to n n times
  if $x = a_i$ 1 op
    $loc = i$
return loc

$f(n) = n + 1$ (or $n + 2$)
$f(n) = O(n)$

Worst-Case Analysis

BINARY-SEARCH

11 op

input : $\{a_1 < a_2 < \ldots < a_n; x\}$
output: location

2, 5, 11, 20, 24, 37, 38, 45; 11

$i = 1$
$j = n$
$loc = 0$
while $i \leq j$
    $m = \lfloor (i + j)/2 \rfloor$
    if $x = a_m$
        $loc = m$
    elseif $x > a_m$
        $i = m + 1$
    else
        $j = m$
return loc

$i = 1$
$j = 8$
$loc = 0$

$m = \lfloor (1 + 8)/2 \rfloor = 4$
$11 \neq 20$
$x \not> 20$
$j = 4$

$m = \lfloor (1 + 4)/2 \rfloor = 2$
$11 \neq 5$
$x > 5$
$i = 3$

Worst-Case Analysis

BINARY-SEARCH

input : $\{a_1 < a_2 < \ldots < a_n; x\}$
output: location

$i = 1$
$j = n$
$loc = 0$

3 op

while $i \leq j$

k times

    $m = \lfloor (i + j)/2 \rfloor$
    if $x = a_m$
        $loc = m$
    elseif $x > a_m$
        $i = m + 1$
    else
        $j = m$
return loc

4 op

$$n \rightarrow n/2 \rightarrow n/4 \rightarrow \cdots \rightarrow 1$$

$$2^k < n < 2^{k+1}$$

$$k = \log n$$

$$f(n) = 4k + 3 \; (\, or \; 4k + 4)$$
$$f(n) = 4 \log n + 3$$
$$f(n) = O(\log n)$$

Average-Case Analysis

LINEAR-SEARCH

input : $\{a_1, a_2, \ldots, a_n; x\}$
output: location

for i = 1 to n
    if $x = a_i$
        return i
return 0

- if $x = a_1$, then the algorithm terminates after 2 operations

  if $x = a_2$, then the algorithm terminates after 3 operations

  $\vdots$

  if $x = a_i$, then the algorithm terminates after $i + 1$ operation

  $\vdots$

  if $x = a_n$, then the algorithm terminates after $n + 1$ operations

  if$x \notin L$, then the algorithm terminates after $n + 1$ operations

- let p be the probability that $x \in L$, and $q = 1 - p$ be the probability that $x \notin L$
- for each element a_i, the probability that $x = a_i$ is p/n
- the expected value for the number of operations

$$E(X) = \sum p(s).X(s)$$
$$= 2.\frac{p}{n} + 3.\frac{p}{n} + \ldots + (n+1).\frac{p}{n} + (n+1).q = p\frac{(n+3)}{2} + q.(n+1)$$

- for $p = 1$ and $q = 0$
  $E(X) = (n+3)/2$
- for $p = 0$ and $q = 1$
  $E(X) = n + 1$
- for $p = q = 1/2$
  $E(X) = (3n+5)/4$