**Steven Murr**

*HW 3.3*

*Problems = { 5, 7, 13, 15, 19, 20, 22ab }*

5) How many comparisons are used by the algorithm given in Exercise 16 of Section 3.1 to find the smallest natural number in a sequence of $n$ natural numbers.
Describe an algorithm for finding the smallest integer in a finite sequence of natural numbers.

> **procedure** smallest integer($a_1, a_2 ... a_n$: natural numbers) Declare a procedure/function called smallest integer that takes in a finite list of natural numbers
> *least* := $a_1$ least gets the first value in the list
> **for** i := 2 **to** n set the variable i to 2 and iterate up to the nth element in the list.
>     **if** least > $a_i$ **then** *least* := $a_i$ if the value is least is greater than the currently iterated item, then least gets the value of the currently iterated item.
> **return** *least* return the variable least to the functions caller

Each time through the loop we are evaluating if $i <= n$ to continue the loop and $least > a_i$ to determine if the $i'th$ integer is less than what's in least. There is also one last comparison of $i <= n$ to exit the loop giving us: $2(n-1)+1$ which becomes $2n-1$. Thus, $2n-1$ comparisons are used to find the smallest natural number in a sequence of n natural numbers.

7) Suppose that an element is known to be among the first four elements in a list of 32 elements. Would a linear search or a binary search locate this element more rapidly?

With a search set with 32 elements, *a binary search* will require at **least** 4 iterations through the algorithm to find an element as shown by the following:

Enter a number to find: 4
Given the set: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Comparison looked like 1 < 32
The result of the floor function is 16
i = 1
j = 16
m = 16
Comparison looked like 1 < 16
The result of the floor function is 8
i = 1
j = 8
m = 8
Comparison looked like 1 < 8
The result of the floor function is 4
i = 1
j = 4
m = 4
Comparison looked like 1 < 4
The result of the floor function is 2
i = 3
j = 4
m = 2
Comparison looked like 3 < 4
The result of the floor function is 3
i = 3
j = 3
m = 3
Found 4 at the 3th position

A *linear search* will require at the **most** 4 iterations (worse case) through the algorithm as shown by the following:

Enter an integer to search: 4
4 was found at position 3
4 times through the loop.

Given the above information, linear search will find the element faster 75% of the time, given the four elements have equal probability of appearing as a search result.

13) The conventional algorithm for evaluating a polynomial $a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$ at $x = c$ can be expressed in pseudocode by:

**procedure** *polynomial* $(c, a_0, a_1 ..., a_n$ :real numbers)
 $power := 1$
 $y := a_0$
 **for** $i := 1$ **to** n
  $power := power * c$
  $y := y + a_i * power$
 **return** $y\{y = a_n c^n + a_n - 1c^n - 1 + .... + a_1 c + a_0\}$

where the final value of y is the value of the polynomial at $x = c$
a) Evaluate $3x^2 + x + 1$ at $x = 2$ by working through each step of the algorithm showing the values assigned at each assignment step.

$a_0 = 1, a_1 = 1, a_2 = 3$ Just wanted to show what the variables were

Begin execution:
x = c = 2
power = 1
y = 1
i = 1
power = 1 * 2 = 2
y = 1 + 1 * 2 = 3
i = 2
power = 2 * 2 = 4
y = 3 + 3*4 = 15

b) Exactly how many multiplications and additions are used to evaluate a polynomial of degree $n$ at $x = c$? (Do not count additions used to increment the loop variable).
 $2n$ multiplications and $n$ additions.

19) How much time does an algorithm using $2^{50}$ operations need if each operation takes these amounts of time?
 a) $10^-6s$
  $(2^{50})(10^{-6}) = 1.125899907E9$ / by seconds in a year ≈ 36 years
 b) $10^{-9}s$
  $(2^{50})(10^{-9}) = 1125899.907$ / 86400(seconds in a day) ≈ 13 days
 c) $10^-12$
  $(2^{50})(10^{-12}) = 1125.89907$ / 60(seconds in a minute) ≈ 19 minutes.

20) What is the effect in the time required to solve a problem when you double the size of the input from $n$ to $2n$, assuming that the number of milliseconds the algorithm uses to solve the problem with input size $n$ is each of these function? [Express your answer in the simplest form possible, either as a ratio or a

difference. Your answer may be a function of $n$ or a constant]

a) $loglog\, n$

We take the ratio $\frac{loglog(2n)}{loglogn}$ This reduces to $\frac{2}{1}$, therefore its twice as many milliseconds.

b) $log\, n$

$log\, n$ becomes $log\, 2n$. We expand it as $log(2n) = log(2) + log(n)$ We take the difference as $log(2n) - log(n) = log_2(2)$ Since this is a log base 2 we can evaluate $log_2(2)$ to be 1 which is constant time increase.

c) $100n$

We take the ratio of 100(2n) over 100n like $\frac{200n}{100n}$ After canceling we are left with $\frac{2}{1}$. It's twice as many milliseconds.

d) $n\, log\, n$ We take the ratio of $\frac{2nlog2n}{nlogn}$ This reduces to $\frac{2}{1}$ therefore twice as many milliseconds.

$2n\, log\, 2n$. Thus $n\, log\, n$ more milliseconds.

e) $n^2$

We take the ratio of 2n to n like, $\frac{4n^2}{n^2}$ We subtract the $n^2$ and are left with $\frac{4}{1}$. It's 4 times as many milliseconds.

f) $n^3$

We take the ratio of 2n to n like, $\frac{8n^3}{n^3}$ We subtract the $n^3$ and are left with $\frac{8}{1}$. It's 8 times as many milliseconds.

g) $2^n$

We take the ratio of 2n to n like, $\frac{2^{2n}}{2^n}$ We subtract the n exponents and we are left with $2^n$ more milliseconds.

Determine the least number of comparisons, or best-case performance,

a) Required to find the maximum of a sequence of n integers, using Algorithm 1 of Section 3.1

Best case running time is n, the entire set must be viewed to know if we've seen the largest integer in the set.

b) Used to locate an element in a list of n terms with a linear search.

Best case running time is 1, if the element we are looking for is the first element in the set.

c) Used to locate an element in a list of n terms using a binary search.

Best case running time is 1, if the element we're looking for is the mid point of the set.