Assignment One

Name	
Student number	

Direction:

Please answer all the questions below and hand in your answers before the due day. All work, must be handed in ON TIME.

Due Date:

March. 27, 2017

Please hand it in by the class time.

Questions:

1. (20 marks) For each group of $f(n)$ and $g(n)$, determine $f(n)=O(g(n))$, $f(n)=\Theta(g(n))$, or $f(n)=\Omega(g(n))$. And why?

(a) $f(n)=n^{1/3}$, $g(n)=(\ln n)^2$

(b) $f(n)=2^n$, $g(n)= n\log n$

Answer:

(a) $n^{1/3}=\Omega((\ln n)^2)$

(b) $2^n=\Omega(n\log n)$

2. (20 marks)

(a) Suppose the running time for some *AlgorithmA* is $T(n)=3\times2^n$ at the input size of n. Its operation time on Computer1 is t seconds. Computer2 has the operation capability which is 64 times of Computer1. Then, the *AlgorithmA* can solve problem of what input size $\boldsymbol{n_1}$ on Computer2 in the same time t?

(b) If the time efficiency for *AlgorithmA* is promoted as $T(n)=n^2$, and other conditions remain unchanged. Then, the *AlgorithmA* can solve problem of what input size $\boldsymbol{n_2}$ on Computer2 in the same time t?

(c) If the time efficiency for *AlgorithmA* is promoted as $T(n)=8$, and other conditions remain unchanged. Then, the *AlgorithmA* can solve problem of what input size $\boldsymbol{n_3}$ on Computer2 in the same time t?

Answer:

(a) $n_1=n+6$

(b) n_2=8n

(c) any input size

(a) Suppose computer2 can solve input size n1 in the time t, so we have

$t = 3\times2^{n} = 3\times2^{n1}/64$ so $n_1 = n+6$

(b) Suppose computer2 can solve input size n1 in the time t, so we have

$t = n^2 = n_2^2/64$ so $n_2 = 8n$

(c) $T(n)$ = Constant, so the input can be any size.

3. (20 marks) Gaussian elimination, the classic algorithm for solving system of n linear equations in n unkowns, requires about $\frac{1}{3}n^3$ multiplications, which is the algorithm's basic operation.

(a) How much longer should you expect Gaussian elimination to work on a system of 1000 equations versus a system of 500 equations.

(a) You are considering buying a computer that is 1000 times faster than the one you currently have. By what factor will the faster computer increase the size of system solvable in the same amount of time as on the old computer?

Answer:

(a) $\frac{t_1}{t_2} = \frac{n_1^3}{n_2^3} = 2^3 = 8$

(b) $\frac{1}{3}n_1^3 = \frac{1}{3}n_2^3/1000 \Rightarrow n_1^3 = \frac{n_2^3}{1000} \Rightarrow n_1^3 = \frac{n_2^3}{10^3} \Rightarrow n_1 = \frac{n_2}{10}$

4. (10 marks) Solve the following recurrence relations.

$x(n) = x(n-1) + 2n$ for $n > 0$, $x(0) = 0$

Answer:

$x(n) = x(n-1) + 2n$

$= [\, x(n-2) + 2\,(n-1)] + 2\,n$

$= [\, x(n-3) + 2(n-2)\,] + 2\,(n-1) + 2n$

… …

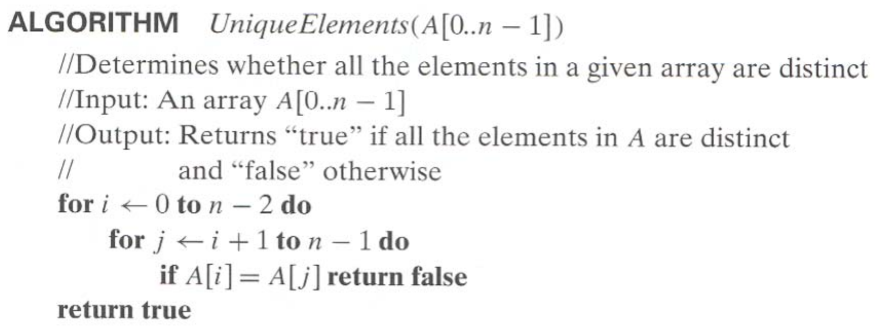
$= x(0) + (\,1+2+3+\ldots+(n-2)+(n-1)+n\,)*2$

$= (1+2+3+\ldots+n)*2$

$= n(n+1)$

5. (30 marks) Element uniqueness problem:

To determine whether all the elements in a given array are distinct.



```
ALGORITHM  UniqueElements(A[0..n − 1])
    //Determines whether all the elements in a given array are distinct
    //Input: An array A[0..n − 1]
    //Output: Returns "true" if all the elements in A are distinct
    //         and "false" otherwise
    for i ← 0 to n − 2 do
        for j ← i + 1 to n − 1 do
            if A[i] = A[j] return false
    return true
```

(a) Please define the input size;

(b) Set up summation for *C(n)* reflecting the number of times the algorithm's basic operation is executed

(c) Estimate the order of growth for *C(n)*, and why? (give your deduction process)

Answer:

(a) the input size is the length of the array: *n*.

(b)

最好的情况下只要进行一次比较，即发现相同元素。

最坏情况下，要进行 C(n) = (n-1) + (n-2) + … + 1 = (n-1+1)(n-1)/2 = n(n-1)/2 次比较。

从 1 次比较到 n(n-1)/2 次比较，每种情况下的出现的概率都是一样的，所以平均要进行比较的次数为：

C（n）平均= [1 + 2 + … + n(n-1)/2] / n(n-1)/2 = $(n^2 – n+2)/4$

所以复杂度是 $\Theta(n^2)$