Department of Computer Science and Engineering

PES UNIVERSITY

UE19CS251: Design and Analysis of Algorithm (4-0-0-4-4)

Q.1 Design an algorithm for swapping two 3 digit non-zero integers *n*, *m*. Besides using arithmetic operations, your algorithm should not use any temporary variable
Solution

ALGORITHM Exchange valueswithoutT(a,b)

//exchange the two values without using temporary variable

//Input:two numbers a,b.

//Output:exchange values of a,b

a=a+b;

b=a-b;

a=a-b;

Q.2 Design an algorithm for computing gcd(*m*, *n*) using Euclid's algorithm.

Solution

ALGORITHM Euclid (m,n)

// Computes gcd(m.n) by Euclid's algorithm

// Input: Two nonnegative, not-both-zero integers m and n

// Output : Greatest common divisor of m and n

while n ≠ 0 do

r ← m mod n

m ← n

n ← r

return m

Q.3 Write a pseudocode for an algorithm for finding real roots of equation $ax^2 + bx + c = 0$ for arbitrary real coefficients a, b, and c.
Solution:

Algorithm Quadratic(a, b, c)
//The algorithm finds real roots of equation ax2 + bx + c = 0
//Input: Real coefficients a, b, c
//Output: The real roots of the equation or a message about their absence
if a _= 0
D ← b ∗ b − 4 ∗ a ∗ c
if D >0
temp ← 2 ∗ a
x1 ← (−b + sqrt(D))/temp
x2 ← (−b − sqrt(D))/temp

return x1, x2
else if D = 0 return −b/(2 ∗ a)
else return ‘no real roots’
else //a = 0
if b _= 0 return −c/b
else //a = b = 0
if c = 0 return ‘all real numbers’
else return ‘no real roots’

Q.4 Design an algorithm to convert a binary number to a decimal integer.

Solution

Start

input n1.

s: = 0.

i: =0

r: = n1%10.

s: =s+ r* .

i++

n1 :=n1/10.

if n1<>0 then goto step 5.

Print s

Q.5 Consider the following algorithm for the searching problem:
Solution

ALGORITHM Linearsearch *(A*[0, ..*n* − 1], key)
//Searches an array for a key value by Linear search
//Input: Array *A*[0..*n* − 1] of values and a key value to search

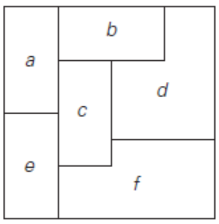
//Output: Returns index if search is successful
for *i* ← 0 **to** *n* − 1 **do**
if (key == *A*[*i*])
return *i*

a. Apply this algorithm to search the list 10, 92, 38, 74, 56, 19, 82, 37 for a key value 74.
b. Is this algorithm efficient?
c. When can this algorithm be used?

Q.6 Design a simple algorithm for string matching problem

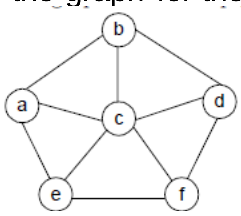
Solution: Align the pattern with the beginning of the text. Compare the corresponding characters of the pattern and the text left-to right until either all the pattern characters are matched (then stop–the search is successful) or the algorithm runs out of the text's characters (then stop–the search is unsuccessful) or a mismatching pair of characters is encountered.
In the latter case, shift the pattern one position to the right and resume the comparisons.

Q.7 Consider the following map:



a. Explain how we can use the graph-coloring problem to color the map so that no two neighboring regions are colored the same.
b. Use your answer to part (a) to color the map with the smallest number of colors.

Solution:
Create a graph whose vertices represent the map's regions and the edges connect two vertices if and only if the corresponding regions have a common border (and therefore cannot be colored the same color). Here is the graph for the map given:



Solving the graph colouring problem for this graph yields the map's colouring with the smallest number of colours possible.
b. Without loss of generality, we can assign colours 1 and 2 to vertices c and a, respectively. This forces the following colour assignment to the remaining vertices: 3 to b, 2 to d, 3 to f, 4 to e. Thus, the smallest number

of colours needed for this map is four.

Q.8 For each of the following algorithms, indicate (i) a natural size metric for its inputs; (ii) its basic operation; (iii) whether the basic operation count can be different for inputs of the same size:
a. computing the sum of *n* numbers
b. computing *n*!
c. finding the largest element in a list of *n* numbers
d. Euclid's algorithm

Solution:

a. (i) n; (ii) addition of two numbers; (iii) no
b. (i) the magnitude of n, i.e., the number of bits in its binary representation;
(ii) multiplication of two integers; (iii) no
c. (i) n; (ii) comparison of two numbers; (iii) no (for the standard list scanning algorithm)
d. (i) either the magnitude of the larger of two input numbers, or the magnitude of the smaller of two input numbers, or the sum of the magnitudes of two input numbers; (ii) modulo division; (iii) yes

Q.9 Define time complexity and space complexity. Write an algorithm for adding '*n*' natural numbers and find the time and space required by that algorithm

Solution:
The time complexity of a problem is the number of steps that it takes to solve an instance of the problem as a function of the size of the input (usually measured in bits), using the most efficient algorithm. The space complexity of a problem is a related concept that measures the amount of space, or memory required by the algorithm. The space complexity for adding sum of n numbers denoted by S (n) which is n+3. The time complexity is denoted by T (n) and it is 2n+3.

Q.10 For each of the following functions, indicate how much the function's value will change if its argument is increased fourfold.
a. log2 *n* **b.**√*n* **c.** *n* **d.** *n*2 **e.** *n*3 **f.** 2*n*
Solution:

a. 2
b. 2
c. 4
d. 4^2
e. 4^3
f. $(2^n)^3$

Q.11 Compare the two functions 2^n and n^2 for various values of *n*. Determine when will the second function become the same, smaller, and larger than the first function.

Solution:

- n=2, then we have the same value for both the functions.

- n>2, the first function is smaller the second.
- n<2, the first function is greater than the second.

Q.12 Use the most appropriate notation among *O*, Theta and omega to indicate the time efficiency class of binary search
a. in the worst case.
b. in the best case.
c. in the average case.
Solution:

		Successful	Unsuccessful
a	In the Worst case	Θ (logn)	Θ (logn)
b	In the Best case	Θ (1)	Θ (logn)
c	In the Average case	Θ (logn)	Θ (logn)

Q.13 From the following equalities, indicate the ones that are incorrect?

a. $6n^2 - 8n = \Theta(n^2)$ **b.** $12n^2 + 8 = O(n)$
c. $3n^2 3^n + n \log n = \Theta(n^2 3^n)$ **d.** $3n^2 \log n = \Theta(n^2)$

Solution:
a) $6n^2 - 8n = \Theta(n^2)$ correct
b) $12n^2 + 8 = O(n)$ incorrect
c) $3n^2 3^n + n\log n = \Theta(n^2 3^n)$ correct
d) $3n^2 \log n = \Theta(n^2)$ incorrect

Q.14 For each of the following functions, indicate the class Theta*(g(n))* the function belongs to. (Use the simplest *g(n)* possible in your answers.) Prove your assertions.

a. $(n^2+1)^{10}$ **b.** $\sqrt{10n^2+7n+3}$

Solutions:

$$\lim_{n\to\infty} \frac{(n^2+1)^{10}}{n^{20}} = \lim_{n\to\infty} \frac{(n^2+1)^{10}}{(n^2)^{10}} = \lim_{n\to\infty} \left(\frac{n^2+1}{n^2}\right)^{10} == \lim_{n\to\infty} \left(1+\frac{1}{n^2}\right)^{10} = 1.$$

Hence $(n^2+1)^{10} \in \Theta(n^{20})$.

$$\lim_{n\to\infty} \frac{\sqrt{10n^2+7n+3}}{n} = \lim_{n\to\infty} \sqrt{\frac{10n^2+7n+3}{n^2}} = \lim_{n\to\infty} \sqrt{10+\frac{7}{n}+\frac{3}{n^2}} = \sqrt{10}.$$

Hence $\sqrt{10n^2+7n+3} \in \Theta(n)$.

Q.15 Arrange the following functions according to their order of decay (from the highest to the lowest)

$(n+1)!2^{3n},\ 2n^4+2n^3+4,\ n\log n,\ \log n, 6n, 8n^2.$

Solution:
Order of decay: $(n+1)!\,2^{3n}$, $2n^4+2n^3+4$, $8n^2$, nlogn, 6n, logn

Q.16. algo what(a[l ..r] , l, r)
```
 if l = r then
                return a[l]
  else if a[l] > a[r] then
                return what(a, l + 1, r)
  else
                return what(a, l, r - 1)
```

i) what does the given function do?
ii) What is the basic operation?
iii) What is the basic size?
iv) express and solve the recurrence relation for number of operations?

Solution:
I) finds the min in the array section l to r
ii)comparison a[l] <a[r]
iii) # of elements in the array section : r – l + 1
iv) one possible solution
let n = r – l + 1
c(1) = 0
c(n) = 1 + c(n – 1)
c(n) = 1 + 1 + c(n-2)
c(n) = n – 1 + c(1) = n – 1 = r – l

Q.17 Consider the following algorithm:
ALGORITHM *Sum (n)*
//Input: A nonnegative integer *n*
$S \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
$S \leftarrow S + i$
return S
a. What does this algorithm compute?
b. What is its basic operation?
c. How many times is the basic operation executed?

Solution:
(a) sum.
(b) addition.
(c) n

Q.18 Consider the following algorithm
ALGORITHM *GE*($A[0..n-1, 0..n]$)
//Input: An n-by-$n+1$ matrix $A[0..n-1, 0..n]$ of real numbers
for $i \leftarrow 0$ **to** $n-2$ **do**
for $j \leftarrow i+1$ **to** $n-1$ **do**
for $k \leftarrow i$ **to** n **do**
$A[j, k] \leftarrow A[j, k] - A[i, k] * A[j, i] / A[i, i]$
a. Find the time efficiency class of this algorithm.
Solution:
The number of multiplications M(n) and the number of divisions D(n) made by the algorithm are given by the same sum:

$$
\begin{aligned}
M(n) &= D(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=i}^{n} 1 = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} (n-i+1) = \\
&= \sum_{i=0}^{n-2} (n-i+1)(n-1-(i+1)+1) = \sum_{i=0}^{n-2} (n-i+1)(n-i-1) \\
&= (n+1)(n-1) + n(n-2) + \ldots + 3*1 \\
&= \sum_{j=1}^{n-1} (j+2)j = \sum_{j=1}^{n-1} j^2 + \sum_{j=1}^{n-1} 2j = \frac{(n-1)n(2n-1)}{6} + 2\frac{(n-1)n}{2} \\
&= \frac{n(n-1)(2n+5)}{6} \approx \frac{1}{3}n^3 \in \Theta(n^3).
\end{aligned}
$$

Q.19 . Solve the following recurrence relations.

a. $x(n) = x(n-1) + 5$ for $n > 1$, $x(1) = 0$

b. $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$

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

Solution:

1. a) x(n)=x(n-1)-2=[x(n-2)-2]-2=x(n-2)-2.2
=x(n-3)-2.3
=x(n-i)-2.i
.....
= x(1)-2(n-1)=1-2(n-1)=3-2n

b) x(n)=4x(n-1)
$=4[4x(n-2)]=4^2x(n-2)$
$= 4^3x(n-3)$
......
$=4^ix(n-i)$
=......
$=4^{i-1}x(1)=4^{i-1}*4=4^i$

c) $x(n)=x(n/2)+n^2$
$x(n)=[x(n/4)+(n/2)^2]+n^2$
$= x(n/8)+n^2/16+n^2/4+n^2$
....
$=n^2(1+1/4+1/16+1/64+\ldots\ldots)$
$=n^2((1)((1/4)^{n/4-1}-1))/(1-1/4)=4n^2((1/4)^{n/4-1}-1)/3$

20. Consider the following recursive algorithm.

ALGORITHM *Q(n)*

//Input: A positive integer *n*

if $n = 1$ **return** 1

else return $Q(n-1) + 2 * n - 1$

a. Set up a recurrence relation for this function's values and solve it to determine what this algorithm computes.

b. Set up a recurrence relation for the number of multiplications made by this algorithm and solve it.

c. Set up a recurrence relation for the number of additions/subtractions made by this algorithm and solve it.

Solution:

$Q(n) = Q(n-1) + 2n - 1$ for $n > 1$, $Q(1) = 1$.

Computing the first few terms of the sequence yields the following:

$Q(2) = Q(1) + 2 \cdot 2 - 1 = 1 + 2 \cdot 2 - 1 = 4$;

$Q(3) = Q(2) + 2 \cdot 3 - 1 = 4 + 2 \cdot 3 - 1 = 9$;

$Q(4) = Q(3) + 2 \cdot 4 - 1 = 9 + 2 \cdot 4 - 1 = 16$.

Thus, it appears that Q(n) = n2. We'll check this hypothesis by substituting this formula into the recurrence equation and the initial condition.

The left hand side yields Q(n) = n2. The right hand side yields

$Q(n-1) + 2n - 1 = (n-1)^2 + 2n - 1 = n^2$.

The initial condition is verified immediately: $Q(1) = 1^2 = 1$.

b. $M(n) = M(n-1) + 1$ for $n > 1$, $M(1) = 0$. Solving it by backward substitutions (it's almost identical to the factorial example–see Example

1 in the section) or by applying the formula for the nth term of an arithmetical progression yields $M(n) = n - 1$.
c. Let $C(n)$ be the number of additions and subtractions made by the algorithm. The recurrence for $C(n)$ is $C(n) = C(n - 1) + 3$ for $n > 1$, $C(1) = 0$. Solving it by backward substitutions or by applying the formula for the nth term of an arithmetical progression yields $C(n) = 3(n - 1)$.
Note: If we don't include in the count the subtractions needed to decrease n, the recurrence will be $C(n) = C(n - 1) + 2$ for $n > 1$, $C(1) = 0$.
Its solution is $C(n) = 2(n - 1)$.

21. Consider the following recursive algorithm.

ALGORITHM *Min1(A*[0..*n* − 1]*)*
//Input: An array *A*[0..*n* − 1] of real numbers
if *n* = 1 **return** *A*[0]
else *temp* ←*Min1*(*A*[0..*n* − 2]*)*
if *temp* ≤ *A*[*n* − 1] **return** *temp*
else return *A*[*n* − 1]

a. What does this algorithm compute?
b. Set up a recurrence relation for the algorithm's basic operation count and solve it.

Solution:
a.The algorithm computes the value of the smallest element in a given array.
b. The recurrence for the number of key comparisons is
$C(n) = C(n - 1) + 1$ for $n > 1$, $C(1) = 0$.
Solving it by backward substitutions yields $C(n) = n - 1$.