# CSE 3131: ALGORITHM DESIGN 1

**ASSIGNMENT 1: Introduction to Algorithm Design; Heaps; Graph and Related Algorithms; Greedy Method**

**Submission due date: 15.01.2022(Saturday)**

---

- *Assignment scores/markings depend on neatness and clarity.*
- *Write your answers with enough detail about your approach and concepts used, so that the grader will be able to understand it easily. You should ALWAYS prove the correctness of your algorithms either directly or by referring to a proof in the book.*
- *The marking would be out of 300.*
- *You are allowed to use only those concepts which are covered in the lecture class till date.*
- *Plagiarized assignments will be given a zero mark.*

---

*SECTION - A*

## Method of induction:

1. Prove that: for $n \geq 5$, $4n < 2^n$.

2. Prove that: for all $n > 0$, $n^3 \leq n^2$

## Counter Examples:

A natural place for counter examples to occur is when the converse of a known theorem comes into question. The **converse** of an assertion in the form "If P, Then Q" is the assertion "If Q, Then P". Whereas a complicated proof may be the only way to demonstrate the validity of a particular theorem, a single counter example is all that is need to refute the validity of a proposed theorem. For example, In Calculus you learn that if a function is differentiable at a point, then it is continuous at that point. What would the converse assert? It would say that if a function is continuous at a point, then it is differentiable at that point. But you know this is false. The counter example is f(x) = |x|. This function is continuous at x = 0, but it is not differentiable at x=0. This one counter example is all we need to refute the converse.

1. State the converse of "If a and b are even integers then a+b is an even integer". Show that the converse is not true by producing a counter example.

2. State the converse of "If a, b and c are real numbers such that a + b = c, then $(a+b)^2 = c^2$". Show that the converse is not true by producing a counter example.

3. State the converse of "If a, b and c are integers such that a divides b, then a divides the product bc." Show that the converse is not true by producing a counter example.

4. State the converse of "If a and b are rational numbers, then so is the product ab". Show that the converse is not true by producing a counter example.

## Proofs of Correctness:

1. Consider the following algorithm:

```
Interchange(a,b)
//Input: Nonnegative integers a and b
```

```
//Output: a and b with swapped contents between them
1       a = a + b
2       b = a – b
3       a = a + b
```

Check the correctness of the pseudocode, using an experimental analysis (i.e. using counter examples).

2. Show that a+b < min(a,b). (Hint: use counter-examples)

3. Consider the following algorithm: (the lines starting with "//" are comments that will not be executed).

```
// precondition: $a \in N$, $b \in N$
m := a
// loop invariant: $m \geq 0 \wedge \exists x \in N, a = bx + m$
while $m \geq b$ do
    m := m – b
end while
// postcondition: $m = a \bmod b$, i.e., $0 \leq m < b \wedge \exists x \in N, a = bx + m$.
```

(a) Prove that the loop invariant is true before the first iteration of the loop.

(b) Let m denote the value of m at the end of some iteration of the loop and assume that the loop invariant is true for m . Furthermore, let x denote the value that makes a = bx + m true. Prove that the loop invariant remains true at the end of the next iteration of the loop, if there is such an iteration (use m to denote the value of m at the end of the next iteration).

(c) Prove that the postcondition is true after the last iteration of the loop. (You may assume that the loop invariant is true at the end of every iteration of the loop.)

4. Let A be a 1D array of n natural numbers, and consider the following pseudocode, with the precondition, postcondition and loop invariants inserted as comments:

```
//pre-condition: $\forall k \in N, 0 \leq k < n \rightarrow 0 \leq A[k]$
j =0
// Outer loop invariant: $\forall s \in N, 0 < j \leq s < n \rightarrow 0 \leq A[0] \leq A[1] \leq \ldots \leq A[j-1] \leq A[s]$
// Outer loop description: the first j elements are sorted in ascending order,
// and the jth element is less than or equal to any element between the j+1st and the nth element.
while(j < n){
    $i_{min}$ = j
    k = j +1
    // Inner loop invariant: $\forall r \in N, j \leq r < k \rightarrow A[i_{min}] \leq A[r]$
    // Inner loop desciption: finding smallest element between the j+1st and the nth element.
    while(k < n){
        if (A[k] < A[$i_{min}$]){
            $i_{min}$ = k
        }
        k ++
    } //end of inner loop
    temp = A[j]
    A[j] = A[$i_{min}$]
    A[$i_{min}$] = temp
    j ++
} //end of outer loop
//post-condition: $\forall j \in N, \forall k \in N, 0 \leq j < k < n \rightarrow A[j] \leq A[k]$
```

(a) Prove that the outer loop invariant is true at the start of the first iteration of the outer loop.
(b) Prove that if the outer loop invariant is true at the start of an iteration of the outer loop and the outer loop is executed, then the inner loop invariant is true at the start of the first iteration of the inner loop.

(c) Prove that if the inner loop invariant is true at the start of any iteration of the inner loop, then it is true at the end of that iteration.
(d) Prove that if the outer loop invariant is true when the outer loop terminates, then the post-condition is true.

5. Examine the method mult(m,n) below. Prove or disprove that if the loop invariant is true at the beginning of a loop iteration, then it is true at the end of a loop iteration. Your proof should be in the structured proof format from class.

```
// mult returns the product of integers m and n
// precondition: m and n are integers
// postcondition: integer product mn is returned

int mult(int m, int n) {
        int x = m, y = n, z = 0;
        // loop condition: z = mn - xy
        while (x != 0) {
                if (x % 2 != 0) {
                        if (x * m > 0) {
                                z = z - y;
                        }
                        else {
                                z = z + y;
                        }
                }
                x = (int)(floor(x / -2.0));
                y = y * 2;
        }
        // postcondition z = mn
        return z;
        }
}
```

6. Prove that the following algorithm for exponentiation is correct.

```
        function power(y, z)
        // comment Return yz , where y ∈ R, z ∈ N
        1       x := 1;
        2       while z > 0 do
        3               x := x · y
        4               z := z − 1
        5       return(x)
```

7. Prove that the following algorithm for the multiplication of natural numbers is correct.

```
        function multiply(y, z)
        // comment Return yz, where y, z ∈ IN
        1       x := 0;
        2       while z > 0 do
        3               if z mod 2 = 1 then x := x + y;
        4       y := 2y; z := z/2 ;
        5       return(x)
```

8. Consider the following merge function. Is it correct?

INPUT: A[1..n1], B[1..n2] sorted arrays of integers
OUTPUT: permutation C of A.B s.t. $C[1] \le C[2] \le \ldots \le C[n1+n2]$

```
i = 1
j = 1
for k = 1 to n1 + n2 do
    if A[i] <= B[j] then
        C[k] = A[i]
        i++
    else
        C[k] = B[i]
        j++
return C
```

9. Here is a bunch of functions of one variable, n.
$\sqrt{n}$, $2^n - 20n$, $n^2 - 20n$, $n \lg n$ , $n^2 - n^3 + n^4$ , $n^2 \lg n$, $(\lg n)^2$ , $17n$
Place them in a list from left to right, so that if f (n) is any function in your list and g(n) any function to its right, we have f (n) = O(g(n)). You do not need to justify your answer.
(For instance, if the functions were 5n, 23, $n^2$ , the correct answer would be 23, 5n, $n^2$ , since 23 = O(5n) and 5n = O($n^2$ ).)
(In this course, lg n represents the binary logarithm of n, i.e., $\lg n = \log_2 n$.)

10. (a) Prove that $2^{\lceil \lg n \rceil + \lfloor \lg n \rfloor} = \Theta(n^2)$.
(b) Prove or disprove: $2^{\lfloor \lg n \rfloor} = \Theta(2^{\lceil \lg n \rceil})$.
(c) Prove or disprove: $2^{2\lfloor \lg \lg n \rfloor} = = \Theta( 2^{2\lceil \lg \lg n \rceil})$.
(d) Prove or disprove: If f(n) = O(g(n)), then log(f (n)) = O(log(g(n))).
(e) Prove or disprove: If f(n) = O(g(n)), then 2 f(n) = O(2 g(n) ).
(f) Prove that $\log^k n = o(n^{1/k})$ for any positive integer k.

11. Solve the following recurrences. State tight asymptotic bounds for each function in the form Θ(f(n)) for some recognizable function f(n). You do not need to turn in proofs (in fact, please don't turn in proofs), but you should do them anyway just for practice. Assume reasonable but nontrivial base cases if none are supplied. More exact solutions are better.
(a) $A(n) = A(n/3 + 5 + \lfloor \log n \rfloor) + n \log \log n$
(b) $B(n) = \min_{0<k<n}(3 + B(k) + B(n - k))$ .
(c) $C(n) = 3C(\lceil n/2 \rceil - 5) + n/ \log n$
(d) $D(n) = n/n-5\ D(n - 1) + 1$
(e) $E(n) = E(\lfloor 3n/4 \rfloor) + 1/ \sqrt{n}$
(f) $F(n) = F (\lfloor \log_2 n \rfloor) + \log n$
(g) $G(n) = n + 7 \sqrt{n} \cdot G(\lfloor \sqrt{n} \rfloor)$
(h) $H(n) = \log 2 (H(n - 1)) + 1$
(i) $I(n) = I(\lfloor n^{1/4} \rfloor) + 1$
(j) $J(n) = J(n - \lfloor n/ \log n \rfloor) + 1$

12. Consider the following code fragment.

```
for i =1 to n do
    for j = i to 2*i do
        output "ALGORITHMS"
```

Let T(n) denote the number of times "ALGORITHMS" is printed as a function of n.
a) Express T(n) as a summation (actually two nested summations)
b) Simplify the summation. Show your work.

13. Consider the following algorithm:

```
Sum(n)
// Input: A nonnegative integer n
S ← 0
for i ← 1 to n do
        S ← S + i
return S
```

(a) What does this algorithm compute?
(b) What is its basic operation?
(c) How many times is the operation executed?
(d) What is the efficiency of this algorithm?
(e) Suggest an improved algorithm and indicate its efficiency.

14.

```
function multiply(y, z)
//return the product yz
if z = 0 then
        return(0)
else
        if z is odd then
                return(multiply(2y, z/2 )+y)
        else return(multiply(2y, z/2 ))
```

Claim: For all y, z ≥ 0, multiply(y, z) returns yz. Proof the claim by induction on z ≥ 0.

15.

```
function multiply(y, z)
//Return yz, where y, z ∈ IN
x := 0;
while z > 0 do
        if z is odd then
                x := x + y;
        y := 2y; z := z/2 ;
return(x)
```

Prove the correctness of the pseudocode.

Claim: if y, z ∈ N, then multiply(y, z) returns the value yz. That is, when line 5 is executed, x = yz.

---

## *SECTION – B (Heaps; Graph and Related Algorithms)*

1. (a) Is the array with values <23, 17,14,6,13,10, 1, 5, 7, 12> a max-heap? If not, then build the max heap.

(b) Write the pseudocode to check whether a given set of data elements is representing is a heap or not.

(c) Build the min-heap on the set data elements given in Q1(a) in this section..

2. Illustrate the operation of MAX-HEAPIFY (A,3) on the array A =<27, 17,3,16,13,10,1,5,7,12,4,8,9,0>.

3. Illustrate the operation of HEAPSORT on the array A=<5,13,2,25,7,17,20,8, 4>.

4. What is the running time of HEAPSORT on an array A of length n, that is already sorted in an increasing order. What if the array is sorted in descending order?

5. Implement the following algorithms using the concepts taught in the lecture classes (Write the Java codes):

(a) Breadth-First-Search

(b) Depth-First-Search

(c) Kruskal's MST algorithm

(d) Prim's MST algorithm

(e) Dijkstra's Shortest Path algorithm

(f) Testing Bipartiteness using BFS

(g) Topological Ordering using DFS

(h) Finding the number of connected components using disjoint-set data structure.

---

------------------------------------------------------------------------------------------------------------------------------------------------------------

## *SECTION – C (Greedy Method)*

1. We are given n objects and a knapsack or bag. Object i has a weight $w_i$ and the knapsack has a capacity m. If a fraction $x_i$, $0 <= x_i <= 1$, of object i is placed into the knapsack, then a profit of $p_i x_i$ is earned. The objective is to fill the knapsack completely so as to maximize the total profit earned.
(a) Give a greedy formulation of the above problem statement.
(b) Write the pseudocode for the greedy-based algorithm to solve the above given problem.
(c) Find an optimal solution to the knapsack instance n = 7, m = 15, (p1, p2, p3, p4, p5, p6, p7) = (10, 5, 15, 7, 6, 18, 3) and (w1, w2, w3, w4, w5, w6, w7) = (2, 3, 5, 7, 1, 4, 1).

2. Let G=(V,E) be an undirected graph. A node cover of G is a subset U of the vertex set V such that every edge in E is incident to at least one vertex in U. A minimum node cover is one with the fewest number of vertices . Consider the following greedy algorithm for this problem:

```
1. Algorithm Cover(V,E)
2. {
3.  U := Ø
4.   repeat
5.     {
6.              Let q be a vertex from V of maximum degree;
7.              Add q to U; Eliminate q from V;
8.              E := E – {(x,y) such that x=q or y=q};
9.      } until(E = Ø);//U is the node cover
10. }
```

Does this algorithm always generate a minimum node cover?

3. Given the arrival and departure times of all trains that reach a railway station,which must be scheduled so that no train is kept waiting and as possible as uses minimum number of platforms required. Consider that all the trains arrive on the same day and leave on the same day. Arrival and departure time can never be the same for a train but we can have arrival time of one train equal to departure time of the other. At any given instance of time, same platform can not be used for both departure of a train and arrival of another train.( Note: Time interval is 24 hour format.)

(a) Write the pseudocode for the greedy-based algorithm to solve the above given problem with a goal to minimize the use of number of platforms/resources.
(b) Write FIND_MINIMUM_PLATFORM( ) function which returns minimum number of platform used to schedule all train in a day where number of train is equal to “n” , Arrival time and departure time of train as input to this function.
(c) Write greedy problem statement with greedy choice property and constraints of above problem.
(d) Calculate the time and space complexity of above (a) greedy algorithm .

---

**Submission and Grading:**

Submit the hard copy of your assignment by the due date, i.e. 15.01.2022(Saturday).
Part of your assignment grade comes from its "external correctness." This is based on correct output on various sample inputs .

The rest of your assignment's score comes from "internal correctness." Internal correctness includes:
1. Use of methods to minimize the number of steps.
2. Appropriate use of rules, axioms, and suitable diagrams to enhance readability of your responses.

Send a zip folder (name of the zip folder must be your registration number_AD1) containing the code and output file/screen-shot of each program implementation mentioned to the official email id of your AD1 class teacher. On the top of each program, you must mention your full name, registration number, title of the program and date.