MCAC 201 : Design and Analysis of Algorithms
Minor
March 2025

Problem 5-1 (Mathematical Induction) 10 points

(a) (5 points) Prove that every amount of postage that is at least *12 cents* can be made from *4-cent* and *5-cent* stamps.

(b) (5 points) Use induction to prove the statement $S(n) : 2 + 3 \cdot 2 + 4 \cdot 3^2$ upto n terms $= n \cdot 2^n$

1. The n^{th} term is _____.
2. Base Case: For $n =$ _____, LHS = _____, RHS =_____.
3. Induction Hypothesis: Suppose the state is true for $n = k$. i.e.
   _____
   (give the above statement mathematically)
4. We will prove that the statement is true for $n = k + 1$. i.e.
   _____
   (give the above statement mathematically)
5. Proof: (give the proof mathematically)
   $LHSofS(k+1) =$
   _____
   $= LHSofS(k) +$ _____
   $=$ _____ $+$ _____ (apply induction)
   $=$ _____ $+$ _____ (simplify the expression)
   $= RHSofS(k+1)$. Hence Proved.

Problem 5-2 (Sorting/Searching) 8 points

(a) (4 points) Given a sorted array $A[1, \ldots, n]$ of n distinct integers, give an $O(lg(n))$ time algorithm that finds out if $\exists k$ such that $A[k] = k$. Fill in the blanks in the following informal idea of an algorithm that returns such an index k if it exists, else returns -1.

1. Compute the middle index "mid"
2. _____
3. If $A[mid] < mid$ then recurse on _____
4. else recurse on _____

Also, give one line justification.

(b) (4 points) For each of the following inputs, state whether it is worst case for Insertion sort or not? In either case, count the number of comparisons. Also, if it is not the worst case, modify it by changing the positions of at most two values - that means you can only swap one pair of numbers.

i. 27, 32, 10, 15, 3, 6, 1, 2

ii. 32, 37, 12, 38, 8, 10, 5, 6

Problem 5-3 (Graphs) 13 points

(a) (4 points) Given an undirected graph G, modify the following code "by adding a line or two" to determine if the graph contains an odd length cycle.

```
BFS(G, s)
    For j=1..n { color[j] = white;  level[j] = ∞; parent[j] = null; } //Initializa
    Enqueue(Q, s)
    color[s] = grey;  // Vertex added in the queue
    level[s] = 0;
    Q = ∅;
    While Q ≠ ∅
        i = Dequeue(Q);
        process(i); //for eg. update the weather profile of city i
        color[i] = black;  //i has been visited/processed
        for each neighbour j of i  //explore each neighbor of i
            if color[j] == white  //If not added to queue earlier
                Enqueue(Q, j)
                color[j] = grey;  // Vertex added in the queue
                level[j] = level[i] + 1;
                parent[j] = i;
```

(b) (4 points) Given an undirected graph G with vertices already labeled with the DFS-start and DFS-finish times. Give a linear time algorithm to sort the vertices on their DFS-finish times.

(Hint: Think more - write answer in one or two lines)

(c) (5 points) Let $G = (V, E)$ be a directed acyclic graph. Suppose each vertex v has weight $Wt[v]$ stored in array $Wt[1:n]$. Give a linear time algorithm to compute for each vertex v the maximum of the weights of the vertices reachable from v, storing the results in $ReachWt[1:n]$.

In a DAG, if there is an edge (v, w), v can reach every vertex that w can. Can we modify DFS-Visit to find all the $ReachWt[v]$ values?

procedure DFS-VISIT(v)

```
    color[v] ← gray
    ???
    for each neighbor w of v do
        if color[w] = white then
            DFS-Visit(w)
        end if
        ???
    end for
    color[v] ← black
end procedure
```

Problem 5-4 (Greedy) — 9 points

(a) (5 points) Consider a set of coins with different denominations in decreasing order say $d_1 > d_2 \ldots d_n$, each with infinite supply and a value v, we want to make change of v with the given coins such that minimum number of coins are used. Consider the following greedy algorithm:

1. Pick as many coins of the largest denomination d_1 such that the total is less than v. Let $res = v -$ total of d_1's
2. remove d_1 from the set of denominations and repeat on the value res until done.

Give a counter example to show that the above algorithm is not guaranteed to give an optimal solution.

(b) (4 points) For the interval partitioning problem, suppose the sorted sequence of starting times and finishing times - all taken together, is given to us. For example for $(1, 10), (4, 15,$ we are given the sequence $1, 4, 10, 15$.

Give a linear time algorithm to determine the optimal number of colors required to color them in a non-conflicting manner.

Give one line informal argument that your algorithm is correct. (don't worry if you are not able to write)