Unique Paper Code : 32347506
Name of the Course : B. Sc. (Hons.) Computer Science (CBCS)
Name of the Paper : Advanced Data Structures and Algorithms
Semester : V
Duration : 3 Hours
Maximum Marks : 75
Year of Admission : (Admission of 2015, 2016, 2017 & 2018)

Instructions for candidates:

Attempt any four questions.
All questions carry equal marks.

Q1.

a) List the trade off between choosing a binary heap vs a Fibonacci heap to implement priority queue.

b) Suppose we implement increase_key function in a min heap using Fibonacci heap as follows:

increase_key(node, new_key)

1. Increase the *key* of *node* to *new_key*
2. Concatenate the *child-list* of *node* to *root-list*, update the parent of all children to NULL. Erase the *mark* attribute (if present) of child nodes.
3. Cut the *node* from its parent *z* and insert it to root list. Apply cascading cut to *z* similar to the *decrease_key()* function of Fibonacci heaps. Reset (erase) the *mark* attribute on *node*.

i) Is this algorithm correct?

ii) What is the amortized cost of this algorithm?

iii) If we add an additional step of consolidating the root list after step 3, would the amortized cost change, explain?

Q2.

a) Suppose a document is composed of six unique characters A-F. Probabilities of occurrence of the characters are given as:

A: 0.45, B: 0.12, C: 0.09, D: 0.13, E:0.16, F=0.05

Compression ratio of a document is defined as the ratio of average code length in Huffman code to average code length in fixed length code. What is the compression ratio achieved by Huffman algorithm for this document?

b) Let Z be an alphabet. We associate a frequency with each character of Z. Let $x, y \in Z$ be two characters having the least frequencies. Show that there exists an optimal prefix code where the codewords of x and y have same length and differ only in their rightmost bit.

Q3.

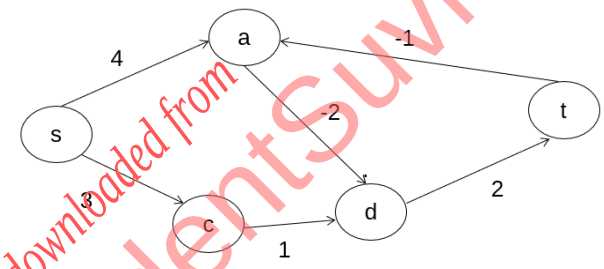
a) Due to ongoing pandemic of covid19, a student will be given a set of n assignments by her teachers through online mode. The student has the following information in advance for each assignment:

1. Starting time- the time when a teacher will upload an assignment.
2. Ending time- the submission deadline by which student needs to upload her solution.
3. Marks- the marks associated with the assignment.

The duration between the starting and the ending time of an assignment is just sufficient time to work on that assignment alone. Therefore if the student chooses to begin working on one assignment, she can not work on another assignment simultaneously. Moreover partial submissions will fetch zero marks, so she always chooses an assignment that can be completed. The good news is that for every complete submission, she always gets full marks. She has to select a subset of assignments to work on so as to maximize her total marks. A dynamic programming solution to this problem first sorts the assignments in the increasing order of their starting times. Let *Opt[i]* denote the optimal marks obtained by considering the first i assignments in the sorted list of assignments.

i) Give a recurrence for *Opt[i]* when i^{th} assignment is included in the selection.
ii) Give a recurrence for *Opt[i]* when i^{th} assignment is not included in the selection.
iii) Give the final recurrence for *Opt[i]*.
iv) Give the base cases for the recurrence.
v) Derive the time complexity of the approach and explain.

b) Using a dynamic programming approach, identify if there is any negative cycle in the following graph. Show all steps.



Q4.

a) Given a min heap of n positive integers. Consider the following strategies to find largest integer in this heap.

Strategy 1: Treating the heap as an unordered array, do a sequential search for the largest integer.

Strategy 2: Now consider the following divide and conquer strategy:

```
find_max(root):
    if root == NULL:
        return -1
    l=find_max(root.left)
    r=find_max(root.right)
    return(max(l, r, root.key))
```

i. What is the time complexity of strategy 1?
ii. Is the strategy 2 correct? Give the recurrence for running time and the solution of recurrence.
iii. Give an algorithm which makes no more than $n/2$ comparisons for finding the maximum element in the min-heap? Justify the time complexity of the approach.

b) Write the recurrence relation for time complexity of Karatsuba's algorithm of fast integer multiplication for multiplying two large integers in $O(n^{1.59})$ time. Using your recurrence compute the number of single digit multiplications for the product 1552×2374. Compare this to the number of single digit multiplications required in basic multiplication method.

Q5.
a) A *k*-coloring of a graph $G = (V,E)$ is a function $f: V \rightarrow \{1, 2, \ldots k\}$ so that for every edge (u, v), we have $f(u) \neq f(v)$. The available colors here are named *1, 2, ... k*, and the function *f* represents our choice of a color (or a label) for each node. The *k-coloring* problem is defined as follows:

"Given a graph G and a bound k, does G have a k-coloring?"

i. What is the lower bound on *k* for a complete graph having *n* vertices?
ii. What is the lower bound on *k* for a bipartite graph having *n* vertices?
iii. Show that *3-coloring* problem is in NP.
iv. Show that *2-coloring* problem is in P.

b) Ankit and Neha have been debating about two optimization problems *X* and *Y*. Both *X* and *Y* are known to be in NP. Ankit shows a polynomial time reduction from the set-packing problem to *X*, and Neha shows a polynomial time reduction from *Y* to vertex cover.

Ankit infers: *X* is NP complete while nothing can be said whether *Y* is NP complete or not.
Neha infers: *Y* is NP complete while nothing can be said whether *X* is NP complete or not.

i. Determine whether Ankit/Neha is/are correct or not. Justify.
ii. Is $Y \leq_P X$? Justify your claim.
iii. If we assume that class P = class NP, then which of the following classes do each of *X* and *Y* belong to: P, NP, NP complete? Justify briefly.

Q6.
Consider the problem of finding the maximum flow in a network graph $G = (V, E)$, with capacity c_e on each edge *e*, source vertex *s* and sink vertex *t*. Let *f* be the flow through the graph at $i\text{-}1^{th}$ iteration of Ford-Fulkerson's max flow algorithm. Let *f'* be the flow in i^{th} iteration. Given that initially, the flow is zero which is a valid flow in terms of both capacity and conservation constraints, argue along the following points regarding validity of flow at each iteration. Let the *s-t* path selected in i^{th} iteration be P.
a) Show that in following two cases for an arbitrary edge e = (u,v) on path P, the capacity constraint is satisfied i.e. $0 \leq f'(e) \leq c_e$:

i) Argue that if $e = (u, v)$ is a forward edge, the capacity constraint is satisfied.
ii) Argue that if $e = (u, v)$ is a backward edge on path P, the capacity constraint is satisfied.

b) Let *v* be a node on P, argue that in each of following four cases, the conservation constraint is satisfied on v, i.e., $f_{in}(v) = f_{out}(v)$.

i) Show that if both *e* and e_0 are forward edges, the conservation constraint is satisfied at *v*.
ii) Argue that if both *e* and e_0 are backward edges, the conservation constraint is satisfied at *v*.

iii) Show that if e is a forward edge and e_0 is a backward edge, the conservation constraint is satisfied at v.
iv) Argue that if e is a backward edge and e_0 is a forward edge, the conservation constraint is satisfied at v.