BITS, Pilani - K K Birla Goa Campus

Year 2010-11 - Semester I

Course : CS C363/IS C363 Data Structures and Algorithms **Test :** 1

Marks : 50 **Weightage :** 50/300 **Mode :** Closed Book

Date : 18/09/2010 **Time :** 1200-1300 hrs **Duration :** 60 min

---

Instructions : Do not use or activate mobile phones. Write relevant answers.

Q1. Give an efficient algorithm to count the number of inversions (pairs $A_i > A_{i+j}, 0 \leq i \leq n-1, 0 < j \leq n-i-1$) in an array A. Choose the most suitable one of the standard known sorting algorithms; **just indicate any modification instead of a full presentation.** **[10]**

Q2.

i. In deterministic quicksort, in the partition function, fix the pivot at $\lfloor (l+r)/2 \rfloor$ for input (A, l, r). What is the running time of this quicksort for an already sorted (forward) array? Which case (array) is the worst case for this algorithm? **[6]**

ii. Suppose here the pivot is instead fixed at some $i, l < i < r$. What will the expressions for time complexity be in the worst, best, average cases? Will they depend upon i, or $i/(r-l+1)$, or i/n, or none? **[6]**

Q3. Insertion sort has the best case when the array is already forward sorted, in which case it takes $O(n)$ time. If the array is reverse sorted, again exchanging between pairs at $(0, n-1), (1, n-2), \ldots, (i, n-i-1), \ldots, (\lfloor n/2 \rfloor - 1, \lfloor n/2 \rfloor + 1)$ sorts it forward in $O(n)$ time. Let us suppose we combine these two techniques and insert them into quicksort such that, first detecting the case (sorted, forward or reverse or none), we choose one of these techniques, or else proceed with the normal quicksort. Thus, the worst case of quicksort may be avoided. Now, answer the following. **[4x3=12]**

i. What is the (minimum) overhead of the check?

ii. Where will you insert the check and the choice (recursive quicksort or partition)? Why?

iii. What is the effect on the worst, best, average case complexities because of the added overhead?

Q4. The recurrence for the run-time of a divide and conquer algorithm is $T(n) = 4T(n/2) + \Theta(n^2)$. Your teacher suggests that a certain change in the algorithm changes it to $T(n) = 9T(n/3) + O(n^2)$. Will you opt for the change? Why? **[4+4+2=10]**

Q5. Choose an efficient sorting algorithm for each of these applications: sorting full names; sorting posted letters on pin-code to different mailbags to be transported via different routes; sorting lunchboxes by the dabbawallahs in Mumbai. **[6]**

END.

BITS, Pilani - K K Birla Goa Campus

Year 2010-11 - Semester I

Course : CS C363/IS C363 Data Structures and Algorithms Test : 2

Marks : 50 Weightage : 50/300 Mode : Open Book

Date : 30/10/2010 Time : 1200-1300 hrs Duration : 60 min

Instructions : Do not use or activate mobile phones, Internet connections or *ANY* communication devices. Any such attempt itself will be regarded as *malpractice.*

Use any of the standard data structures, their methods and algorithms, and theoretical analyses so far discussed in the course. You may just name them wherever needed without detailing out. For writing new data structures and algorithms, you may use pseudo-code or C or Java or Python; but omit the unimportant common code/text. Time needed is proportional to the marks. Write only as much as required. *Lengthy answers, if fake and irrelevant, may be penalized.*

Q1. In our DSA class, every student has some knack for solving some tough problems. The problems each student solves easily is recorded in a score-sheet which contains the id-number, the number of problems, and the list of problems, for each student. We are enlising students from our class to lead Quark'11 events. A student can lead a group if she can solve all the problems that each in her group can solve, and more. Whether one student can lead another or not will be known either by directly comparing the problems each solves using the score-sheet, or else by transitive inference on students' sequences: if A can lead B and B can lead C then A can lead C. Design a suitable space-efficient data structure that represents the "A can lead B" relationships, and give an appropriate algorithm that fills in the data structure with the least possible accesses to the score-sheet. Assume that there is a function `CanLead(record,record)` that compares two student records in the score-sheet and returns 1 if the problem-list of the first includes that of the second (so the first can lead the second), or 0 if none of their lists includes the other's (none can lead the other), or -1 if the second's includes the first's (the second can lead the first), respectively. The score-sheet is sorted on id-numbers. You are free to preprocess the score-sheet, but count those operations also while measuring efficiency. **[14]**

Q2. State if true or false, with reason. Marks are for the reason *only*. **[8]**

i. Given an array of integers about which nothing is known except the size n, it takes no more than $O(n)$ time in the worst case to place it in a BST.

ii. The number of tree node assignments needed to perform a delete operation and its subsequent restructuring in an AVL tree is $O(log_2 n)$ whereas it is $O(1)$ for an insert operation and subsequent restructuring.

Q3. Choose one abstract data structure each for each of the following applications. The criterion is that it must give the most run-time efficient search, insert, and delete operations. (Effectively you are asked to choose exactly **4** data structures, but they need not be distinct. Justify only if you are unsure. Inconsistencies like naming bucket sort and justifying radix sort may result in negative marks.) **[4x4=16]**

i. Storing Indian income-tax PAN id numbers along with photographs of id holders for a website that lets security agencies verify identities by the PAN id number instead of a card.

ii. Storing key and index terms with links to research articles and books in online search databases for research material.

iii. Representing the reporting and controlling structure of personnel in an army.

iv. Representing the sharing of hostel wings, common campus clubs, and common electives between pairs of students of a class.

Q4. You are storing id-numbers of 162 students in our class obtaining n marks out of 300 in DSA for each $n = 0, 1, 2, ..., 300$ in a hash table. Devise the simplest and the least collision-prone hash-function for this such that wasted memory cells also are minimum. Here collision is when two students get marks n_1 and n_2, $n_1 \neq n_2$, but get the same slot in the table. What will be the number of slots in (i.e. the length of) the hash table? Which collision-handling scheme will you use? If your answer is open addressing, fix the probing function (linear, quadratic, etc.) too. You may use a different collision handling scheme or a modified hash table structure than any listed and explained in the books. Justify all your answers. **[4+4+4=12]**

END.

BITS, Pilani - K K Birla Goa Campus

Year 2010-11 - Semester I

Course : CS C363/IS C363 Data Structures and Algorithms Test : Comprehensive

Part A Marks : 50 Weightage : 50/300 Mode : Closed Book

Date : 13/12/2010 Time : 0900-1000 hrs Duration : 60 min

Write answers in the space given on the same question paper itself

Q1. Give compact expressions for the following: **[15]**

i. The number of distinct binary search trees of given n keys. (Give only a recurrence, you need **not** solve it.)

ii. The number of distinct AVL trees of given 7 keys.

iii. The number of distinct heaps (complete binary trees with the heap-order property) of given 7 keys.

Q2. Encircle all the correct choices for each (more than one can be correct). Overwriting is forfeiting all claims on rechecks: **[4x4=16]**

i. Run times of dynamic programming and memoization algorithms for the same problem will be
(a) of different orders (b) the same (c) of the same order but with different constants
(d) unrelated

ii. If the median of an array can be found in linear time then the order of run-time of quicksort using the median as the pivot will
(a) remain unchanged (b) be lower only in the average case (c) be lower in the worst case
(d) be higher

iii. Which of the following algorithms are suitable for external (large hard disk files) sorting?
(a) mergesort (b) quicksort (c) heapsort (d) bucket sort

iv. Does an acyclic graph have a path between every pair of nodes? (a) Always (b) Never (c) Sometimes (d) Only in the empty graph

RTO

Q3. Choose an efficient algorithm for each of the following applications (name the algorithms and also their parameters): **[10]**

i. Library book shelves sorted manually on the subject. The call number visible on the spine is in a "num.num/auth" format, the num.num part giving the subject classification, and the auth part the author names abbreviated. (e.g. Our course text-book by Goodrich and Tamassia is numbered 005.1/GOO.TAM).

ii. Ranking students on total marks obtained in all evaluation components, with the ties broken by compre marks, assignment marks, test1+2 marks, roll number in that order of significance.

iii. Deciding job schedules between n jobs taking $t(i)$ time and giving $p(i)$ profit for $i = 1, \ldots, n$ to maximize profit in minimum time.

iv. Identifying the right sequence of officials to talk to about your application for the govt. UID card depending upon the chain of command that affects whether your application will be granted or not, with the least possible rounds of the office.

v. Fixing locations in the Audi for Wi-Fi transponders that work with line-of-sight radiation, so as to maximize connectivity depending upon the capacity and geometry of the Audi and transponder power.

Q4. Let $f(x)$ and $g(x)$ be monotonically increasing functions for $x > 1$ (i.e. $\forall x > 1, y > x \Leftrightarrow f(y) > f(x)$), and let $f(10) = 100$, $f(100) = 1000$, $g(10) = 10$, $g(100) = 10000$. Is this information enough to detect the order relation (in big-O notation) between f and g? If yes, give the order relation; if not, give two examples of f, g pairs to demonstrate that both $f(n) = O(g(n))$ and $g(n) = O(f(n))$ allow the given values even when $f(n) \neq \Theta(g(n))$. **[9]**

BITS, Pilani - K K Birla Goa Campus

Year 2010-11 - Semester I

Course : CS C363/IS C363 Data Structures and Algorithms **Test :** Comprehensive

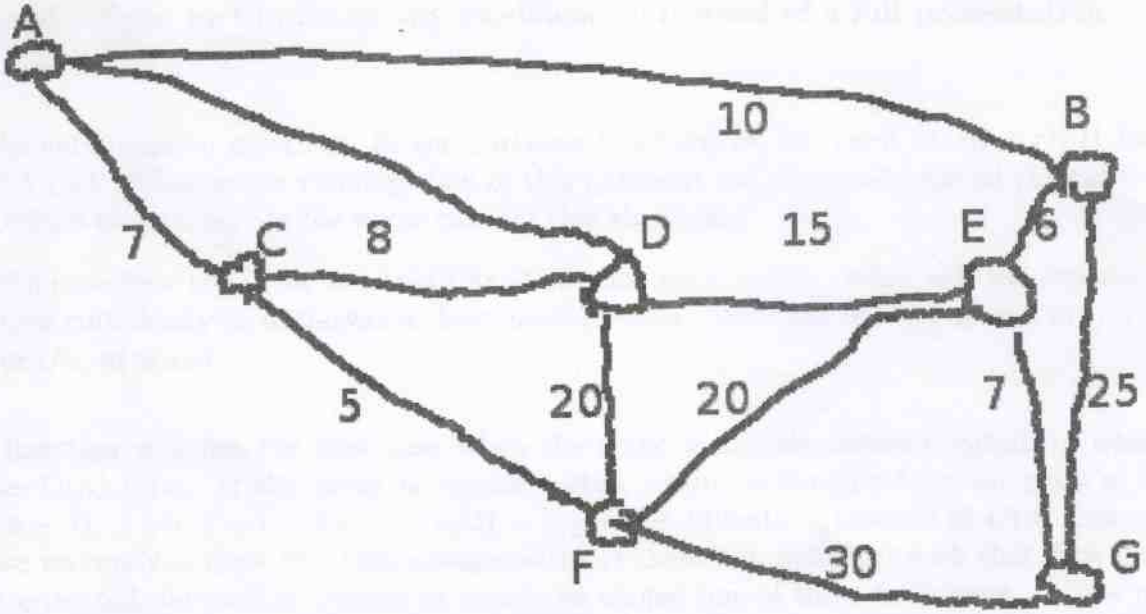
Part B Marks : 50 **Weightage :** 50/300 **Mode :** Closed Book

Date : 13/12/2010 **Time :** 1000-1200 hrs **Duration :** 120 min

---

Instructions : Rough work must be done separately, *not* on the same page as your answers. It is examiners' discretion to interpret untidy, unclear answers, no rechecks on them will be entertained.

Q5. Show the $D^{(0)}$ to $D^{(2)}$ and $\Pi^{(0)}$ to $\Pi^{(2)}$ tables computed by the Floyd-Warshall Algorithm for all-source shortest paths for the following graph. **[25]**



Q6. Give an algorithm (or a C/C++/Java/Python program - precise and concise) to compute binomial coefficients by memoization. **[15]**

Q7. State yes or no with reasons (**not** more than one sentence). **[10]**

i. Is a BST a priority queue?

ii. If an application generates data that are distributed normally (by the Gaussian distribution: the bell-shaped curve) over the range, is it advisable to use a hash table to store the data for fast storage and fast access?

iii. Will overstepping of array indices beyond the array size during access always result in a segmentation fault (on a reliable true POSIX system like GNU/Linux)?

iv. Is it possible to execute a program in the gdb's shell uninterrupted until a particular line in the program changes the value contained in a particular variable?

v. Is it always advisable to implement divide-and-conquer using recursion?