Introduction to Algorithms: 50.004
Singapore University of Technology and Design
14 Dec 2017 9:00am-12:00pm

Final Exam

i. This is an open book and open notes exam.
ii. No computer, smartphone and Internet access.
iii. Write your answer on the answer booklet. You may request for additional answer booklet if necessary.

Part 1 : Asymptotic Notation, Complexity, Master theorem, Sorting (20 points)

Problem 1: Find the asymptotic complexity (6 points)

Q1. (3p) $T(n) = T(\frac{n}{2} + \sqrt{n}) + \sqrt{6046}$

Q2. (3p) $T(n) = T(\sqrt{n}) + \Theta(lglgn)$

Problem 2: True of False (6 points)

Q1. (2p) There exists a comparison sort of 5 numbers that uses at most 6 comparisons in the worst case.

Q2. (2p) Heapsort can be used as the auxiliary sorting routine in the radix sort, because it operates in place.

Q3. (2p) Suppose that an array contains n numbers, each of which is -1, 0, or 1. Then, the array can be sorted in ***O(n)*** time in the worse case.

Problem 3: Insertion sort on small arrays in merge sort (8 points)
Although merge sort runs in ***θ(nlgn)*** worst-case time, the constant factors in insertion sort can make it faster in practice for small problem sizes on many machines. Thus, it makes sense to **coarsen** the leaves of the recursion by using insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which ***n/k*** sublists of length ***k*** are sorted using insertion sort and then merged using the standard merging mechanism, where ***k*** is a value to be determined.

Q1. (2p) Show that insertion sort can sort the ***n/k*** sublists, each of length ***k***, in ***θ(nk)*** worst-case time.

Q2. (3p) Is it possible to merge the sublists in ***θ(nlg(n/k))*** worst-case time? Why?

Q3. (3p) Given that the modified algorithm runs in **θ(nk+nlg(n/k))** worst-case time, what is the largest value of ***k*** as a function of ***n*** for which the modified algorithm has the same running time as standard merge sort, in terms of ***θ*−notation.**

Part 2: Heap, BST, AVL, Linear time sorting (20 points)

Problem 1: Heap (8 points)

1. **(2p)** Given a binary max heap containing n elements, the smallest element can be found in
   A. $O(1)$. $O(\log n)$ C. $O(n)$ D. $O(n \log n)$

2. **(6p)** For each of the following max-heap operation, fill in the array to show the state of the max-heap after the operation. Draw the heap trees to show working and get partial credits.

   a. Insert (26)

   Before

0	1	2	3	4	5	6
24	10	15	8	2	7	3

   After

0	1	2	3	4	5	6	7

   b. ExtractMax()

   Before

0	1	2	3	4	5	6
32	15	7	6	2	4	3

   After

0	1	2	3	4	5

Problem 2: BST (3 points)

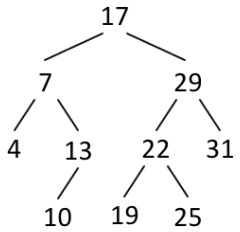
(3p) Recall that in a binary tree, a node may have 0,1, or 2 children.
In the following questions about binary trees, the height of a tree is the length (number of edges) of the longest path. A tree consisting of just 1 node has height 0.

(a) What is the minimum number of nodes in a binary tree of height h?
(b) What is the minimum height of a binary tree containing n nodes?
(c) What is the maximum height of a binary tree containing n nodes?

Problem 3: AVL (6 points)

1. **(2p)** What is the smallest number of nodes an AVL tree of height 4 can store? A tree consisting of just 1 node has height 0.

2. **(4p)** Draw the result when 24 is added to the AVL tree on the right.
   Show your work, in particular, identify clearly any rotation performed.
   For instance, specify "left-rotate(x)" or "right-rotate(x)" for each rotation performed, where x is the value of the node to be rotated.



Problem 4: Sorting (3 points)

(3p) True or False. There exists a comparison-based sorting algorithm that can sort any 6-element array using at most 9 comparisons. Justify your answer.

Part 3: Hashing (20 points)

Use hashing to solve these problems.

1. **(7p)** For an array A of t integers, assume that elements {p, q, r, s} ∈ A. Can you find quadruplets such that a + b + c + d = *Val* , where *Val* is a given target value? Find all unique quadruplets, given that
   • Elements in a quadruplet must be in non-descending order. (ie, a ≤ b ≤ c ≤ d)
   • Your solution set must not contain duplicate quadruplets.

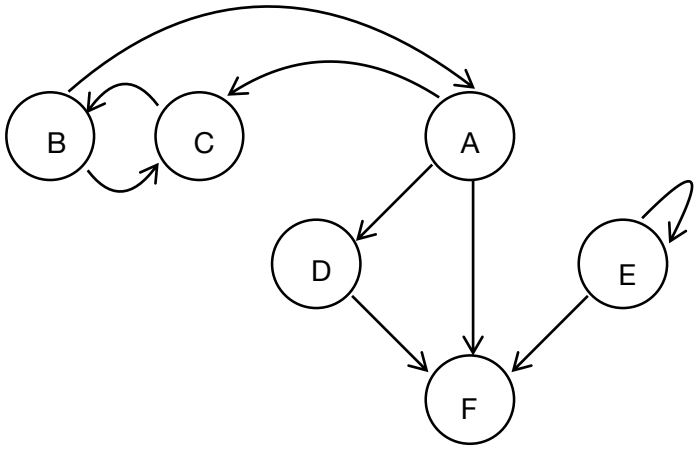
2. **(6p)** Use the following hash function to answer subsequent questions:
   int hash(int key)
   {int result;
   result = key %2;
   return result; }
   (a) Insert the following keys into a chained hashmap. Show all of your work for each insertion.
   5, 4, 2, 1, 3, 7, 8.

   (b) Is the hash() function, as designed here, perfect? Why or why not?

3. **(7p)** URL shortening websites like bit.ly work much like a hashtable where a cryptic short string (e.g., the XYZ in *http://bit.ly/XYZ*) is the key, and a long URL is the value. You decide to start a company for yet another URL shortening service and call it *sutdalgo.ly.* you decide to maintain the maps from short URLs to long ones entirely in an in-memory hashtable.

   You need a good hash function for your custom-built hashtable data structure. It's known that every character has a numerical value dictated by the ASCII encoding scheme (for example A = 65), so he suggests using the sum of the ASCII values of the characters in short URL as the hash key. Is that a good idea? Explain briefly.

Part 4: Graphs, BFS, DFS (20 points)



NOTE: For all the below questions, in case of choice please follow alphabetical order.

1. Consider the above graph G = (V, E), in which V are the set of vertices, and E are the set of edges of the graph. Please answer the below questions.

   a. **(1p)** Enumerate the set of vertices V in G?
   b. **(2p)** Write the adjacency-list representation of the graph G.
   c. **(2p)** Write the adjacency-matrix representation of the graph G.
   d. **(2p)** The graph G is a directed graph. If there would be no arrows in the lines that connect the nodes, and no self-connections, it would be an undirected graph. Write the adjacency-list representation for the resulting undirected graph if there were no arrows in the lines and no self-connecting lines between the nodes in the graph G.
   e. **(4p)** Give the sequence of vertices visited with BFS traversal of the directed graph G = (V, E) starting at node A, and indicate the parent of each visited vertex.
   f. **(4p)** Give the sequence of vertices visited with DFS traversal of the directed graph G = (V, E), and enumerate the back edges and the forward edges.
   g. **(2p)** Indicate the length of the shortest path from A to F.

2. Chose the correct answer. There is only one correct answer.

   a. **(2p)** The sum of all the elements of the adjacent list of an undirected graph G = (V, E) is: *(Note: |E| indicates the number of edges in the the graph G, and |V| indicates the number of vertices in the graph G.)*

   (a) |E| (b) 2|E| (c) |E|+|V| (d) 2|V|

   b. **(1p)** Given a graph G = (V, E), the shortest path from one vertex of the graph to another vertex:
   (a) can be computed with BFS
   (b) can be computed with DFS
   (c) can be computed with BFS and DFS
   (d) cannot be computed with neither BFS nor DFS

Part 5: Shortest Path (20 points)

1. **(1p)** Relaxing a directed edge in a graph will *always* lead to a reduction in the shortest path estimate of the terminating vertex of the edge. Is this a true statement?

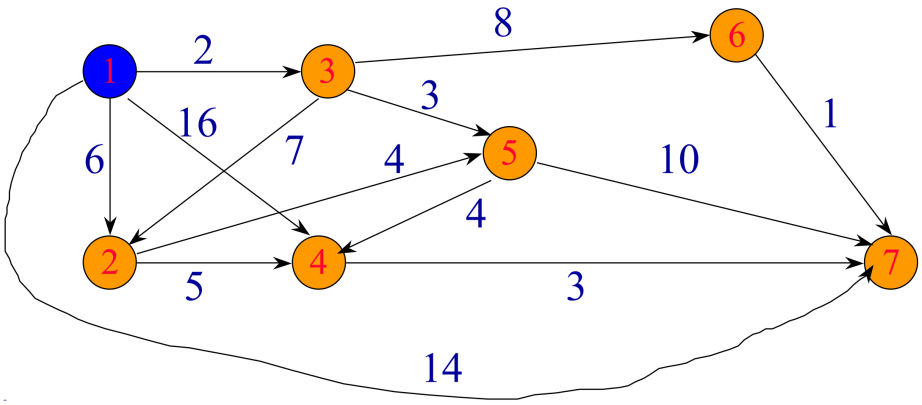
2. **(1p)** Give an example of a particular data structure whose use can significantly improve the time complexity of an algorithm, among the single source shortest path algorithms covered in class. You need to mention the names of the algorithm whose time complexity improves and the data structure whose use causes the improvement.

3. **(1p)** In some specific situations, can we execute the main loop of the Bellman-Ford algorithm fewer than |V|-1 times and still get the correct result? Justify your answer.

4. **(2p)** In the Bellman-Ford algorithm, if we move the mechanism for detecting negative weight cycles from its current position to a position *before* the execution of the main loop, will the mechanism still be able to detect negative weight cycles? Justify your answer.

5. **(3p)** Customize the Bellman-Ford algorithm to be able to detect positive weight cycles. Describe the customized algorithm in pseudocode and derive its time complexity.

6. **(5p)** With reference to the graph below, find the shortest paths from vertex number 1 all other vertices using a *greedy* algorithm. To get points, you need to name the algorithm you are using, and demonstrate that you indeed use a greedy algorithm, by showing the steps in your calculation.



7. Your friend from another pillar tells you about an money-making scheme. It seems, if you start with x units of country A's currency, and convert that to country B's currency, and then to country C's currency, and so on, and finally back to country A's currency, you can end up with mx units of country A's currency, where m is a positive integer greater than 1, so that you make a profit of $mx-x = (m-1)x$ units.

   This sounds pretty exciting, but since you are such a diligent student of this 50.004 course, you want to try things out algorithmically, before committing your money!

   a. You are given n currencies $c_1, c_2, \ldots, c_n$ and an n x n table R of exchange rates, where one unit of currency c_i can be converted to $R[i,j]$ units of currency c_j. You can assume that every currency can be converted to all other currencies.
      i. **(1p)** Represent this set-up as a graph.
      ii. **(1p)** How many edges will the graph have in terms of n?

   b. **(5p)** On the basis of your representation in part (a)design an efficient algorithm to determine whether or not there exists a sequence of currency conversions that will yield a profit; you need to describe the algorithm in pseudocode (4 points) and derive the time complexity of the algorithm (1 point).

Part 6: Dynamic Programming, P & NP (20 points)

1. **(9p)** Propose a dynamic programming solution for this problem. You can answer in any format (such as pseudocode, text description, drawing with example… etc.)

Given an input string and a dictionary of words, find out if the input string can be segmented into a space-separated sequence of dictionary words.

e.g.
Dictionary:{I, like, apple, mango, orange, egg}

Input String: ilikemango
answer: can

Input String: ilikebanana
answer: cannot

2. **(9p)** Propose a dynamic programming solution for this problem. You can answer in any format (such as pseudocode, text description, drawing with example… etc.)

Given a rope of length n meters, cut the rope in different parts of integer lengths in a way that maximizes product of lengths of all parts. You must make at least one cut. Assume that the length of rope is more than 2 meters.

e.g.
Input: n = 2
Output: 1 (Maximum obtainable product is 1*1)

Input: n = 3
Output: 2 (Maximum obtainable product is 1*2)

Input: n = 4
Output: 4 (Maximum obtainable product is 2*2)

Input: n = 5
Output: 6 (Maximum obtainable product is 2*3)

Input: n = 10
Output: 36 (Maximum obtainable product is 3*3*4)

3. **(2p)** If someone is able to proof that P = NP, what will happen to cryptography system? Explain briefly.