CS31005 Algorithms – II

Class Test 3

Date: 18-Nov-2020 Maximum marks: 50

Instructions

- **Answer all of Q1–Q4, and only one of Q5 and Q6.** Be brief and precise.
- If you use any algorithm/result/formula covered in the lectures/tutorials, just mention it, do not elaborate.

---

1. Consider the following instance of the set-cover problem, where X is the universal set, and S_i are the subsets of X.

$$X = \{1,2,3,4,5,6,7,8,9,10\},$$

$$\begin{aligned} S_1 &= \{3,8,9\},\\ S_2 &= \{1,4,8,10\},\\ S_3 &= \{2,8,9,10\},\\ S_4 &= \{5,6,10\},\\ S_5 &= \{3,7,9\},\\ S_6 &= \{1,3,5,7,9\},\\ S_7 &= \{2,4,5\},\\ S_8 &= \{1,2,4,6\}. \end{aligned}$$

You run the greedy set-cover algorithm (as taught in the class) on this instance. Prove/Disprove: The algorithm produces an optimal solution for this instance. **(8)**

2. Consider a Bloom Filter with 15 bits for storing positive integers, with all bits set to 0 initially. The filter uses the following three hash functions to hash an integer key x: $h_1(x) = x \bmod m$, $h_2(x) = (4x+3) \bmod m$, and $h_3(x) = (5x+4) \bmod m$, where m is the size of the filter.

(a) Choose four integers as follows: one integer randomly from 10 to 25, one integer randomly from 26 to 40, one integer randomly from 41 to 55, and one integer randomly from 56 to 75. Make sure that the four integers you choose are distinct modulo 15. Insert the integers in the bloom filter. Your answer should first show the set of integers chosen in ascending order. Then, for each insertion, show the values of the hash functions and the bloom filter after the insertion. Do not give any explanation or write anything else. **(6)**

(b) Now, find one integer x such that $h_1(x), h_2(x), h_3(x)$ are not all the same as for any of the four integers inserted (that is, for each of these four integers y, at most two of the following equalities may hold: $h_1(x) = h_1(y)$, $h_2(x) = h_2(y)$, and $h_3(x) = h_3(y)$) such that searching for x will cause a false positive. Show the integer x chosen by you as a false positive and the hash function values for it. Do not give any explanation or write anything else. **(4)**

3. You are given a stream of songs lasting for $t_1, t_2, t_3, \ldots, t_n$ seconds. You have two write-once devices D_1 and D_2, each capable of storing songs of total duration T seconds. When the i-th song starts, you have three options: (i) copy the song to D_1, (ii) copy the song to D_2, and (iii) discard the song. The copy of a song to a device is allowed only when there is enough memory left in that device. Assume that the individual song durations t_i are known to you beforehand, and these durations satisfy $t_i \leqslant T$ for all i, and $\sum_{i=1}^{n} t_i \leqslant 2T$. Your goal is to maximize the total copy time in the two output devices together. To that effect, you run the following greedy algorithm. Derive a tight approximation ratio of the algorithm. Prove that the approximation ratio is tight. **(6 + 4)**

```
for i = 1,2,3,...,n (in that order) {
    If D1 can accommodate the i-th song, copy the i-th song to D1,
    else if D2 can accommodate the i-th song, copy the i-th song to D2,
    else discard the i-th song, and continue.
}
```

4. You have only two weeks left to solve the pending assignments of Algorithms–II. Let there be n such assignments with the i-th assignment demanding time t_i. Assume that you can solve all the assignments in one week, but since you have other courses too, you would like to distribute the Algorithms–II assignments in the available two weeks as evenly as possible. That is, you want to partition the assignments into two subsets A, B such that $\left| \sum_{a \in A} t_a - \sum_{b \in B} t_b \right|$ is minimized. Prove that this problem cannot have any polynomial-time approximation algorithm (unless P = NP). **(7)**

Answer either Q5 or Q6. Do not answer *both* Q5 and Q6. In case you do, we will grade *only* the first one we see, and ignore the other.

5. Consider an experiment in which two people are each asked to choose n numbers one by one independently and randomly from a very large range of positive integers. The numbers chosen need not be distinct, so a person can choose the same number more than once also. At the end, we want to find out if they have chosen the same collection of numbers (the same set of distinct numbers, each repeated the same number of times) or not. You are asked to design a Monte Carlo algorithm that outputs YES if they have chosen the same set of numbers, NO otherwise. The NO answer must always be correct, but the YES answer may be wrong with a probability of at most 1/2. Your algorithm should run in $O(n)$ time and $O(n)$ space.

Your solution must use hashing in some form. You do not need to specify any hash function explicitly, just explain how your algorithm will work using hashing.

Your answer should have:

(a) The pseudocode for the algorithm. **(8)**

(b) A proof that the error probability is at most 1/2. (State any assumptions you make, if any.) **(7)**

6. Consider the following variation of the traveling salesperson problem on a complete undirected weighted graph of n nodes with positive edge weights: The salesperson starts from a particular city x, and travels to a particular city y going through every other city exactly once. There is no requirement of coming back to city x. The goal is to find a path such that the maximum edge weight on the path is the minimum among all such paths. You are asked to design a branch-and-bound algorithm for this problem with DFS order of generating the state-space tree.

(a) Show the structure of the solution (one sentence only). **(2)**

(b) Show the bounding function you will use to prune the state-space tree. The bounding function will be used as follows at a node u: Before generating any child v of u, u will apply the bounding function on the state represented by v; v will be generated if and only if the bounding function returns true, otherwise v will not be generated. Your bounding function must prune nodes leading to both infeasible solutions, and feasible solutions that are not optimal. Write the bounding function clearly describing any notations/terms that you use. Do not explain the working of the bounding function. **(7)**

(c) Consider the following graph, with the numbers inside the nodes indicating the IDs of the nodes, and the numbers on the edges showing the weights of the edges. Show the state-space tree generated by applying the branch-and-bound algorithm to this graph, with node 2 as the starting city, node 5 as the final destination city, and neighboring nodes explored in increasing order of their IDs wherever required (strictly follow this order). You should only show the nodes generated (strictly as per the method of generating nodes mentioned above). Do not show any node not generated. **(6)**