National University of Singapore

CS3230: Design and Analysis of Algorithms
(Semester 2: AY2020/21)

Time Allowed: 2 hours

Instructions

- This paper consists of FOUR questions and comprises of TEN (10) printed pages, including this page.
- Answer ALL the questions.
- Write ALL your answers in this assessment book.
- This is an OPEN BOOK assessment.
- Please write your Student Number (that starts with "A") below and on the top of every page. Do not write your name.

Student Number:

Question	**Maximum**	**Score**
Q1	15	
Q2	25	
Q3	15	
Q4	25	
Total	**80**	

Question 1 [15 marks]: Consider a standard (FIFO) queue that supports the following operations:

- PUSH(x): Add item x at the end of the queue.
- PULL(): Remove and return the first item present in the queue.
- SIZE(): Return the number of elements present in the queue.

We can easily implement such a queue using a doubly-linked list so that each of the above three operations takes $O(1)$ worst-case time. (No need to show this. You can assume such an implementation.) Now, suppose we are asked to consider the following new operation:

- DECIMATE(): Remove every tenth element from the queue starting from the beginning.

```
DECIMATE():
  n ← SIZE()
  for i ← 0 to n − 1
      if i mod 10 = 0
          PULL()    ⟨⟨result discarded⟩⟩
      else
          PUSH(PULL())
```

Figure 1: Pseudocode of the DECIMATE operation (assuming starting index to be 0)

The above DECIMATE operation takes $O(n)$ time in the worst-case (where n is the number of items present in the queue). Show that in any intermixed sequence of PUSH, PULL and DECIMATE operations, the amortized cost of each of them is $O(1)$. (Use any one of the three methods we have learned in the module, i.e., aggregate method, accounting method, or potential method.)

Question 2 [25 marks]: For any set S of integers and two non-negative integers r, ℓ, we call a subset $S' \subseteq S$ an (r, ℓ)-*spaced subset* if the sum of all the integers in S' is r and for every two distinct integers $a, b \in S'$, $|a - b| \geq \ell$. Describe an algorithm that given two non-negative integers r and ℓ, and a set S of n distinct non-negative integers, counts the number of possible (r, ℓ)-spaced subsets of S in $O(rn \log n)$ time. (Assume, a single machine word is large enough to hold any integer computed during your algorithm.)

[Significant partial credits will be awarded if your algorithm runs in time $O(rn^2)$.]

Question 3 [15 marks]: A set H is said to be a *hitting set* for a family of sets $\{S_1, S_2, \cdots, S_n\}$ if and only if for all $1 \le i \le n$, $H \cap S_i \ne \phi$ (i.e., H has a non-empty intersection with all the sets S_i).

Consider the following hitting set problem: Given a family of finite integer sets $\{S_1, S_2, \cdots, S_n\}$ and a positive integer K, decide whether there exists a hitting set of size at most K. Show that the hitting set problem is NP-complete.

(You may show a reduction from any of the NP-complete problems introduced in the lectures/ tutorials/ assignments/ practice set, including Circuit Satisfiability, CNF-SAT, 3-SAT, Vertex Cover, Independent Set, Max-Clique, Hamiltonian Cycle, Traveling Sales Person Problem.)

Question 4 [25 marks]: Consider the following *shortest superstring* problem: Given a set of n strings $S = \{x_1, x_2, \cdots, x_n\}$ (all the strings are over some arbitrary finite alphabet Σ), find a shortest string s over the alphabet Σ that contains each x_i as a substring. Assume, no string x_i is a substring of another string x_j, for $j \neq i$. (E.g., for the set $S = \{abab, abc, cab\}$, a shortest superstring is $cababc$.)

For any string z, define $set(z) := \{x_i \in S \mid x_i \text{ is a substring of } z\}$. For any two distinct strings $x_i, x_j \in S$ and an integer $k > 0$, if the last k symbols of x_i are the same as the first k symbols of x_j, define $y_{i,j,k}$ to be the string obtained by overlapping these k positions of x_i and x_j (i.e., remove the last k symbols from x_i and then concatenate x_j after this leftover string). E.g., let $x_i = ababcda$ and $x_j = bcdab$. Then $y_{i,j,4} = ababcdab$. However, $y_{i,j,3}$ is not defined since the last three symbols of x_i are not the same as the first three symbols of x_j.

Next, consider the set cover problem we have seen in lecture 12. Given S, create a set cover instance $\mathcal{S}$ as follows: For all the valid choices of i, j, k, create $y_{i,j,k}$, and let M be the set of all these strings. The universe for the set cover instance is S, and the input sets are $set(z)$, for all $z \in S \cup M$. Let the *cost* of $set(z)$ be equal to the length of the string z.

Now, use the following algorithm for the shortest superstring problem:

1. Use any approximation algorithm $\mathcal{A}$ for the set cover problem (e.g., that covered in the lecture) to find a cover for the instance $\mathcal{S}$ (described above). Let $set(z_1), \cdots, set(z_r)$ be the sets output by that algorithm.

2. Concatenate the strings $z_1, \cdots, z_r$ in any arbitrary order.

3. Output the resulting string.

Let the approximation ratio of the set cover algorithm $\mathcal{A}$ be $\alpha(n)$. Then show that the approximation ratio of the above algorithm for the shortest superstring problem is $2\alpha(n)$.

[**Hint:** Take a shortest superstring s of the input S, and then consider the starting positions of x_i's in it. Next, observe how some of the strings $y_{i,j,k}$ appear (with overlap) in s. Then consider the corresponding sets $set(y_{i,j,k})$ to form a valid set cover of the instance $\mathcal{S}$. Using this, establish a relation between the optimum cost of the set cover instance $\mathcal{S}$ and the length of s.]

—End of Paper—