430.329 (001)
Introduction to Algorithms

HW#2

2018 Fall

1. **[10 pts]** Show how to implement a queue using two stacks. Analyze the running time of the queue operations: *enqueue, dequeue*.

2. **[20 pts]** Suppose you combine open addressing with chaining. You build a hash table with *m* buckets implemented by an array that can store at most two keys in each bucket. Suppose that you have already inserted *n* keys using the following algorithm.

   A. Hash to one of the *m* buckets.
   B. If the bucket has less than two keys, insert it to the bucket using chaining.
   C. Otherwise, increment the probe number and go to step A.

   Assuming that you already have a hash table with *n* keys and you want to insert a new key. You are going to derive the probability to insert the key at the first probing, i.e the probability of finding a bucket that is completely empty or half-empty at the first probe.

   You can make the uniform hashing assumption for all the parts of this question.

   (a) **[5 points]** There are ***w*** buckets in the table that are entirely full. Derive the probability p(*w*) that the first probe is successful, assuming that there are exactly ***w*** full buckets.('Entirely full' means that a bucket already has two keys.)

   (b) **[5 points]** We define q(*t*) as the probability that **t** buckets in the table are entirely full, assuming that the table have ***n*** keys already. Derive the probability that the first search would be successful in terms of q(*t*).

   (c) **[10 points]** Derive probability q(0) in terms of *m* and *n*.

3. **[35 pts]** Suppose you are an administrative staff in Computer Vision Laboratory in SNU. There are many researchers and interns in the laboratory, so there is high demand for GPUs. CVLab has N GPUs, and the researchers use and return those GPUs through the day. When a new student applies for CVLab, they want to use GPU whose number is in the range [a, b].

   You decide to design a data structure for 'GPU-managing' operation to allocate GPU as efficiently as possible.Your data structure has the following operations.

   1. **`Init(N)`** : Initialize your data structure for **N** GPUs (indexed 1, 2, … , N) as **'Not-in-use',** in polynomial time.
   2. **`Count(a, b)`** : Return the number of GPUs which is not being used in [a, b] , in O(logN) time.
   3. **`Assign(a, b)`** : If all GPUs are being used, return **NIL.** Otherwise, return the first GPU index in [a, b] which is not being used and mark it as **'In-use'.**
   4. **`Return(q)`** : Mark GPU **q** as **'Not-in-use'**, in O(logN) time.

   (In this problem, you can use basic data structure operations such as **insert**, **delete** without any specification. Scores will be given when your answer is logically correct)

   (a) **[7 points]** What data structure will you use? Discuss why you use the data structure.

   (b) **[7 points]** Design an algorithm of **`Init(N)`**. Running time of algorithm you design should be polynomial with respect to N.

   (c) **[7 points]** Design an algorithm, **`Count(a, b)`** in O(logN) time.

   (d) **[7 points]** Design an algorithm, **`Assign(a, b)`** in O(logN) time.

   (e) **[7 points]** Design an algorithm, **`Return(q)`** in O(logN) time.

4. **[Programming Exercise]**

[35 pts] Given two strings **A** and **B**, let *L* be the length of their LCS (The Longest Common Subsequence). We want to increase *L* by one, to *L*+1. This can be achieved by inserting a single character to string **A** (or to string **B**, but just consider the case of inserting a character to string **A** for convenience in this homework). Let *n* be the number of total ways to insert a character at any position in string **A** such that increase *L* to *L*+1. Please check the examples below.

1) **[15 points]** Design a function that returns *L,* the length of the LCS between two strings.

2) **[20 points]** Design a function that returns *n,* the number of total ways to insert a character at any position in string **A** such that increase *L* to *L*+1.

The skeleton code for the homework will be provided. Please implement your function in that file, and submit the file to the ETL as “Student-ID.py”. e.g.) 2018-9999.py

Example 1) :

Input: A = ‘bca’

B = ‘baaa’

Length of LCS : len(‘ba’) = 2

Way to increase LCS by one:

→ Insert ‘a’ to A → ‘b**a**ca’, ‘bc**a**a’, ‘bca**a**’ → Length of LCS : len(‘baa’) = 3

→ 3 ways

Example 2) :

Input: A = ‘caa’

B = ‘baaa’

Length of LCS : len(‘aa’) = 2

Way to increase LCS by one:

→ Insert ‘a’ to A → ‘**a**caa’, ‘c**a**aa’, ‘ca**a**a’, ‘caa**a**’ → Length of LCS : len(‘aaa’) = 3

→ Insert ‘b’ to A → ‘**b**caa’, ‘c**b**aa’ → Length of LCS : len(‘baa’) = 3

→ 6 ways