Algorithms Homework #1

Due: 2017/10/11 (Wed.) 03:00 (Programming)
2017/10/11 (Wed.) 14:20 (Hand-written)

Contact TAs: alg2017ta@gmail.com

Instructions

- There are two sections in this homework, including the hand-written part and the programming part.
- **Hand-written.** Please submit your answers to the instructor at the beginning of the class. TA will collect the answer sheets in the first break of the class. **NO LATE SUBMISSION IS ALLOWED**.
- **Programming.** Please upload your source codes in a single .tar or .zip file (e.g. b04901001_hw1.zip) to Ceiba. Your codes should be implemented in **Python 3**. In addition, any code that does not follow the rules specified in the README.txt file will not be graded, so you are encouraged to read the requirements carefully before starting your work.
- **Delay policy.** You have a six-hour delay quota for the programming assignments for the whole semester. Once the quota is used up, any late submission will receive no credits.
- **Collaboration policy.** Discussion with other students is strongly encouraged, but you must obtain and write the final solution by yourself. Please specify the references (e.g. the name and student id of your collaborators and/or the Internet URL you consult with) for each question. If you solve some problems by yourself, please also specify "no collaborators". Homeworks without reference specification may not be graded.
- **Academic honesty.** Cheating is not allowed and is against university policy. Plagiarism is a form of cheating. If cheating is discovered, all students involved will receive an F grade for the course (NOT negotiable).

Hand-Written Problems

Problem 1

Let all functions be positive. Prove or disprove the following statements by using only the definition of asymptotic notations. No credit will be given if you simply answer True/False without any explanation.

1. (4%) $\sqrt{n} \in \Omega(n)$.
2. (5%) $\binom{2n}{n} \in \Omega(2^n)$.
3. (5%) $\binom{2n}{n} \in O(4^n)$.
4. (6%) If $f(n) \in \Theta(h(n))$ and $g(n) = f(\lfloor \frac{n}{2} \rfloor)$, then $g(n) \in O(h(n))$.

Problem 2

Solve the following recurrences (in $\Theta()$ notation). If you decide to apply any method/technique which is not taught in class or stated in the textbook, you must prove it before using it. Assume that $T(1) = 1$.

1. (6%) $T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + cn$, where c is a positive constant.
2. (6%) $T(n) = 3T(\frac{n}{3}) + n^2 \log_3 n$.
3. (8%) $T(n) = 3T(\frac{n}{3}) + \frac{n}{\log_3 n}$.

Problem 3

(20%) Frank is a shopaholic. One day he goes to a mall and finds that the T-shirts from his favorite brand are all on-sale. However, he bought too many shirts last month and has only C dollars left. There are n T-shirts in the store with distinct prices $a_1, \ldots, a_n$. Frank loves them all, so he decides to buy as many shirts as possible under his budget. At this moment, his wife calls him to go back home immediately because their daughter is crying for daddy. Please design a **linear time** algorithm to help Frank solve this shopping task. Assume that Frank is only allowed to directly compare the prices of two T-shirts, while he is able to ask the shopkeeper to check whether the total cost of the shirts he picks exceeds the budget in $O(1)$ time (i.e., if $A = \{a_1, ..., a_n\}$ and B is any subset of A, then Frank is able to know whether $\sum_{a_j \in B} a_j \leq C$ in $O(1)$). Also, since Frank is very familiar with this brand, he can always find the k-th cheapest T-shirt in linear time for every k. Briefly explain the correctness of your algorithm and analyze the running time.

Programming Problems

Problem 1

(15%) Sort an array by algorithms based on merge sort. In addition, you need to count how many times you split the array and return the value (e.g., an array of length 4 requires 2 splits, $4 \to [2, 1, 1]$ and $2 \to [1, 1, 0]$). In this problem, you MUST split an array into **three** parts instead of two as usual. Assume that an array A is split into sub-arrays a_0, a_1 and a_2. The length of the sub-arrays must satisfy the following constraints:

1. $|a_0| \geq |a_1| \geq |a_2|$
2. $|a_i - a_j| \leq 1$ for all $0 \leq i, j \leq 2$

You must split an array until the length of its sub-arrays is 1 or 0. Some examples are listed below:

- An array of length 4 must be split into sub-arrays of length [2, 1, 1].
- An array of length 5 must be split into sub-arrays of length [2, 2, 1].
- An array of length 2 must be split into sub-arrays of length [1, 1, 0].

Problem 2

The stable matching problem (a.k.a. stable marriage problem) is described as below:
Given two groups with each containing n people (e.g., n men and n women). Each person ranks all members of the other group in order of preference. A matching M is a one-one correspondence between the men and the women. A matching is *unstable* if and only if the matching exists a pair (x, y) such that:

- x prefers y to his originally assigned partner y', **and**
- y also prefers x to her originally assigned partner x'

The goal is to find a matching that is stable.

1. (2%) Anthony learned G-S algorithm in class last week, but he did not pay enough attention and misunderstood the definition of *unstable*. He thought that the matching is unstable if either of the conditions is true, that is, he misinterpreted 'and' as 'or'. Can you give Anthony a counter example and explain why the process of G-S algorithm will not terminate if he uses the wrong definition? (hand-written problem)

2. (3%) The stable roommate problem is described as below:
Given 2n people. Each people ranks other 2n-1 people in order of preference. Similar to the stable marriage problem, the goal is to find a stable matching of n pairs, but pairing is allowed between any two people. However, there may not be a solution to this problem. Give an example of the stable roommate problem that is not stable and explain why it is unstable. (hand-written problem)

3. (20%) Please implement the stable matching problem. For more information, please refer to the zip file.

Please upload your source codes using the following format. Only `mergesort.py` and `match.py` should be uploaded. Compress the folder and rename it as {student id}_hw1.zip or {student id}_hw1.tar.

```
b04901001_hw1/
  P1/
    mergesort.py
  P2/
    match.py
```