CIS507: Design & Analysis of Algorithms

Homework 1, Spring 2014

Submissions are due as a zip or rar file emailed to **mjha@masdar.ac.ae** and **bhayk@masdar.ac.ae**; CC: **kelbassioni@masdar.ac.ae** by **Wednesday, March 5th, 23:59:59 UAE time**. Acceptance of late submissions and penalties for the same are at the sole discretion of the course faculty.

Code is to be written in **python**, completed and submitted individually. Copying of code will invite severe penalties. Source code is to be submitted in a file **<last-name>-cis507hw1.src** and analyses/explanations are to be submitted in a LaTeX-generated PDF file **<last-name>-cis507hw1.pdf** (add your answers to the file <last-name>-cis507hw1.tex provided, then compile it using pdflatex).

Please ensure that the source files are tested thoroughly before committing to submission - post-submission bug fix requests will be entertained, again, at the sole discretion of the course faculty. Also, please ensure that your source file encoding is cross-platform readable (does not break when compiled on Linux/Windows/etc).

There is a total of 13 points in this assignment; for a perfect score, you need 10 points.

Q1. Give asymptotic upper bounds for $T(n)$ for each of the following recurrences (use the O-notation). Assume that $T(n)$ is a non-negative constant for n sufficiently large (in terms of α). Make your bounds as tight as possible, and justify your answers.

1. **(1 point)** $T(n) = n^{1-\alpha} \cdot T(n^{\alpha}) + \Theta(n)$, for a constant $\alpha \in (0, 1)$.
2. **(1 points)** $T(n) = T(n-1) + T(\alpha \cdot n) + 1$, for a constant $\alpha \in (0, 1)$.

Q2. **(2 points)** Consider the following problem called MAXCUT: given an undirected graph $G = (V, E)$ with non-negative edge weights w_e for $e \in E$, find a partition $(S, V \setminus S)$ of the vertices that maximizes the total weight of the edges crossing the cut, that is, $\sum_{e\in\delta(S)} w_e$, where $\delta(S)$ is the set of edges that have one end-point in S and another in $V \setminus S$.

Consider the following randomized algorithm: Select a subset S by picking each vertex in V independently with probability $\frac{1}{2}$. Show that the expected wight of the edges in the cut $(S, V \setminus S)$ is a factor of $\frac{1}{2}$ of the total weight, that is:

$$\mathbb{E}\left[\sum_{e\in\delta(S)} w_e\right] = \frac{1}{2}\sum_{e\in E} w_e$$

(*Hint:* use an indicator random variable for each edge.)

Q3. Suppose that we would like to analyze the change in price for a given stock. We observe the different prices over a period of n days. Let $A[i]$ be the observed price in day i. We would like to compute:

(I) the smallest absolute price difference: $\min_{1\le i,j\le n,\ i\ne j} |A[i] - A[j]|$;

(II) the largest absolute price difference: $\max_{1\le i,j\le n} |A[i] - A[j]|$;

(III) the average absolute price difference: $\frac{1}{n(n-1)} \sum_{1\le i,j\le n} |A[i] - A[j]|$;

(IV) the median absolute price difference: $\text{median}(\{|A[i] - A[j]| : 1 \le i, j \le n\})$.

(i) **(1 point)** give an $O(n^2)$ deterministic algorithm for computing (I), (II), (III) and (IV);

(ii) **(1 point)** give an $O(n \log n)$ deterministic algorithm for computing (I);

(iii) **(1 point)** give an $O(n)$ deterministic algorithm for computing (II);

(iv) **(1 point)** give an $O(n \log n)$ deterministic algorithm for computing (III);

(v) **(1 point)** give a randomized algorithm with $O(n^2)$ expected running time for computing (IV).

Implement the four algorithms in (ii), (iii), (iv) and (v). For testing purposes, your program should accept as an input a file "test.in", containing n, followed by the set of n numbers (1 per line). It should output the four values described in (I), (II), (III), and (IV).

Q4. (4 points) Implement a *perfect* hash table, where keys are decimal numbers, each having at most 10 digits. For both hash levels, use the class of universal hash functions of the dot-product form: if the hash table size is a prime m, pick a random sequence $\mathbf{a} := \langle a_0, a_1, \ldots, a_9 \rangle$, where each $a_i \in \{0, 1 \ldots, m-1\}$; given a key k, decompose it into a sequence of decimal digits $\mathbf{k} := \langle k_0, k_1, \ldots, k_9 \rangle$, then use hash functions of the form $h_{\mathbf{a}}(k) = (\sum_{r=0}^{9} a_i k_i) \bmod m$. Your table should have no collisions, and and uses at most $8n$ table entries, in total. (*Hint:* use the fact that for any positive integer n, there is at least one prime between n and $2n$.)

For testing purposes, your program should accept as an input a file "test.in" containing the number of keys n, followed by the set of keys to be hashed (1 per line). It should output in another file "test.out", the following lines: the first line (call it line 0) contains the values chosen for the first-level hash function in the following order (separated by spaces): $m, a_0, a_1, \ldots, a_r$; then for $i = 1, \ldots, m$, the ith line contains the values corresponding to the second level-hash function chosen at the ith row in the first level table (again in the order $m(i), a_0(i), a_1(i), \ldots, a_r(i)$; output "0 0" if that row is empty). Following this, the file should contain triples (one per line): $(k, h(k), h_i(k))$, where k is the key, $i = h(k)$ is the index in the first-level hash table, $h_i(k)$ is the index in the second level hash table.