CS 473u: Algorithms, Spring 2005
Homework 1, due February 10 23:59:59, 2005

Name:
Net ID: Alias: U G

Name:
Net ID: Alias: U G

Name:
Net ID: Alias: U G

<table>
<thead>
<tr>
<th>#</th>
<th>Score</th>
<th>Grader</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You will submit problems 1,2,3 and problems 4,5 in separate stapled sets. Print and fill out two copies of this sheet; attach a copy to the top of each of the two parts of your homework. Indicate which part is which by circling the relevant problems [1,2,3 or 4,5] on the cover sheet. On both copies of the cover sheet, neatly print your name(s), NetID(s), and the alias(es) you used for Homework 0 in the boxes above.
“Today I know that everything watches, that nothing goes unseen, and that even wallpaper has a better memory than ours. It isn’t God in His heaven that sees all. A kitchen chair, a coat-hanger a half-filled ash tray, or the wood replica of a woman name Niobe, can perfectly well serve as an unforgetting witness to every one of our acts.” – The tin drum, Gunter Grass

Required Problems

Note: Be sure to check out the updated Homework FAQ on the course webpage:

http://www-courses.cs.uiuc.edu/~cs473u/

1. Swapyland  
[20 Points]

A swap is the exchange of position of two adjacent elements in an array. Consider an array $A[1..n]$ storing the integer numbers $1 \ldots n$ in a certain permutation, and the array $B[1..n]$ which is another permutation of those numbers.

The swap distance $d(A, B)$ denotes the minimum number of swaps needed to move from $A$ to $B$.

(a) [5 Points] Show that for permutations $A$ and $B$ selected uniformly at random, the expected value of $d(A, B)$ is $n(n - 1)/4$.

(b) [15 Points] Show a $O(n \log n)$ time algorithm for computing the swap distance between $A$ and $B$ (hint: use a procedure similar to merge sort).

2. Median Land  
[20 Points]

Let $A[1..m]$ and $B[1..n]$ be two sorted arrays, and let $k$ be a positive integer. The element of rank $k$ in the set $C = \{A[1], A[2], \ldots, A[m], B[1], \ldots, B[n]\}$, is the number in this set having exactly $k - 1$ numbers smaller than it in $C$.

Provide an algorithm that computes the element of rank $k$ given $A$ and $B$ in $O(\log(n+m))$ time. (Partial credit will be given to a solution with running time $O((\log n)(\log m))$.)

3. 3Sum  
[20 Points]

Let $A$ be a set of $n$ positive integer numbers in the range $[1, \ldots, M]$. The 3Sum problem asks if there are three numbers $a, b, c \in A$ such such that $a + b = c$.

(a) [10 Points] Give a simple $O(n^2)$ time algorithm for this problem. (Hint: Getting $O(n^2)$ is somewhat tricky. Try first to get $O(n^3)$ time algorithm, and then $O(n^2 \log n)$ time algorithm. Finally improve it into the required running time.)

(b) [10 Points] Give a $O(M \log M)$ time algorithm for this problem, assuming $M > n$. (Hint: Use FFT.)
4. Discrepancy

[20 Points]

Let \( R \) and \( B \) be two sets of red and blue points, respectively, on the real line. Let \( n \) be the total number of points in both sets. The discrepancy of a closed interval \( I \) is the quantity \( \mathcal{D}(I, R, B) = |I \cap R| - |I \cap B| \). In words, this is the difference in the number of red points to blue points inside the interval \( I \).

(a) [5 Points] Give an \( O(n \log n) \) time algorithm that outputs the interval with the maximum discrepancy.

(b) [5 Points] The Kullback-Leibler divergence for an interval \( I \) that contains at least one blue point, and at most \( |B| - 1 \) blue points, is

\[
\text{disc}(I) = r \ln \frac{r}{\hat{b}} + (1 - r) \ln \frac{1 - r}{1 - \hat{b}},
\]

where \( r = |R \cap I| / |R| \) and \( b = |B \cap I| / |B| \). Describe an algorithm that computes the interval with the maximum KL-divergence, that works in \( O(n^2) \) time.

(c) [10 Points]

Now assume that \( |R| = |B| = n \). We consider the minimum matching question, as follows. Partition the set \( R \cup B \) into pairs \( (r_1, b_1), \ldots, (r_n, b_n) \) such that each pair has one blue point and one red point, and every point of \( R \cup B \) participates in exactly one pair. The price of the matching is \( \sum_{i=1}^{n} |r_i - b_i| \). Give an \( O(n \log n) \) time algorithm that computes the minimum-price matching. Prove the correctness of your algorithm.

5. k-means Clustering

[20 Points]

Let \( P \) be a set of \( n \) points on the real line. In the k-means clustering problem, one wishes to compute a set \( C \) of \( k \) points, such that the price of clustering \( P \) using \( C \) is minimized. The price for a point \( p \in P \) to be in a cluster that uses a center \( c \in C \) is the squared distance \( |p - c|^2 \). Thus, naturally, to minimize the price, every point is going to be served by the closest point to it in \( C \). Let \( C(p) \) denote the closest point to \( p \) in the set \( C \). Thus, the clustering price of \( P \) using \( C \) is

\[
\mu_C(P) = \sum_{p \in P} |p - C(p)|^2.
\]

(a) [5 Points] Given \( P \) and \( C \) such that \( |P| = n \) and \( |C| = k \), show an algorithm that computes \( \mu_C(P) \) in \( O(n \log k) \) time.

(b) [5 Points] Prove that if \( Q \) (a set of points on the real line) is served by a single center \( c \), then \( \mu_{\{c\}}(Q) \) is minimized when \( c \) is the center of gravity of \( Q \) (i.e., the mean of \( Q \)).

Formally, the minimum is achieved when \( c = \left( \sum_{q \in Q} q \right) / |Q| \).

(c) [10 Points] In the k-means clustering problem, one wants to partition \( P \) into \( k \) clusters, such that the k-means clustering price is minimized. Formally, describe an algorithm that receives as input a set \( P \) of \( n \) points, and outputs the set \( C \) of \( k \) numbers such that \( \mu_C(P) \) is minimized. Your algorithm should be as fast as possible.