Programming Assignment

Design and Analysis of Algorithms

March 9, 2010

1 Introduction

This is a programming assignment where you will learn to apply the algorithms that you have learnt in class to real life problems. We have talked about different algorithms in class and we have also analyzed their running times. In this programming project you will be given a problem and you are expected to write a program to solve the problem. In the process you are expected to understand the problem and come up with an effecient algorithm for solving the problem and then implement your algorithm in a programming language of your choice. For the problem that is stated, there is always a naive algorithm and, there is a better and faster algorithm. Implementing the naive algorithm will be fine but you will get extra credit for faster algorithms. Your implementation will be timed for the speed that you can achieve. Also the implementation will be fed a dataset other than the one that will be provided for testing and your implementation will be evaluated on the basis of the time it gives on the dataset. The algorithm and its implementation will also be evaluated on the basis of correctness. So if you have an implementation that is fastest of all but is WRONG then you will get no credit. Your code should be well documented and should follow the paradigms of object oriented design. For this problem at the minimum it should have two classes.

2 Problem Definition

The United States Census Bereau conducts census on various aspects of the population in the United States. Based on the data that is collected, the different states in the United States are ranked on the basis of the data. The data aims to collect and capture different aspects of the population in the different states. The aspects may be percentage of hispanic population, or the mortality rates in the different states. The data is available from the United States Census Bereau in HTML format and can be mined to generate the same data in plain text format. All these data rank the diffeent states in the United States on the basis of some parameter based on the collected data. These rankings of the different states generate what is known a permutation of a base set. So for example given the set $\{1, 2, 3\}$ the permutation of this set may be $\{3, 2, 1\}$. And in general given a set of n positive integers we can have $n!$ permutations. The ranks of the states can be treated as a permutation in the sense that if we have 50 states in the United States then the rankings are actually a subset of the 50! permutations that we can create of the numbers $\{1, 2, 3, 4, ..., 50\}$. One interesting problem with permutations is, how to define the notion of distance between two different permutations. This is a widely studied problem in Statistics, Mathematics and Computer Science and there are different notions of the distance between two permutations and one such is the notion of inversions. Given two permutations p_1 and p_2 the inversion distance between them is the number of changes you have to do in one to get the other. We will study this problem in the more general case when the permutations are such that the same number may be repeated at different positions. This is analogous to the case where a number of states have the same rank with respect to a particular variable. The problem is how to define the notion of distance between two permutations that have the same

number repeated at different positions. Now when such rankings are done if there is a tie at a position then the next rank is skipped. So if you have a base set of the states $\{s_1, s_2, s_3, s_4, s_5\}$ that you want to rank and, there is a tie for the third position between the states number 1 and 3, then the ranking will be $\{3, 2, 3, 1, 5\}$, meaning that s_1 and s_3 have rank 3, s_2 has rank 2, s_4 has rank 1, and s_5 has rank 5. Note that rank number 4 is missing as there is a tie. In

this problem we will consider two distance measures for computing the distance between two permutations with ties. Both of these measures are variations of what is called the Kendell Tau measure. In the discussion that follows we call permutations with ties as partial rankings in keeping with the accepted norm in the literature. Now we put what we have said till now in more formal grounding.

Formal Problem Statement

First we start by defining a bucket order. A bucket order is a linear order with ties. More formally, a bucket order is a transitive binary relation $\prec$ for which there are non-empty sets $B_1, ..., B_k$ (called buckets) that form a partition of the domain such that $x \prec y$ if and only if there are i, j with $i < j$ such that $x \in B_i$ and $y \in B_j$. We say that bucket B_i precedes bucket B_j if $i < j$. The buckets are actually collection of objects that have the same rank and these divide the whole set into classes and more formally they form an equivalence class.

We associate a partial ranking with each bucket order, by letting $\sigma(x) = i$ when $x \in B_i$. We assume that all partial rankings have the same domain, denoted D. Without any loss in generality, we shall assume that $D = \{1, ..., n\}$. We say that x and y are tied in σ if $\sigma(x) = \sigma(y)$.

Let σ_1 and σ_2 be two partial rankings on the domain D with bucket sets $B_1, B_2, ..., B_k$, and $C_1, C_2, ..., C_m$ respectively (where bucket B_i precedes bucket $B_{(i+1)}$ for $1 \leq i < k$, and bucket C_i precedes bucket $C_{(i+1)}$ for $1 \leq i < m$). Let $P = \{\{i, j\} : i \neq j \text{ and } i, j \in D\}$ be the set of unordered pairs of distinct elements from D. We partition P into the following three sets:

- $D(\sigma_1, \sigma_2)$ is the set of all $\{i, j\} \in P$ such that i and j appear in different order in σ_1 and σ_2 ; that is, either $\sigma_1(i) < \sigma_1(j)$ and $\sigma_2(i) > \sigma_2(j)$, or vice versa.
- $R1(\sigma_1, \sigma_2)$ is the set of all $\{i, j\} \in P$ such that i and j are tied in σ_1 but not tied in σ_2 ; that is, $\sigma_1(i) = \sigma_1(j)$ and $\sigma_2(i) \neq \sigma_2(j)$.
- $R2(\sigma_1, \sigma_2)$ is the set of all $\{i, j\} \in P$ such that i and j are tied in σ_2 but not tied in σ_1 ; that is, $\sigma_2(i) = \sigma_2(j)$ and $\sigma_1(i) \neq \sigma_1(j)$.

The Kendall distance with penalty parameter p, denoted by $K^{(p)}$, is defined to be $|D(\sigma_1, \sigma_2)| + p * (|R1(\sigma_1, \sigma_2)| + |R2(\sigma_1, \sigma_2)|)$ where p is a real number in the interval $[0, 1]$.

The Hausdorff distance based on Kendall distance, denoted by K_{Haus} , is equal to $|D(\sigma_1, \sigma_2)| + max\{|R1(\sigma_1, \sigma_2)|, |R2(\sigma_1, \sigma_2)|\}$.

Given the dataset, you will be expected to write programs to implement the calculation of the above distance measures. The dataset that is provided has rankings for the different states and territories of the United States. The ground set has 56 states and territories :

```
Alabama
Alaska
American Samoa
```

Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Northern Mariana Islands
Ohio
Oklahoma
Oregon
Pennsylvania
Puerto Rico Rhode Island
South Carolina
South Dakota
Tennessee
Texas
US Virgin Islands
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin

`Wyoming`

3 Deliverables

At the least you are expected to deliver the following:

- A working algorithm that will compute the distance measures for the different permutations.
- The algorithm should compute the distances as defined above correctly.
- The full source code and any documentation that you would like to make to illustrate your methods.

4 Important Notes

- The given dataset has 242 variables and 56 states.
- There are sets of rankings where there are less than 56 states that have been ranked in a set. But you are expected to compute the diatance measure between two permutations of the same size namely 56. So you will need to figure out how you will make each of the ranking set to be equal to the size of the ground set that is 56. The way you do this will affect the reliability of the measure that you will be computing. As a result you should think how you can take say a set of rankings of 50 states and then complete this ranking set to get a set of rankings of size 56. One way to do this would be to add the missing states with same rankings for all of them. But this is a suggestion. You should think and find out ways of doing this. You should be able to justify your decision.
- The data source that you are given is in the form of a text file (.txt) with all the information in it. The first column gives the ranks, the seconf the states and the last the variable based on which the rankings are done. There will be multiple datasets in the file and each of the dataset has the title that describes that dataset. Also, even and odd rankings are separated in each list, so you'll need to merge/sort them. You might want/need to split the file using "Title:" as a grep marker first. If the number of states listed below a "Title" is less than 10, ignore that variable (might be too tough to guess the ranking of every other state for this variable).
- You will need to write code to preprocess the datafile to extract the relevant information and then use your algorithm and the definitions given above to compute the distances.
- Your code should take the datafile as the input and for each pair of the permutations in the datafile print the pairs in two different rows and in the third row should print the two distances seperated by a tab space. For example (use p=0.5):

  Row 1:1 3 2 4 4

  Row 2:2 2 1 4 5

Distance1 Distance2

- The full source code and any documentation that you would like to make to illustrate your methods.
- The weights for the different aspects of your implementation are as follows: Documentation 10 percent, Correctness 50 percent, Speed 30 percent and Design of the code 10 percent.
- The data will be available on the Black Board site and you can download the zip file containing the data from there.The location is in the assignments folder in the course site for the class. There should be a link in your black board site to the course page for Design and Analysis of Algorithms. Inside that is the Assignments link. The data file is in data.zip in the assignments link. You should download and unzip the file and it will create a directory on the disk named data. The actual text file containing the data is in that directory.

You may be required to give a presentation of what you have done. There you may be asked to explain why you did certain things that you did. So try to be logical and try to have reasons for whatever you do. The algorithms and the implementation will be judged on the basis of correctness and speed. As speed is a consideration in the evaluation it is expected that you code in C or C++. You can code in any other language but languages like Java or Python will run more slowly than either C or C++.