Middle East Technical University Department of Computer Engineering

CENG 242

Programming Language Concepts

Spring 2013-2014

Homework 6 - Kendall's τ

Due date: 8 June 2014, Sunday, 23:55

1 Objective

This homework aims to help you get familiar with basic programming concepts of Prolog.

2 Problem Definition

In the real world, rankings between items are frequently used especially in user studies. For example, when a cosmetics firm produces a set of perfumes, it conducts an experiment to assess how much the people like the new candidate perfume formulas. In such a study, a set of participants may be asked to compare and rank the fragrances from the best to the worst according to their subjective evaluation. Such an experiment provides insight about the preferability of an item over the other alternatives.

In this homework, you will write a Prolog predicate to find the Kendall's τ (tau) distance between two rankings, which is a measure of similarity between the rankings. A ranking is represented as a list of integers $[r_0, r_1, ..., r_{n-1}]$ where n is the number of the items compared. r_i is the rank of item i and $r_i \in \{1, 2, ..., n\}$. In this homework, there will not be a tie between the items (i.e. two different items will have distinct ranks).

Given these definitions, the Kendall's τ distance between two rankings R_1 and R_2 is defined as follows:

$$\tau = \frac{n_c - n_d}{n(n-1)},$$

where n_c is the number of *concordant pairs*, n_d is the number of *discordant pairs* and n is the number of the items ranked. With this formula, the maximum value of τ is 1 (complete agreement between the two rankings) and the minimum value is -1 (complete disagreement).

Given two rankings $R_i = [r_{i0}, r_{i1}, ..., r_{in-1}]$ and $R_j = [r_{j0}, r_{j1}, ..., r_{jn-1}]$, a pair of items $\langle a, b\rangle$ is a *concordant pair* if

$$((r_{ia} < r_{ib}) \wedge (r_{ja} < r_{jb})) \vee ((r_{ia} > r_{ib}) \wedge (r_{ja} > r_{jb})),$$

and a *discordant pair* if

$$((r_{ia} < r_{ib}) \wedge (r_{ja} > r_{jb})) \vee ((r_{ia} > r_{ib}) \wedge (r_{ja} < r_{jb})).$$

Suppose that two rankings are given as $R_1 = [1, 3, 2]$ and $R_2 = [1, 2, 3]$. All possible item pairs and whether they are *concordant* or *discordant* are given in the Table 1. For example, the item pair $\langle 0, 1\rangle$ is

Table 1: Pairs of items and their ranks in R_1 and R_2

Item 1	**Item 2**	**Rank of Item 1 in R_1**	**Rank of Item 2 in R_1**	**Rank of Item 1 in R_2**	**Rank of Item 2 in R_2**	**Concordant?**	**Discordant?**
0	0	1	1	1	1	No	No
0	1	1	3	1	2	Yes	No
0	2	1	2	1	3	Yes	No
1	0	3	1	2	1	Yes	No
1	1	3	3	2	2	No	No
1	2	3	2	2	3	No	Yes
2	0	2	1	3	1	Yes	No
2	1	2	3	3	2	No	Yes
2	2	2	2	3	3	No	No

a *concordant pair* since $r_{10} = 1 < r_{11} = 3$ and $r_{20} = 1 < r_{21} = 2$. On the other hand, the pair $\langle 1, 2 \rangle$ is a *discordant pair* since $r_{11} = 3 > r_{12} = 2$ and $r_{21} = 2 < r_{22} = 3$.

According to the given R_1 and R_2 there are 4 *concordant pairs* and 2 *discordant pairs*. Since there are 3 items in each ranking, n is 3. The Kendall's τ distance is calculated as:

$$\tau = \frac{4-2}{3(3-1)} = \frac{2}{6} \approx 0.33333$$

3 Specifications

- You will write a Prolog predicate named `ktau`/3 to calculate the Kendall's τ distance between two given rankings.
- The first variable will be a list representing the first ranking result.
- The second variable will be a list representing the second ranking result.
- The third variable will be a floating point number representing the Kendall's τ distance.
- The third variable will be queried during the grading and your predicate definitions must find only a single value satisfying the predicate, which is the correct answer according to the definition of the Kendall's τ distance.
- The rankings will be represented as lists consisting of integers. Each integer in $\{1, 2, 3, ..., n\}$ will appear exactly once in the list, where n is the number of items ranked.
- The length of the lists will be the same and will be at least 2.
- You may write other helper predicates; however, all of your code should be in a single file (see Section 6).
- There is no restriction on the maximum length of the lists.
- You are free to implement any algorithm to solve the problem as long as it is designed and coded by you.
- No external libraries allowed (your code must not contain calls to any predicate such as `consult`/1, `use_module`/1, `load`/1, `import`/1, `include`/1 or any other similar predicate for this purpose).

4 Sample Run

```
| ?- ktau([1,2], [1,2], T).

T = 1.0 ?

yes
| ?- ktau([1,3,2], [1,2,3], T).

T = 0.33333333333333331 ?

yes
| ?- ktau([1,2,3], [3,2,1], T).

T = -1.0 ?

yes
| ?- ktau([1,6,7,3,5,2,4], [6,2,5,3,7,1,4], T).

T = 0.14285714285714285 ?

yes
| ?-
```

5 Regulations

- **Programming Language:** You must code your program in Prolog. Your submission will be graded with `prolog` on department lab machines.
- **Late Submission:** The allowed number of late days is the minimum of $\{3, 10 - n\}$ where n is the sum of the number of late days used in all of the previous homeworks.
- **Cheating:** It is ok to discuss material in the lecture and recitation notes with others. However, please do not discuss anything specific to this programming assignment or to your implementation with anyone else. Please do not look anyone else's code (including source code found on the internet), or show your code anyone else. Everything you submit must be your own work. People involved in cheating will receive zero from all of the homeworks and the university regulations will be applied.
- **Newsgroup:** You must follow the ceng242 newsgroup on `news.ceng.metu.edu.tr` for discussions and possible updates on a daily basis.
- **Grading:** This homework will be graded out of 100.

6 Submission

Submission will be done via COW. Create a tar.gz file named `hw6.tar.gz` that contains a single file named `hw6.pl` which has your source code. Your code should be able to run correctly using the following command sequence.

```
$ tar xfvz hw6.tar.gz
$ prolog
| ?- consult('hw6.pl').
| ?- ktau([1,2], [1,2], T).

T = 1.0 ?

yes
| ?-

...
```