CSE 5311 Lab Assignment 1

Due July 12, 2004

Goals:

1. Understanding (and evaluation) of self-organizing list techniques.

2. Understanding of Markov chains and iterative methods for determining stationary distributions.

3. Understanding of ranking and unranking for permutations.

Requirements:

1. Write a C or C++ program to evaluate the move-to-front and transpose techniques for lists with 2-8 elements under uniform and Zipf distributions. The input is a single line with the following values:

   a. n - the number of list elements.
   b. strategy - 0 = move-to-front, 1 = transpose.
   c. distribution of request probabilities for the n elements - 0 = uniform $\left(P_i = \frac{1}{n}\right)$, 1 = Zipf $\left(P_{i-1} = \frac{1}{iH_n}\right)$.
   d. iterations - maximum number of iterations for the iterative solver.
   e. epsilon - threshold for terminating the iterative solver. If every value in the stationary distribution changes by no more than epsilon in a given iteration, the iterative solver should terminate. (`1e-8` is typical assuming `double`s are used.)

   Every case should have the following outputs:

   a. The actual number of iterations used by the iterative solver.
   b. The overall expected number of probes.
   c. The expected number of probes for each element.
   d. If $n \leq 4$, then provide the stationary probability for each list permutation.

2. E-mail your program to `ozcan@cse.uta.edu` before 2:45 pm on July 12.

Getting Started:

1. Review Notes 4, especially the Markov models for n = 2 and n = 3 for move-to-front.

2. When implementing code for the systems of equations, it is useful to have a bijection between permutations of n elements and the values 0 . . . n! - 1. Mapping from a permutation to an integer is known as *ranking*, while the inverse mapping is known as *unranking*. There are many resources available for this concept, including pages 29-35 of `http://reptar.uta.edu/NOTES4351/02notes.pdf` that gives *lexicographic* ranking/unranking code. *Be sure to give credit for any code that you use*.

3. Notes 4 cites the usual approach for solving for the probability of the list being in each configuration (known as the stationary distribution) by replacing one of the equations with 1 = sum of all probabilities for the configurations and then applying a general method such as Gaussian elimination, LU decomposition, or Householder reduction.

   Since Markov models are usually *sparse*, iterative methods are convenient and fast:

   a. The system of equations is constructed for the particular technique, e.g. move-to-front or transpose, and references the ranks of the configurations that are possible predecessors for each configuration.
   b. Conceptually, there are two tables of n! probabilities each. One is the *old* values from the previous iteration and the other is the *new* values from the current iteration. During each iteration, each equation is evaluated exactly once.
   c. For performing the first iteration, the *old* values are set to arbitrary probabilities that sum to exactly 1.

This implementation may be viewed as being a row-oriented or predecessor-oriented iterative method. It is also possible to have a column-oriented or successor-oriented iterative method that substitutes an *old* value in all relevant equations at the same time.

4. Tables may be preallocated for the maximum value of n (8).

5. The following example demonstrates the convergence. Such tracing is useful for debugging, *but should be disabled in the version that you submit*.

```
Enter n strategy (MTF/trans) dist (uni/Zipf) iterations epsilon
3 1 1 10 1e-2
Transpose
Zipf
By perms:
P012<-p0*P012+p0*P102+p1*P021
P021<-p0*P021+p0*P201+p2*P012
P102<-p1*P102+p1*P012+p0*P120
P120<-p1*P120+p1*P210+p2*P102
P201<-p2*P201+p2*P021+p0*P210
P210<-p2*P210+p2*P120+p1*P201
By ranks:
P0<-p0*P0+p0*P2+p1*P1
P1<-p0*P1+p0*P4+p2*P0
P2<-p1*P2+p1*P0+p0*P3
P3<-p1*P3+p1*P5+p2*P2
P4<-p2*P4+p2*P1+p0*P5
P5<-p2*P5+p2*P3+p1*P4
Start of iteration 0
012: 0.166667
021: 0.166667
102: 0.166667
120: 0.166667
201: 0.166667
210: 0.166667
Start of iteration 1
012: 0.227273
021: 0.212121
102: 0.181818
120: 0.121212
201: 0.151515
210: 0.106061
Start of iteration 2
012: 0.280992
021: 0.239669
102: 0.177686
120: 0.095041
201: 0.123967
210: 0.082645
Start of iteration 3
012: 0.315552
021: 0.249437
102: 0.176935
120: 0.080766
201: 0.111195
210: 0.066116
Start of iteration 4
012: 0.336657
021: 0.254081
102: 0.178369
120: 0.072229
201: 0.101632
210: 0.057032
Start of iteration 5
012: 0.350218
021: 0.255236
102: 0.179859
120: 0.067684
201: 0.095783
210: 0.051220

Used 6 iterations
0.358743: 012
0.255141: 021
0.181485: 102
0.065130: 120
0.091760: 201
0.047742: 210
Expected probes is 1.826930
Element 0 expected probes 1.498987
Element 1 expected probes 2.100286
Element 2 expected probes 2.400727
```