STA 247 — Assignment #3, Due Wednesday, December 8

Please submit your assignment on 8 1/2 *by* 11 *inch paper, stapled in the upper-left corner. Do* ***not*** *put it in an envelope or folder. Put your name, student number, and lecture section (Day or Evening) on the first page. You can hand the assignment in at the start of the Wednesday day section lecture, or in SS 6016A by 6pm.*

Late assignments will be accepted only with a valid medical or other excuse.

This assignment is to be done by each student individually.

A hard disk drive with a single platter contains a circular disk on which data is recorded. The disk is divided into a number of *tracks*, with each track being a circular region (ie, the part of the disk at a certain distance from the centre). The *head* of the disk can be positioned over any of these tracks. As the disk rotates, the head will be able to read any of the data on the track where it is positioned. To read data on another track, the head has to move to that track. This is called a *seek*. The time needed to seek is often a substantial part of the total time required for disk accesses. People have therefore devised various strategies to try to minimize this seek time. In this assignment, you will evaluate two such strategies, both in terms of their average performance, and in terms of the amount of variation in performance that they lead to.

In detail, we'll assume that the hard disk has 20 tracks, that it takes 5ms to read any amount of data off one track, once the head is positioned at that track, and that it takes 2ms to seek from one track to an adjacent track. So, for instance, if the head is positioned at track 3, it would take 25ms to seek to track 13 and then read data from that track — 20ms to move the head by 10 tracks, and 5ms to then read the data. (Real hard disks are more complicated than this, but this is complicated enough for this assignment.)

To evaluate how well a scheme for scheduling disk accesses works, we need a model of how processes on the computer access the disk. For this assignment, we will assume that there are 5 processes running, and that they all read the disk constantly — doing very little computation after one access before asking for another disk access to be done. We'll also assume that each process tends to access the same track as it previously accessed, as would tend to happen if a process reads many blocks of data from a single file before switching to reading another file.

In detail, let $A_{i,1}$, $A_{i,2}$, $A_{i,3}$, etc. be the successive numbers (1 to 20) of the tracks from which process i wishes to read data (where i is 1, 2, 3, 4, or 5). We will model this sequence of track accesses as a Markov chain, in which $A_{i,j+1}$ is the same as $A_{i,j}$ with probability p_i, and is otherwise equally likely to be any of the tracks (including the same track as before). In other words, the transition probabilities for this Markov chain are given by

$$P(A_{i,j+1} = b \mid A_{i,j} = a) \;=\; \begin{cases} p_i \;+\; (1-p_i)/20 & \text{if } a = b \\ (1-p_i)/20 & \text{if } a \neq b \end{cases}$$

To start, $A_{i,1}$ is equally likely to be any of the tracks. We will assume that the numbers p_i vary from process to process, and model this by letting $p_i = i/6$, so that they vary from 1/6 to 5/6.

Since the processes do very little computing between disk accesses, at any time, all five processes will be asking to access some track of the disk. Note that it is possible that more than one process will want to access the same track. Also, at any time, the disk head will be positioned at some track. The operating system needs some scheduling scheme, according to which it can decide which track to move the disk head to next. We will consider two schemes.

Using the *closest track* scheme, the operating system decides to move the head to the track closest to the track where the head is now which at least one process wants to access. For instance, if the five processes currently want to access tracks 3, 19, 3, 8, 12, and the head is currently at track 6, the decision will be to move the head to track 8, since that is the closest track that one or

more processes want to access. If there are two equally close tracks (eg, if the head were at track 10 in the above situation), the choice is made randomly, with equal probabilities. If at least one process wants to read from the track where the head is currently positioned, the head isn't moved at all. This includes the situation where a process wants to read more data from the same track that it just read data from.

Using the *elevator* scheme, the operating system schedules disk accesses by moving from one side of the disk to the other, then back again, etc. While moving to larger track numbers, the operating system moves the head to the closest track with a bigger number that at least one process wants to access. If there is no such track, the direction is reversed, and the operating system starts moving to lower numbered tracks, again until there are no lower numbered tracks that any process wants to read from. Note that with the elevator scheme, the head will always be moved (ie, data is never read from the same track that was just read from), *except* when the direction reverses, in which case the same track can be accessed again, if any process wants data from it.

You should write R functions to simulate these two scheduling schemes. You should start with the head positioned at a track chosen at random. For the elevator scheme, you should also set its current direction of movement randomly. You also begin the simulation with each of the five processes trying to access a randomly chosen track. After deciding which track to move to (according to whichever scheme you are simulating), you should figure out how much time this will take, and add this to a total time counter. You then figure out which processes have had there current access satisfied from this track, and for these processes decide which track they will access next, using the transition probabilities from above. You will then will need to update various statistics, after which you decide which track to move to next, etc.

You should continue the simulation until you have read data from some track 2000 times. You should return a list containing the following fields:

`time`	The total time (ms) taken for the 2000 accesses
`track`	A vector of length 2000 holding the track number of each access
`satisfied`	A vector of length 5 holding the number of requests satisfied for each process
`wait`	A vector (of indefinite length) holding the times from when each access was requested to when it was satisfied

The last vector will be at least 2000 long, but may be longer, since the total number of access requests that are satisfied can be greater than the number of tracks from which data was read (because two processes may request data from the same track).

To see how how efficiently data is being transferred from the disk, we can look at two numbers obtained from the simulation results. The rate at which requests to read data are satisfied can be estimated by dividing the sum of all the elements of the `satisfied` field by the `time` field. A related number is the average time waiting time, which you can estimate from the mean of the `waiting` field. You should try to get an idea of how accurate these estimates are by running each of the simulations at least five times.

In some situations (eg, when responding to user input), it is more important to know how long a process may need to wait until it gets the data it needs from disk. You can investigate this by looking at the histogram of numbers in the `wait` field.

You should hand in a listing of your functions, formatted in a readable way, with appropriate comments. You should also hand in a plot of the first 200 values from the `track` field returned by your simulations of each scheme, two histograms of times from the `wait` field, and any other plots that you think are interesting. Finally, you should hand in your estimates for how efficiently data is transferred using each scheme, along with a brief discussion of what you learned from these estimates and from the plots.

A list of some features of R that may be useful will soon be on the web page.