Class: CS304, "Problem Seminar." Meets 2:45–4:00 Tuesdays and Thursdays in room 301, Margaret Jacks Hall.

Discussion leader: Don Knuth. Office is 328 Jacks, phone 723-4367. [Please see Phyllis Winkler, 326 Jacks, if you want to talk to him outside of normal class hours.]

Teaching assistant: Ken Ross. Office is 450 Jacks, phone 723-3088; computer address `KAR @ POLYA`. Office hours on Tuesdays, 10:00–11:00, and Fridays, 11:15–12:15.

Problems: There are five problems and we will take them in order, spending about two weeks on each. Also, there will be a special two-day Trivia Hunt. Students should work in teams of two or three on each problem, and also when participating in the Trivia Hunt. No two students should be members of the same team more than twice; this way everybody will get to know almost everybody else. We stress cooperation and camaraderie, not concealment and competition! (Exception: The Trivia Hunt will be a contest to see which team can score the most points.)

Computer use: You may use any computer you can steal time on. Problem 5 may be done in the Macintosh Lab in Sweet Hall.

Grading: You should hand in a well-documented listing of your computer programs for each problem, along with a writeup that describes the approaches you took. This writeup should include a discussion of what you did that worked or didn't work, and (when appropriate) it should also mention what you think would be promising approaches to take if there were extra time to pursue things further. Your written work will be graded on the basis of style, clarity, and originality, as well as on program organization, appropriateness of algorithms, efficiency, and correctness of the results. These grades will be given on an A–E scale for your own information, but your overall grade for the course will be either ‘A’ or ‘nothing’.

Class notes: Classroom discussions will mostly involve the homework problems, but we will try to emphasize general principles of problem solving that are illustrated by our work on the specific problems that come up. Everyone is encouraged to participate in these discussions, except that nobody but Knuth will be allowed to talk more than three (3) times per class period. After class, the TA will prepare notes about what happened; therefore you will be able to participate freely in the discussions instead of worrying about your own note-taking. These class notes will eventually be published as a Stanford report; similar reports from previous years can be found in the library [CS606 (Michael J. Clancy, 1977); CS707 (Chris Van Wyk, 1979); CS863 (Allan A. Miller, 1981); CS989 (Joseph S. Weening, 1982); CS990 (John D. Hobby, 1983); CS1055 (Ramsey W. Haddad, 1985); CS1154 (Tomas G. Rokicki, 1987).

Special dates: The Trivia Hunt will begin at 4:00pm on Tuesday, January 24, and it will end at 2:45pm on Thursday, January 26. The Great Races for Problem 5 will be held on March 21 (possibly in Sweet Hall).

Caveat: This course involves more work than most other 3-unit courses at Stanford.

Problem 1, due January 24: **Late-binding solitaire.**

This problem is based on a game of solitaire that can be habit-forming: Shuffle a deck and deal out 18 cards, $c_1\,c_2\ldots c_{18}$. Then try to reduce these 18 piles to a single pile, using a sequence of “captures” in which one pile is placed on top of another pile. A pile can capture only the pile to its immediate left, or the pile obtained by skipping left over two other piles. Furthermore a capture is permitted only if the top card in the capturing pile has the same suit or the same rank as the top card in the captured pile.

For example, consider the following deal:

A♠ 7♠ 3♠ 4♣ 10♣ 9♡ 4♡ Q♣ 3♣ 10♡ A♡ J♣ 6♡ 8♣ 9♣ Q♡ K♠ K♡.

There are fifteen possible captures, which can be denoted by

7♠ × A♠	Q♣ ×× 10♣	6♡ ×× 10♡
3♠ × 7♠	3♣ × Q♣	9♣ ×× J♣
10♣ × 4♣	10♡ ×× 4♡	9♣ × 8♣
4♡ ×× 4♣	A♡ × 10♡	Q♡ ×× 6♡
4♡ × 9♡	J♣ ×× 3♣	K♡ ×× K♠

Some captures make others possible. For example, the sequence 9♣ ×× J♣ ×× 3♣ × Q♣ ×× 10♣ can be followed by 9♡ × 9♣.

If the leftmost capture is performed repeatedly in this example, the sequence of captures is 7♠ × *A*♠, 3♠ × 7♠, 10♣ × 4♣, 4♡ × 9♡, 3♣ ×× 10♣ × 3♠, *A*♡ ×× 4♡, 9♣ ×× *J*♣,

$K\heartsuit \times K\spadesuit \times\times 6\heartsuit$. This "greedy algorithm" corresponds to a standard solitaire game called *Skip Two*. But it leaves a configuration of 8 piles

$$3\clubsuit\ \mathrm{A}\heartsuit\ \mathrm{Q}\clubsuit\ 10\heartsuit\ 9\clubsuit\ \mathrm{K}\heartsuit\ 8\clubsuit\ \mathrm{Q}\heartsuit$$

in which no further moves are possible. If we hold back until all 18 cards are dealt in this example, we can find a sequence of 17 captures that reduces everything to a single pile.

Well, actually, a good Skip Two player would have captured in a different order when the $3\clubsuit$ was played; after $3\clubsuit \times Q\clubsuit \times\times 3\spadesuit$ and $10\clubsuit \times 3\clubsuit$, the $10\heartsuit$ would be able to make a double capture. But there still would have been 5 piles left after all 18 of the cards had been dealt.

[This game is an interesting way to waste time if you ever get lost with a pack of cards on a desert island. If you succeed in reducing the original 18 piles to a single pile, you can continue by dealing 17 more cards and trying to reduce the new 18 piles. And if you succeed also at that, you have 17 more cards left for a third try, since $52 = 18 + 17 + 17$. Three consecutive wins is a Grand Slam.]

The object of Problem 1 is to try to discover something about the chances of winning the game when 18 cards are dealt at random. Write a computer program that determines whether victory is possible or not, given a starting configuration. Speed will be important, as it will be desirable to run your program on as many different deals as possible in order to estimate the probability of success; computer time is limited.

Problem 2, due February 7: **Toetjes II.**

This problem was suggested by Sape Mullender, who described it as follows:

> In Amsterdam, where I grew up, dessert is usually referred to as "toetje" (Dutch for "afters"). The problem of allocating a left-over toetje to one of the children in my family became the Toetjes Problem. The algorithm was the following: First my mother would choose a secret number between one and a hundred. Then the children, in turn, youngest to oldest, could try to guess the number. After the last guess my mother would tell whose guess was closest to her secret number and the winner would get the toetje.
>
> It quickly became clear to us kids that a clever strategy for choosing the number would help to increase one's chances of winning. In a family of two children, for instance, the first child would have to choose the middle number, 50, and the other child could then choose either 51 or 49. Years later, when our reasoning skills were more developed we could even do the optimal choice for three kids. (Naturally, we assumed that each child would attempt to maximize his or her chances without resorting to conspiracy with one of the others.) The first child had to choose 25 (or 75), the second 75 (or 25), and the third could choose any number between 26 and 74, influencing the other two kids' chances, but not his or her own.
>
> Now that I have a degree in mathematics, the problem still puzzles me: Given that the secret number is chosen randomly from the interval $[0, 1]$, what is the optimal strategy for choosing a number for the ith child in a family of n

> children? The ith child knows what the first $i-1$ children chose, and knows that all the children choose optimally (*i.e.*, choose to maximize their own chance without consideration for the chances of any other child in particular).
>
> I grew up in a family of five children, and I never worked out the optimal strategy for $n=5$. Fortunately, I was the oldest, so choosing optimally was easy for me. But I don't think it mattered very much what I chose; I think my mother cheated. I think she chose the number after we had all announced our guesses, because I can't remember anyone ever winning two times in succession.

CS304 students investigated the Toetjes Problem two years ago [1]; unfortunately, the problem turned out to be much harder than it looked at first glance. We weren't even able to find a good formulation for the infinite case where real numbers, not integers, are to be chosen. But Tomás Feder made considerable progress after the course was over; he wrote a beautiful paper about Toetjes [2], solving several variations of the problem, although he left some fundamental questions unresolved. We will try to obtain further information by determining the optimum solution for as many n as possible when the following additional rule is added to Mullender's incomplete formulation:

> A player who has more than one optimum move must choose between them uniformly at random.

An optimum move is one that maximizes the player's expectation of winning the toetje, in a sense that will be further clarified below

The basic theory in [2] gives us a handle on this problem; we can formulate it as follows: Given a sequence of $k \ge 0$ distinct real numbers $x_1, \ldots, x_k \in [0\,..\,1]$, a *completion* of these numbers is a tuple $\langle x_{k+1}, \ldots, x_n; \pi\rangle$ where the x's are arbitrary elements of $[0\,..\,1]$ and where $\pi = \pi_1\pi_2\ldots\pi_n$ is a permutation of $\{1, 2, \ldots, n\}$ with the property that

$$x_{\pi_1} \le x_{\pi_2} \le \cdots \le x_{\pi_n}\,.$$

For example, if $k=2$ and $n=6$, one of the completions of $\langle\frac{1}{3}, \frac{2}{3}\rangle$ is $\langle\frac{2}{3}, \frac{4}{5}, \frac{1}{3}, \frac{2}{3}; 512634\rangle$. A completion represents a potential "limiting play" in the game of Toetjes. For example, if there are six players and if the first two plays are $\frac{1}{3}$ and $\frac{2}{3}$, this completion indicates that player 3 chooses a number slightly more than $\frac{2}{3}$, then player 4 chooses $\frac{4}{5}$, then player 5 chooses a number slightly less than $\frac{1}{3}$, and player 6 chooses a number between $\frac{2}{3}$ and the choice of player 3.

Let $\langle x_{k+1}, \ldots, x_n; \pi\rangle$ be a completion of $\langle x_1, \ldots, x_k\rangle$. We define the *payoff function* $p_j(x_1, \ldots, x_n; \pi)$ as follows:

$$p_j(x_1, \ldots, x_n; \pi) = \begin{cases} (x_j + x_{\pi_2})/2, & \text{if } j = \pi_1; \\ (x_{\pi_{l+1}} - x_{\pi_{l-1}})/2, & \text{if } j = \pi_l,\ 1 < l < n; \\ 1 - (x_j + x_{\pi_{n-1}})/2, & \text{if } j = \pi_n. \end{cases}$$

For example, the respective values of $p_j(\frac{1}{3}, \frac{2}{3}, \frac{2}{3}, \frac{4}{5}, \frac{1}{3}, \frac{2}{3}; 512634)$ as j varies from 1 to 6 are $\frac{1}{6}, \frac{1}{6}, \frac{1}{15}, \frac{4}{15}, \frac{1}{3}, 0$. It's easy to check that $p_j(x_1, \ldots, x_n; \pi)$ is the limit of the probability that player j wins the toetje, if a limiting play corresponding to $\langle x_{k+1}, \ldots, x_n; \pi\rangle$ is performed.

The players in this game are required to make only "optimum" moves, but there usually *isn't* any optimum move! For example, if $n = 2$ and if the first player chooses $\frac{1}{2}$, the second player should choose a number as close as possible to $\frac{1}{2}$; but there is no "closest possible" choice. Instead, we require the players to choose *optimum near-plays* when the optimum payoff is obtainable only as a limit. For example, if $n = 2$ and if the first player chooses a number $x_1 = x < \frac{1}{2}$, the second player's optimum strategy is to choose the number $x + \epsilon$, where ϵ is an arbitrarily small positive value. This leads to the continuation $\langle x; 12\rangle$, and the second player's limiting payoff will be $p_2(x, x; 12) = 1 - x$. If $x_1 = x > \frac{1}{2}$, the second player should choose $x - \epsilon$, getting payoff $p_2(x, x; 21) = x$. And if $x_1 = \frac{1}{2}$, the second player has two optimum choices, namely $\frac{1}{2} - \epsilon$ and $\frac{1}{2} + \epsilon$; a random selection should be made between these two choices, with probability 50% of going either way. The payoff will be $\frac{1}{2}p_2(\frac{1}{2}, \frac{1}{2}; 21) + \frac{1}{2}p_2(\frac{1}{2}, \frac{1}{2}; 12) = \frac{1}{2}$.

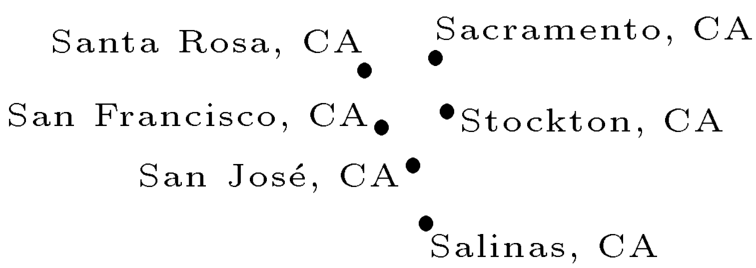
In this problem we wish to determine the optimum strategy for each player $k + 1$, given each sequence of opening moves $\langle x_1, \ldots, x_k\rangle$, for $0 \le k < n$ and for n as large as we can handle. The optimum strategy in each case will be a random variable $f(x_1, \ldots, x_k)$ whose value is either a finite set of points (perhaps including $+\epsilon$ or $-\epsilon$), or a finite set of open intervals. In the first case x_{k+1} should be selected from the given points, with equal probability; in the second case x_{k+1} should be uniformly distributed in the intervals. For example, the optimum strategies for $n = 3$ are tabulated in Appendix B.

Problem 3, due February 21: **Label placement.**

Consider n points on a page and n corresponding boxes of text. We want to place the boxes of text so that (a) no two boxes overlap; and (b) the boxes don't get closer than ϵ to any of the points, where ϵ is a given radius. Furthermore, subject to these conditions, we want each box to be as close as possible to its corresponding point, and as far as possible from all other points, in some loosely defined sense; then a viewer will know which box goes with which point.

The data set for this problem consists of 128 city names and locations, defined by three integer coordinates (w, x, y). The name of a city fits in a box of width w and height 50; the city is located at point (x, y). For example, San Francisco has $(w, x, y) = (634, 70, 1659)$. The value of ϵ is 20.

When coordinates for the lower left corners of the city names have been specified, TeX will be able to typeset a map of the United States and Canada, where the vicinity of San Francisco might look like this:



(The data for all 128 cities appears in Appendix C.)

Problem 4, due March 7: **AND-OR tree formulas.**

An AND-OR tree of order k is an arrangement of $2^k - 1$ gates in a complete binary tree. The root of the tree is either AND or OR; the two gates leading into each AND are ORs and vice-versa. The leaves of the tree are either variables x_i or the constants 0 or 1.

For example, here's an AND-OR tree of order 3:

```
                         |
                     — OR —
                   /         \
             AND                   AND
            /    \                /    \
         OR        OR          OR        OR
        /  \      /  \        /  \      /  \
      x1    x2  x3    x4    x1    x3  x2    x4
```

This function turns out to be equal to the "threshold function" $\theta_2(x_1, x_2, x_3, x_4)$, where

$$\theta_m(x_1, \ldots, x_n) = \begin{cases} 1, & \text{if } x_1 + \cdots + x_n \geq m; \\ 0, & \text{if } x_1 + \cdots + x_n < m. \end{cases}$$

If we interchange AND and OR the resulting function is $\theta_3(x_1, x_2, x_3, x_4)$.

Leslie Valiant [*Journal of Algorithms* **5** (1984), 363–366] has proved that, for all n, there is an AND-OR tree with $O(n^{5.3})$ gates for the function $\theta_{n/2}(x_1, \ldots, x_n)$. His method was to choose the 2^k leaves of a k-level tree at random, setting a leaf equal to x_i with probability p_i, to 0 with probability p_0, and to 1 with probability p_∞, where $p_0 + p_1 + \cdots + p_n + p_\infty = 1$. Given a random tree function $f(x_1, \ldots, x_n)$ chosen in this way, it's not difficult to compute the probability that $f(x_1, \ldots, x_n) = 1$, for any given $(x_1, \ldots, x_n)$. Valiant showed that probabilities can be chosen in such a way that, for every fixed $(x_1, \ldots, x_n)$, the probability that $f(x_1, \ldots, x_n) \neq \theta_{n/2}(x_1, \ldots, x_n)$ is less than 2^{-n}, if k is suitably large. Hence the probability that $f(x_1, \ldots, x_n) = \theta_{n/2}(x_1, \ldots, x_n)$ for all 2^n choices of $(x_1, \ldots, x_n)$ is > 0. Hence, an AND-OR tree must exist!

Valiant did not determine the constant of proportionality in his formula $O(n^{5.3})$. Since 2^n is less than $n^{5.3}$ when $n < 25$, it would be nice to know whether Valiant's method is likely to be of any practical use.

The purpose of this problem is to find the sharpest results obtainable by Valiant's method. Let

$$\theta_{a,b}(x_1, \ldots, x_n) = \begin{cases} 1, & \text{if } x_1 + \cdots + x_n \geq b; \\ 0, & \text{if } x_1 + \cdots + x_n < a. \end{cases}$$

(This is a partially unspecified function; we don't care about the values of $\theta_{a,b}$ when $a \leq x_1 + \cdots + x_n < b$.) Given values of k and n, our goal will be to find probabilities p_i such that a random AND-OR tree of order k has positive probability of being $\theta_{a,b}(x_1, \ldots, x_n)$, where $a \leq n/2 \leq b$ and $b - a$ is as small as possible.

Write a computer program to determine the best values of a and b, given k and n. Then try to find the best possible constants c and α such that Valiant's method can be used to prove the existence of a formula for $\theta_{n/2}(x_1, \ldots, x_n)$ using at most cn^α gates.

Also try by heuristic means to construct economical circuits for $\theta_{n/2}(x_1,\ldots,x_n)$ built entirely from AND and OR gates, when n isn't too large.

Notes: It can be shown that

$$\theta_4(x_1,\ldots,x_8) = f(1234|5678) \vee f(1256|3478) \vee f(1278|3456)$$

where $f(a_1a_2a_3a_4|b_1b_2b_3b_4) = \theta_2(x_{a_1},x_{a_2},x_{a_3},x_{a_4}) \wedge \theta_2(x_{b_1},x_{b_2},x_{b_3},x_{b_4})$. Since the function $\theta_2(x_1,x_2,x_3,x_4)$ can be computed with 7 gates, $\theta_4(x_1,\ldots,x_8)$ can be computed with 47 gates. This construction doesn't seem to generalize to an efficient formula for $\theta_{n/2}(x_1,\ldots,x_n)$ when n is large; the best upper bound known is Valiant's $O(n^{5.3})$. Pippenger, Paterson, and Peterson have shown that formulas of length $O(n^{3.4})$ are possible if NOT gates are allowed as well. Knuth's best formula for $\theta_8(x_1,\ldots,x_{16})$ has 767 gates. On the other hand, if circuits are allowed to have multiple fanout, i.e., with outputs used repeatedly as inputs to other gates, constructions with $O(n^2)$ gates are not difficult to find.

Problem 5, due March 21: **Antomology.**

The purpose of this final problem is to design the brain of an ant. A colony of ants sharing your design will be released into an environment that contains particles of food; their task will be to locate the food and bring it to their nest as soon as possible.

We're not dealing with real live ants here; these are artificial, discrete, synchronous, two-dimensional, low-tech ants that can more properly be called *antomata*. An antomaton is a finite state device that can move, carry, sniff, and leave scents on a cellular grid according to rules specified below. The problem is to design it so that your team's colony of antomata will fetch the food fastest, during the Great Ant Races of March 21, 1989. On that day your design will be tested in challenging environments that won't be revealed beforehand. Each race will feature ants of a single design, whose performance will be measured by a simulated clock; there will be no direct confrontations between antomata of different species.

An antomaton's environment is an $m \times n$ grid surrounded by barrier cells. The values of m and n may be as large as 100. Some interior cells of the grid may also be barriers, representing obstacles that the ants must avoid. At least one interior cell is a *nest*, from which an unlimited number of ants may emerge and into which an unlimited number of ants may retire.

Cells that are neither barrier nor nest contain at most one ant each. Such cells may also contain morsels of food, possibly several morsels in a single cell. Moreover, there are four *scents* called S, T, U, V, each of which may or may not be present in a given non-barrier, non-nest cell.

Ants have four orientations called *up*, *down*, *left*, and *right*. (We think of the environment as an upright computer screen.) They can move at most one step at a time, according to this orientation; they can also sense whether the cell they're facing is a barrier or a nest, or whether it's occupied by an ant and/or food. And they can smell any scents that may be present in the cell they face.

Each nest cell has a "top ant," which has the same capabilities as ants in ordinary cells. When the top ant moves out of the nest, another top ant materializes there, initially facing upward. Ants carry no food when they leave the nest, but they acquire a morsel of food as soon as they enter a cell where food is present. From then on they carry this morsel with them, until they move into a nest cell—at which time they effectively disappear.

The ant's brain is a computer program in a special machine language. The program consists of up to 1024 instructions, each 32 bits long. Every ant in the system has a program location l representing its current "state of mind"; the ant's behavior at a given time depends on l and on what the ant can see or smell.

Here's how the mechanism works: The first 8 bits of each instruction denote values of sensory inputs, and the next 8 bits denote a mask. The instruction *applies* to a given situation if the ant's actual sensory inputs match those of the instruction wherever a 1 occurs in the mask. These 8 bits have the following significance, from left to right:

a: 1 if the cell ahead contains another ant.
b: 1 if the cell ahead is a barrier, or if it contains an ant bearing food.
f: 1 if the cell ahead contains food (in addition to what an ant may be carrying).
r: a random bit, which is 1 with probability $\frac{1}{2}$. These bits are generated independently for every ant at every unit of time.
s, t, u, v: 1 if the cell ahead contains the scents S, T, U, V, respectively.

The three bits abf will never be 011, but the other seven combinations of these bits are all possible. These bits will be respectively 101 if and only if the cell ahead is a nest cell. (A nest cell always contains a top ant that's not bearing food; a nest also has food hidden below. A non-nest cell with an ant and food is possible only if that ant is carrying additional food, in which case $abf = 111$.)

The last 16 bits of an instruction define the ant's actions when an applicable instruction is found. There are 6 action bits followed by 10 bits of 'next address'. The action bits are

ds, dt, du, dv: 1 if the current occurrence of scents S, T, U, or V is to be complemented, respectively.
m: turn 90° counterclockwise.
p: turn 90° clockwise.

For example, if an ant is facing left in a cell containing the scents T and U but not S or V, and if the six action bits are 110010, the ant will begin the next cycle facing down, in a cell containing the scents S and U.

If $mp = 11$, the ant moves ahead instead of turning (because its legs on both sides go into action). In this case the $ds \ldots dv$ bits complement the scents of the new cell into which the ant now moves; the old cell retains its former odors.

Moves are slightly complicated because there are three situations in which an instruction can have $mp = 11$ but the ant will stay where it was: (1) If the cell ahead is a barrier. (2) If the cell ahead is not a nest and it contains an ant. (3) If an ant of higher priority is moving to the same cell at the same moment. (Priority is determined from top to bottom and from left to right within a row.) In these three cases the move *fails*; the ant stays put, and no scents are complemented. Otherwise the move succeeds.

All actions for all ants are performed simultaneously. When an ant's state of mind is determined by location l, the environment simulator decides what to do by computing the relevant sensory bits $abfrstuv$ and then finding the first applicable instruction that is present in locations $\geq l$. (Applicability is based on the first 16 bits of the instructions, as explained earlier.) Then the specified action is performed (based on the next 6 bits), and l is reset to the binary value of the last 10 instruction bits. Exception: If the action was a move that failed, for one of the three reasons in the previous paragraph, the new value of l is *one less* than the address in the last 10 bits.

All cells are initially scent-free, and all ants are initially confined to the nests. The initial value of l, when an ant first appears on the scene by becoming a new top ant (either when the race starts or when the old top ant has left a nest), is 0. We assume by convention that locations -1 and 1024 both contain the all-zero instruction $00\ldots0$; this instruction applies to all sensory inputs (since its mask is zero), hence it denotes a unconditional jump to location 0.

A simulator will be provided for testing your constructions and for administering the Great Races. An ant brain must be specified to the simulator in an ASCII file that adheres to the following rigid format: (1) The first line of the file must contain the name of this ant colony, for display purposes; the name can be up to 12 characters long. (2) The second line of the file must contain 16 characters to be used as symbolic screen-display representations of the 16 possible combinations of scents, in "binary" order $\overline{STUV}$, $\overline{STU}V$, $\overline{ST}U\overline{V}$, …, $STUV$. (3) The next lines of the file must contain the instructions for locations 0, 1, 2, …, one instruction per line, as a sequence of eight hexadecimal digits followed by optional comments. (4) The final line of the file must begin with '*'. All subsequent instructions in the ant's brain will be zero.

Here, for example, is a simple ant brain that uses only scent S:

```
      RANDY ANTY
      .???????S???????
 0    10100800 INIT half the time, turn left
 1    00E88C03      otherwise leave nest for an empty, scentless cell
 2    00000000      repeat forever
 3    00000004 GO   goto ALF
 4    10100808 ALF  half the time, turn left
 5    00000006      get a new random bit
 6    10100409 ALF1 half the time, turn right
 7    00000009      goto ALF3
 8    10100809 ALF2 half the time, turn left again
 9    00E88C04 ALF3 move to an empty cell and scent it
 A    08E80C04      move to an empty cell already scented
 B    20E00C0D      move to a cell with food in it
 C    00000004 ALF! go back to ALF
 D    0000000E BET- goto BET
 E    10100812 BET  half the time, turn left
 F    00000010      get a new random bit
10    10100413 BET1 half the time, turn right
11    00000013      goto BET3
12    10100813 BET2 half the time, turn left again
13    08C88C0E BET3 move to empty cell and remove scent
14    00000C0E      move if possible
      * END OF DATA
```

The idea here is to leave the nest (`INIT`) if there's an empty unscented cell to start on; then to make a random walk (`ALF`) while laying down scent, until finding food; then to make another random walk (`BET`) while erasing scent, until finding the nest. This brain design works—albeit slowly—in many environments; but it has a fatal flaw that's not too hard to rectify.

The simulator is also able to read ASCII files that define an environment. The first line of such a file contains two values '$m\,n$' that specify the number of rows and columns in the playing field. The next m lines of the file contain n characters each, where the allowable characters are

`B` barrier cell
`N` nest cell
`.` empty cell
`1` cell with one morsel of food
`2` cell with two morsels of food
… `9` cell with nine morsels of food

For example, here's a simple environment specification:

```
7 19
...................
.......BB.B...123..
BB.......B.B..244..
BB..BB.....B...2..B
...BBBN..BB.B....B.
....B..B..BB...BB..
......BB...........
```

(Barrier cells are also implicitly present at the top, bottom, and sides.) Further details of the simulator—including facilities for, ahem, debugging—will be available later.

The environments used in the Great Ant Races will include only one nest cell. Moreover, all of the food will appear in a "connected" region of cells. (At least this will be true in the initial races. Tie-breaking competitions may violate these conditions; the environments might even change dynamically.)

Appendix A: Data for Problem 1.

(Here 'T' means 'ten'; 'C' means 'clubs', etc.)

```
 1   AS7S3S4CTC9H4HQC3CTHAHJC6H8C9CQHKSKH
 2   2CQSAS6D6C8C4SJC7D3DQH9D9S3S2D6S7STH
 3   QD6H7HADKD4S3CKH9HJDTD4H8C5H5D2D5CAS
 4   TC2CTS5D9S9DAC4C6CKD8C9CKS4SJS8H5C9H
 5   9CQSTS8C4C5H4HTCQDAD9SKC5D8S5S3HAHJS
 6   4C5DQD2C7D2D6C4SAC4HQHKH6HTSJC6DTC3H
 7   3C8S5H7DTHKHQH4H7SQCJC2H5C8H6C3D2CKC
 8   4D4HTDTSKH8S2CTC9S8HTHJC6S9HAH6HQHAD
 9   6C8SJC8D9S2H6SKC4S7CTD3H5H9HKH7H6DTS
10   2HJHJSAD5D6CQD7H9S4H3C6HJC9CJDASQH8S
11   KH3C5HQD4SJC6H2DKD5D9C8C3D5C9DAD2C8H
12   5DQC2H8D3S9D8CAHACADQS9C4DJS7HKCTH6C
13   2SKS2D2HQC8C4DTSJH6SADACJD6H9H4H6DQD
14   7D4H4D8H8C8D3HTS2DKD2C7STD6C5D9S4S6D
15   JC6C9DQC9CASQD3C2D8STC7CJD7SKS5H4SQH
16   6DAD3D3C3H2C8C6S5S4S5C6H4C5HQCJC7SQS
17   AH4HAS9S7C8S4SACTDTC2D5HKC3HKDQDTSKS
18   2H5H3S4HTH4DAD6C7H7S2D9D8CQH9HQC4C9S
19   QH5STCQS4HAHKS5D4S4C6D2S7C3C2D7D3H2C
20   7HJC9HAHQC4HAC6H8S4SKC6DKD2DJS3S7CKS
21   ASQCJC2D8C9STH7SQD9D7C8H5D9H7HADJSTD
22   JHQS8S2C9C3HJD2HAHJC8D7SQD4CKSKCAD6C
23   QCJS7C2H4H9S4D5CQDTS5DAS7H8D8S6H3DAC
24   7S5D7C7D4SQC6S2H3CKHKDQH8CAD5C8SKC6D
```

Appendix B: The case $n = 3$ in Problem 2.

The optimum play or near-play $f(x_1, \ldots, x_k)$ is given either as a finite set or as a finite union of open intervals. We also list the random variable $g(x_1, \ldots, x_k)$ whose values are continuations $\langle x_{k+1}, \ldots, x_n; \pi\rangle$ that correspond to all optimum limiting plays; the possible values of $g(x_1, \ldots, x_k)$ are shown here as a sum or integral of continuations multiplied by the appropriate probabilities. For example, if $f(x_1, \ldots, x_k) = \{a + \epsilon, b\}$ then $g(x_1, \ldots, x_k) = \frac{1}{2}\langle a, \lim_{\epsilon\to 0} g(x_1, \ldots, x_k, a + \epsilon)\rangle + \frac{1}{2}\langle b, g(x_1, \ldots, x_k, b)\rangle$. The continuations $g(x_1, \ldots, x_k)$ can be used to calculate the payoffs, which are the expectations $E\bigl(p_j(x_1, \ldots, x_n; \pi)\bigr)$.

$k = 2$, $0 \le x_1 = x < x_2 = y \le 1$:

Case 1, $x + y > 1$ and $3x > y$: $f(x, y) = x - \epsilon$; $g(x, y) = \langle x; 312\rangle$.
Case 2, $3x \le y$ and $2 + x \le 3y$: $f(x, y) = (x \mathinner{\ldotp\ldotp} y)$; $g(x, y) = \int_x^y \langle t; 132\rangle\, dt/(y - x)$.
Case 3, $x + y < 1$ and $2 + x > 3y$: $f(x, y) = y + \epsilon$; $g(x, y) = \langle y; 123\rangle$.
Case 4, $x+y = 1$ and $x > \frac{1}{4}$: $f(x, y) = \{x-\epsilon, y+\epsilon\}$; $g(x, y) = \frac{1}{2}\langle x; 312\rangle + \frac{1}{2}\langle y; 123\rangle$.

$k = 1$, $0 \le x_1 = x \le \frac{1}{2}$:

Case 1, $x \le \frac{1}{4}$: $f(x) = \frac{2+x}{3}$; $g(x) = \int_x^{(2+x)/3} \langle \frac{2+x}{3}, t\rangle\, dt/(\frac{2+x}{3} - x)$.
Case 2, $\frac{1}{4} < x < \frac{1}{2}$: $f(x) = 1 - x + \epsilon$; $g(x) = \langle 1 - x, x; 312\rangle$.
Case 3, $x = \frac{1}{2}$: $f(x) = \{\frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon\}$; $g(x) = \frac{1}{2}\langle \frac{1}{2}, \frac{1}{2}; 213\rangle + \frac{1}{2}\langle \frac{1}{2}, \frac{1}{2}; 312\rangle$.

$k = 0$: $f() = \{\frac{1}{4}, \frac{3}{4}\}$; $g() = \frac{1}{2}\int_{1/4}^{3/4} \langle \frac{1}{4}, \frac{3}{4}, t; 132\rangle\, dt/(\frac{1}{2}) + \frac{1}{2}\int_{1/4}^{3/4} \langle \frac{3}{4}, \frac{1}{4}, t; 231\rangle\, dt/(\frac{1}{2})$.

The expected payoffs are $\frac{3}{8}$, $\frac{3}{8}$, $\frac{1}{4}$ for players 1, 2, 3.

Appendix C: Data for Problem 3.

```
Ravenna, OH (458,4188,2166)
Reading, PA (430,4719,2041)
Red Bluff, CA (492,88,2019)
Regina, SA (385,1847,3555)
Reno, NV (336,331,1920)
Rhinelander, WI (569,3370,2838)
Richfield, UT (466,1103,1807)
Richmond, IN (478,3823,1966)
Richmond, VA (498,4567,1623)
Roanoke, VA (445,4318,1582)
Rochester, MN (511,3066,2595)
Rochester, NY (499,4551,2466)
Rockford, IL (435,3402,2332)
Rock Springs, WY (635,1389,2230)
Rocky Mount, NC (623,4532,1383)
Roswell, NM (435,1859,1002)
Rutland, VT (440,5015,2533)
Sacramento, CA (558,163,1780)
Saginaw, MI (426,3918,2506)
Saint Augustine, FL (697,4180,475)
Saint Cloud, MN (583,2895,2827)
Saint Johnsbury, VT (715,5110,2655)
Saint Joseph, MI (583,3664,2307)
Saint Joseph, MO (613,2828,1957)
Saint Louis, MO (566,3293,1785)
Saint Paul, MN (534,3002,2734)
Salem, OR (368,9,2733)
Salida, CO (375,1712,1771)
Salina, KS (362,2551,1818)
Salinas, CA (403,147,1492)
Salisbury, MD (488,4752,1747)
Salt Lake City, UT (655,1124,2106)
San Angelo, TX (549,2268,711)
San Antonio, TX (586,2462,405)
San Bernardino, CA (695,581,1108)
San Diego, CA (508,597,898)
Sandusky, OH (484,4041,2209)
San Francisco, CA (634,70,1659)
San Jos\'e, CA (459,124,1593)
Santa Ana, CA (520,525,1056)
Santa Barbara, CA (658,342,1155)
Santa Fe, NM (474,1717,1344)
Santa Rosa, CA (547,40,1758)
Sarasota, FL (441,4059,93)
Sault Sainte Marie, MI (789,3877,2965)
Savannah, GA (490,4203,804)
Schenectady, NY (579,4917,2415)
Scottsbluff, NB (530,1946,2272)
Scranton, PA (455,4745,2203)
Seattle, WA (412,79,3132)
Sedalia, MO (423,2989,1798)
Selma, AL (357,3610,855)
Seminole, OK (472,2644,1276)
Sheridan, WY (483,1616,2712)
Sherman, TX (459,2651,1038)
Shreveport, LA (527,2937,868)
Sioux City, IA (494,2673,2365)
Sioux Falls, SD (528,2639,2523)
South Bend, IN (536,3687,2244)
Spokane, WA (456,571,3142)
Springfield, IL (498,3347,1962)
Springfield, MA (547,5053,2307)
Springfield, MO (549,2983,1575)
Springfield, OH (537,3931,1980)
Staunton, VA (465,4405,1714)
Sterling, CO (433,1990,2085)
Steubenville, OH (584,4250,2046)
Stevens Point, WI (624,3355,2670)
Stockton, CA (464,183,1686)
Stroudsburg, PA (570,4793,2140)
Sumter, SC (393,4277,1080)
Swainsboro, GA (553,4078,882)
Syracuse, NY (462,4697,2449)
Tacoma, WA (439,69,3078)
Tallahassee, FL (536,3884,559)
Tampa, FL (381,4067,184)
Terre Haute, IN (548,3571,1912)
Texarkana, TX (515,2907,1006)
Toledo, OH (394,3958,2239)
Topeka, KS (396,2745,1849)
Toronto, ON (435,4374,2539)
Traverse City, MI (605,3749,2706)
Trenton, NJ (416,4835,2026)
Trinidad, CO (459,1861,1567)
Tucson, AZ (394,1215,825)
Tulsa, OK (353,2721,1416)
Tupelo, MS (396,3441,1131)
Tuscaloosa, AL (526,3555,973)
Twin Falls, ID (500,865,2376)
Tyler, TX (345,2782,844)
Uniontown, PA (521,4339,1977)
Utica, NY (347,4789,2458)
Valdosta, GA (462,3984,616)
Valley City, ND (545,2511,3030)
Vancouver, BC (514,0,3382)
Vicksburg, MS (503,3224,844)
Victoria, TX (439,2611,313)
Vincennes, IN (480,3559,1794)
Waco, TX (345,2598,724)
Walla Walla, WA (591,479,2902)
Warren, PA (402,4398,2269)
Washington, DC (565,4609,1825)
Waterbury, CT (519,5007,2224)
Waterloo, IA (445,3078,2367)
Watertown, NY (536,4720,2589)
Watertown, SD (524,2601,2727)
Waukegan, IL (474,3529,2346)
Wausau, WI (421,3348,2736)
Waycross, GA (482,4077,675)
Weed, CA (345,73,2205)
Wenatchee, WA (540,280,3105)
West Palm Beach, FL (748,4307,0)
Wheeling, WV (500,4240,2002)
Wichita, KS (420,2578,1645)
Wichita Falls, TX (618,2463,1077)
Williamson, WV (567,4084,1644)
Williamsport, PA (601,4612,2179)
Williston, ND (478,1950,3214)
Wilmington, DE (563,4757,1954)
Wilmington, NC (565,4520,1128)
Winchester, VA (537,4496,1870)
Winnipeg, MB (500,2597,3474)
Winston-Salem, NC (679,4287,1407)
Wisconsin Dells, WI (698,3335,2536)
Worcester, MA (514,5132,2332)
Yakima, WA (430,261,2982)
Yankton, SD (438,2573,2424)
Youngstown, OH (578,4247,2157)
```