

Consider a forest made up of a square matrix of plots ( $L \times L$ ). Each plot is initially occupied by a tree with probability  $p$ . The first row of trees is set afire and we want to know whether or not the fire makes it all the way to the last row if it can only hop to nearest neighbors occupied by trees.

Your assignment is to:

1. Write a program that gets values for  $L$  and  $p$  from the user and draws the forest on the screen. (For each forest, you will want to assign *fixed* random numbers between 0 and 1 to each plot and then “plant trees” according to different possible values of  $p$ .)
2. Determine if the forest fire will make it to the bottom for any given value of  $p$ . You will need to write a function that will actually “set the forest fire” and then propagate it through the forest.
3. Determine the average minimum probabilities and the standard deviation of the minimum probabilities required for forests with  $L = 4, 8, 16, 32,$  and  $64$  to burn to the bottom. Do this by looking at at least  $10$  forests of each size.
4. What do you suppose happens in the limit  $L \rightarrow \infty$  ?

Here is a suggested flow for your main program:

```
void main(void)
{
    int new_fire;
    int direction;

    initialize_graphics();

    do
    {
        randomize();
        pick_size();                // pick the value of L
        initialize_values();        // assign values from 0 to 1 to each plot

        do
        {
            pick_prob();            // choose the probability p of a tree on each plot
            clear_cut();            // clear all plots graphically (paint them yellow)
            plant_trees();          // plant trees graphically (paint them green)
            light_first_row();      // set first row on fire (paint trees red)
            direction=1             // set first sweep direction

            do
            {
                new_fire=sweep(direction); // set new fires downwind (paint them red)
                direction=-direction;      // toggle "wind direction"

            } while(new_fire=='y');        // until there are no new fires

            print_results();            // print info on results
            ask_new_prob();            // want to try same forest, new probability?

        } while(new_prob!='n');

        ask_new_forest();            // want to try a new forest?

    } while(new_forest!='n');

    closegraph();
    return(1);
}
```