UT Dallas High School Programming Contest 10-28-2017

How to solve the problems

A. Sticks

Some sticks are cut into some unknown number of parts. How can we to combine then to form a set of sticks, all of the same length, such that they are of the smallest possible length?

For example, given lengths 1, 2, 3, 4, the smallest possible length for the original sticks is 5.

Note the following observations (from one of my students):

1. your candidate must be a factor of the sum of the input numbers
2. your candidate must be at least as large as the largest input number
3. a candidate x is not feasible unless you can create every multiple of x from 0 to sum.

Example: Consider the stick set 5 3 7 6 5 3 2 1 4 6: The sum is 42. Factors of 42 are 1, 2, 3, 6, 7, 14, 21.
Those at least as large as the largest input are 7, 14, 21, 42.

How many ways are there to make up 7 from the input data?
$7 = 7 = 6 + 1 = 5 + 2 = 4 + 3 = 4 + 2 + 1 = 3 + 3 + 1$.
Note that a 6 must be paired with a 1, but there are two 6s and only one 1. So 7 is infeasible.

Let's try 14.
$14 = 7 + 6 + 3 = 6 + 6 + 2 = 6 + 5 + 3 = 6 + 5 + 2 + 1 = 6 + 4 + 3 + 1 = 5 + 5 + 4 = 5 + 4 + 3 + 2 = 5 + 3 + 3 + 2 + 1$.
Eight ways. All input values appear in at least one sum. Next, see if we can combine a subset of three of these sums to make up 42 using each input number only once.

Number the ways 0 .. 7 and form an array of 8x8 booleans to indicate which pairs are mutually exclusive. For instance, $6 + 6 + 2$ and $6 + 5 + 3$ are mutually exclusive. Then use this array to limit the recursive search for a solution. We find $(7 + 6 + 1), (6 + 5 + 3), (5 + 4 + 3 + 2)$ Bingo! The solution is 14.

B. Digit sum

You are given number strings X,Y, and radix r.
Convert the two number strings X, Y, from radix, r, to binary, giving x and y . Then, for each value in the range [x,y], find its number of digits in radix r, and sum the results. There are, of course speedups, but they are not needed.

C. Knights Path

With the given board use Breadth-First-Search, trying all eight possible (valid) next moves of the knight at each level. You'll need a class to store the current position of the knight and its level number (how many moves it took to get there), an array to record which cells you have visited on previous, or the current, level(s) of BFS, and a queue for BFS. For each of the eight next-positions generated you'll need a function to check if it is within the edges of the board.

D. Land

You are given descriptions of parcels of land in a standard Land Management format. You have to find the total area and number of disjoint parcels of land.

Create an array of booleans for the entire region, that's 12x12 units (each of 40 acres). Parse the input strings and set the corresponding booleans to true while accumulating the number of booleans that you set. The total is the area of Julie's land in 40-acre units.

The tricky part is in finding the number of disjoint regions. I used a Union-Find (Disjoint Set) data structure. There is one set for each of the 144 units of land. These are represented by an array of ints, all initially set to -1. Set the elements corresponding to units that Julie has inherited to -2, meaning that they are the root-nodes of sets of size 1. Then use Union-Find in the usual way by searching the boolean array and calling Union for each pair of neighboring true cells. At the end the number of sets is the number of disjoint parcels of land.

You can also use *Flood Fill* to find the disjoint sets.

E. Polygons

The simplest way to solve this problem is to use the "Shoelace Theorem":

Given any simple polygon (no holes, edges do not cross each other) traverse its perimeter in clockwise or counter clockwise order placing the vertex coordinates into a column matrix. Repeat the start point at the end:

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \dots & \\ x_n & y_n \\ x_1 & y_1 \end{bmatrix}$$

Then form products and sum them as follows:

$$\text{Area } = \frac{1}{2} \left| (x_1 y_2 + x_2 y_3 + \dots + x_n y_1) - (y_1 x_2 + y_2 x_3 + \cdots + y_n x_1) \right|$$

F. Trips

This problem is called *Longest Common Subsequence* not to be confused with "Longest Common Substring".

There is a DP solution as follows:

A 2-D array C keeps track of the largest common substrings. C[i][j] contains the length of the longest common substring in $S_{0\cdots i}$ and $T_{0\cdots j}$.

Here is the recurrence relation:

```
If (S[i] = T[j])                  // if the characters Si and Tj match
    C[i][j] = 1 + C[i-1][j-1]     // add 1 to the upper left diagonal cell
else
    C[i][j] = max(C[i-1][j], C[i][j-1])  // copy the max of the left and above cells
```

Here is an example using S = [C,E,B,F] and T = [E,F,B,F].

	j=	0	1	2	3
i=		E	F	B	F
0	C	0	0	0	0
1	E	1	1	1	1
2	B	1	1	2	2
3	F	1	2	2	3

Recovering the longest sequence is easy and in this problem it is guaranteed to be unique. Start with the highest count in the array C and backtrack along the path of reducing values and prepend the common character value to the result string each time the count drops. This results in the answer EBF in the example above.

G. Containers

Create an ArrayList of chars containing the top character (so far) of each stack (we don't need to know what is in each stack). For each new character read, subtract its Ascii code from that of the Ascii codes of each element of the array and find the array index corresponding to the minimal difference - this is a *best-fit stack*. There could be several of them. Just choose any of them and place the new character in that position in the ArrayList.

If there is no stack for which the new character has lower Ascii code than the top character of every stack, add another stack, i.e. extend the ArrayList length by one by adding the new character.

H. Where's the Treasure

We must find the center of the circle that passes through three points p, q, r. If you join the three points to form a triangle, the circle is called the *Circumcircle* and its center is the *Circumcenter*. Its center is given by the equations:

$$U_x = [(p_x^2 + p_y^2)(q_y - r_y) + (q_x^2 + q_y^2)(r_y - p_y) + (r_x^2 + r_y^2)(p_y - q_y)]/D$$
$$U_y = [(p_x^2 + p_y^2)(r_x - q_x) + (q_x^2 + q_y^2)(p_x - r_x) + (r_x^2 + r_y^2)(q_x - p_x)]/D$$

where $D = 2[p_x(q_y - r_y) + q_x(r_y - p_y) + r_x(p_y - q_y)]$.

BUT

We know that the circumcenter is the point where perpendicular bisectors of the sides of the triangle meet. The perpendicular bisector of side AB (which lies on the x-axis) has equation $x_t = (B_x - A_x)/2$.

The equation of the side AC is $y_{AC} = (C_y/C_x)x$.

The perpendicular bisector of AC has equation $y = C_y/2 - (C_x/C_y)(x - C_x/2)$. It has the inverse slope of y_{AC} and passes through $C_x/2, C_y/2$.

Substitute the equation for x_t into the equation for y giving the answer y_t

$$y_t = C_y/2 - (C_x/C_y)(x_t - C_x/2)$$

Round the x_t and y_t values to 4 d.p.