ECOO 2012 Programming Contest Solutions and Notes

Final Competition (Round 3)

May 12, 2012

Problem 1: The Word Garden

Recommended Approach

Trying to generate and draw the trees on the screen all at once will be quite difficult. A better idea is to create a two-dimensional array of characters (24 rows, 40 columns), "draw" the trees into this array, then draw the array to the screen. Once the array is created, you can fill it with spaces and draw the ground in the final row immediately. Then you need an algorithm (preferably in a method, function, or procedure) for drawing a tree with its trunk at a given location. Test that algorithm to make sure you can draw a single word tree anywhere you want. It's a good idea to put it in a method, procedure, or function.

The next part is figuring out where to draw each tree. The first one is not hard. If the word is 5 characters long, its trunk should be in the 5th column of the array. But figuring out where to draw each of the next trees is harder. You could try to do some mathematical or logical reasoning to figure out where each one should go, but once you have the drawing algorithm done, you can copy it and modify it so that instead of actually drawing the tree, it checks all the array locations the tree would need for a "collision" with other trees. This second algorithm can be used to test possible locations for the next tree. If you know where the trunk of the last tree was, you can start at that location plus 1 and then keep moving to the right until your tree checking algorithm says it's ok. Then draw the tree there.

The Test Cases

The text for both data sets is from Martin Luther King Jr.'s famous "I have a dream" speech.

Solution to DATA11.txt

```
         d
        drd
   h    drerd    t
  hah  dreaerd  tht
 havahdreamaerdthaht  o    d     t
havevah   d   thatahtono  dad   tht
   h      r      t  onenodayadotheht
   a      e      h    o    d ono t
I  va     a      a    n    a  o  h
I  ea     m      t    e    y  n  e
=======================================
```

```
               G
              GeG
             GeoeG            f
            GeoroeG          fof
       h    GeorgroeG        forof
      hih  GeorgigroeG      formrof
     hilihGeorgiaigroeG s  formemrof
    hilllih     G      sosformeremrof
  rhillsllih    e  t  sonos    f
 rer   h        o thtsonsnos   o
reder  i o      rtheht  s o    r
  r    lofo     g  t    oofo   m
  e    l o      i  h    n o    e
  d    s f      a  e    s f    r
=======================================

     s                  f
    sls                fof
   slals              forof      s
  slavals            formrof    sls
 slavevals        s  formemrof  slals
slavesevals      sosformeremrofslavals
     s  a    t  sonos    f    slavevals
     l ana  thtsonsnos   o        s
     aandnatheht  s o    r        l
     v  a    t    oofo   m        a
     e  n    h    n o    e        v
     s  d    e    s f    r        e
=======================================

     o
    owo
   ownwo
  ownenwo
 ownerenwo  w      a         d
ownersrenwowiw    aba       dod
     o    wiliw  ablba  s  dowod
     w   willliwablelbasisdownwod
     n      w b    a tsitis  d
     e      ibeb   btot s    o
     r      l b    l t  i    w
     s      l e    e o  t    n
=======================================
```

```
                    b
                   brb
                  brorb
                 brotorb
                brothtorb
               brothehtorb
       t      brotherehtorb
      tot    brotherhrehtorb
     togot  brotherhohrehtorb
    togegot brotherhooohrehtorb
   togetegotbrotherhoodoohrehtorb
  togethtegot         b
 togethehtegot   t    r
togetherehtegot tat   o
       t       tabat  t
       o      tablbat h
       g    ttablelbate
       e   tht   t    r
       t atheht  a o  h
       hata t    bofo o
       e a  h    l o  o
       r t  e    e f  d
=======================================
```

Solution to DATA12.txt

```
                             l
                            lil
          d                litil
         drd              litttil
   h    drerd    t      f  littlttil
  hah  dreaerd  tht    foflittlelttil
 havahdreamaerdthaht  fouof    l
havevah   d   thatahtfouruof   i
   h      r      t m    f      t
   a      e      hmym   o      t
I  va     a      a m    u      l
I  ea     m      t y    r      e
=======================================

       c
      chc
     chihc
    chilihc
   childlihc                   n
  childrdlihc                 nan
 childrerdlihc               natan
childrenerdlihc             natitan
       c   w              l  natioitan
       h  wiw            lilnationoitan
       i wiliw  o    d  livil    n
       lwillliwono  dadlivevil   a
       d   w  onenodayad  l i    t
       r   i    o    d    iini   i
       e   l    n    a    v ia   o
       n   l    e    y    e na   n
=======================================
```

```
                           j
                          juj
    w                    juduj
   whw                  judgduj
  whehw    t      w     judgegduj
 wherehw  tht    wiw   judgedegduj
whererehwtheht  wiliw  n    j    t
    w   theyehtwilliwnon   u   tht
    h      t      w  notonb d btheht
    e      h      i    n bebgbyb t
    r      e      l    o  b e b  h
    e      y      l    t  e d y  e
========================================

     c
    coc
   coloc      t
  cololoc    tht
 colouoloc  theht    s
colouruoloctheieht  sks
     c    theiriehtskiks  b     t
     o        t   skiniksbub   tht
     l o      h      s  butubbtheht
     oofo     e      k    b byb t
     u o      i      i    u  b  h
     r f      r      n    t  y  e
========================================

                       c
                      chc
                     chahc
                    charahc
      c            chararahc
     coc           characarahc
    conoc         charactcarahc
   contnoc       charactetcarahc
  contetnoc    tcharacteretcarahc
 contenetnoc  tht       c
contentnetnoctheht      h
      c     theieht     a
      o    theirieht    r
      n        t        a
      t o      h        c
      eofo     e        t
      n o      i        e
      t f      r        r
========================================
```

Problem 2: Jewelry Tips

Recommended Approach

The best way to solve this is just a top to bottom, left to right sweep of the entire board. For each jewel, simulate swaps in all 4 directions (in order of priority – LT, UP, RT, DN) and check for lines, then undo the swap and continue. Keep track of the first "good" and "excellent" tips found, but quit immediately if you find a "normal" tip and return that. If you get to the end of the board without any normal tips, return the good tip you found, or the excellent tip if there was no good tip, or "game over" if no tips of any kind were found.

I found it helpful to create a special counting method (a.k.a. procedure or function) that takes the board and a location as parameters, then counts the jewels of the same colour vertically and horizontally in both directions. If there is only one line >= 3, it returns the length of it. If there are two, it adds their lengths and returns that. If there are none, it returns 0. So 0 means no line, 3 or 4 means a single line, and 5 or more means either more than one line or a line of five.

Some Further Notes

When you swap two jewels, you have to check that the other jewel in the swap doesn't end up creating lines as well. If it does, you've got an excellent tip, so you shouldn't return it as a normal tip.

There are not many situations in which you would return a good tip, since when you one jewel to create a line of 4, there is always another jewel you could move to make a line of 3 instead. It is only when this move creates multiple lines (making it excellent rather than normal) that you can return a good tip.

The Test Cases

The test cases were generated automatically by randomly generating thousands of boards. The boards were thrown away if they already contained a line of 3 or more. If not, they were checked using the jewelry tip routine written above to see what kind of tips they contained and kept if they met the criteria I was looking for.

I went with 10 cases because there are so many different kinds of errors you can make in programming this. I thought that the more test cases there were, the more errors I would shake out of the code being tested. But because rule 5 was added at the last minute to cover an ambiguity in the rules, none of the test cases required the use of that rule.

Solution to DATA21.txt

```
Norm: Y.UP@7,3
Good: W.UP@6,2
Excl: O.RT@4,3
Game Over
Norm: W.RT@3,4
Norm: Y.UP@7,0
Good: W.RT@6,2
Excl: B.DN@6,2
Norm: O.LT@6,3
Excl: O.RT@6,2
```

Solution to DATA22.txt

```
Excl: P.DN@4,5
Excl: G.DN@2,1
Excl: P.RT@7,3
Excl: P.RT@4,0
Good: W.DN@3,6
Good: G.RT@6,4
Norm: R.LT@0,6
Norm: Y.DN@2,2
Norm: G.DN@2,4
Norm: G.RT@3,0
```

Problem 3: Steam Arithmetic

The name “Steam” is a spoof of “Scheme”, an actual dialect of Lisp that is currently in use and is taught as a first programming language at Waterloo and some other universities. You might think that programming this in a Lisp dialect would give an advantage, but because of the requirement that the program read the data from a file and interpret it, it is not necessarily so easy.

Recommended approach

Recursion is the best approach here. Write a method (or procedure or function) that takes a 3-element list, separates the operator and the operands and applies the operator to the operands. If an operand is a list, it should be recursively evaluated before being applied.

The restrictions on spacing and the restriction of operators and operands to a single character make it easy to process the list by pulling apart the string one character at a time. There is no need for any further tokenizing , except in the case of lists as operands.

To isolate list operands, you could use a bracket counting method. That is, start to the right of the first open bracket with the counter set to 1. Step through the string one character at a time. Increment the counter at an open bracket, and decrement it at a closed bracket. When the counter hits 0 you’ve found the end of the list.

The Test Cases

In the test cases, I tried to hit a couple of simple cases, then progressively harder ones. I tried for a few cases with very deep nesting, some with tail recursion, and some with head recursion (http://en.wikipedia.org/wiki/Tail_call)

Solution to DATA31.txt

```
1
0
-35
2
75
77
130
-19683
0
61
```

Solution to DATA32.txt

```
0
10
157
8
-20
160
6
-3456
1152
-9
```

Problem 4: Splitsville

This problem is inspired by the field of Machine Learning, specifically Concept Learning Systems. In a concept learning system, you have a list of training objects (e.g. satellite pictures of possible oil spills, data from medical scanners, etc.) with category labels (e.g. is/is not an oil spill, is/is not a tumor, etc.). Each object is described using a list of properties (usually numbers) and the task is to try to form a theory that separates the objects into their different categories. This theory can be viewed as a partitioning of a multi-dimensional space.

In this problem the training objects are houses, the labels are A and B, and the list of properties for each object are just the x and y location of the house. The "theory" is the set of fences that separate the A's from the B's. In machine learning, the theory you form should eventually do well at categorizing previously unseen objects, but that would only work in this case if A and B families tend to live close to other families of the same type.

There are a number of ways to partition spaces like this, including "nearest neighbour" classification and neural network classifiers. The algorithm used in this question is a simplification of machine learning algorithms that produce decision trees or decision rules as their output.

Recommended approach

This problem lends itself well to a recursive solution in which you partition a region of space into two parts, then recursively subdivide the two new regions created. The base case for the recursion is a "pure" region (i.e. a region with only A or B houses in it). Within each region the simplest strategy is to try all possible partitions and count the number of houses out of place in each one, where the number of houses out of place is just the sum of the minima of the counts of A and B houses on each side. So if the left side has 1 A and 3 B's while the right side has 10 A's and 2 B's, the total number of out of place houses is 3.

There are at least two ways to represent a region in this space. The first is to think of it as a 2-dimensional space bounded by integer coordinates above and below and on the left and right. To process it, you try all partitions half way between each pair of integer coordinates. (To keep life simpler and avoid floating point numbers, you could also double the original coordinates and then only consider splits on odd numbered coordinates.) This approach is not very efficient but it's fast enough for problems of this size. One potential pitfall is that you have to rule out splits that would not separate at least one house from the others.

The second, more efficient, method is to think of a region as just a list of houses. If there are 12 houses in the region, you only need to try at most 11 different cuts on each dimension, even if the houses are separated by thousands of units. To do this, you would have to sort the houses first, then find a split between each pair. Because of the sorting step, and the problems of creating and managing lists, this solution is more difficult to implement.

The Test Cases

The last test case in each set requires a few seconds of processing in Java when using the less efficient representation described above.

Solution to DATA41.txt

```
3
1
60
0
132
```

Solution to DATA42.txt

```
25
0
1
65
156
```