ECOO 2013

Programming Contest Solutions and Notes

Final Competition (Round 3)

May 11, 2013

Sam Scott, Sheridan College (sam.scott@sheridancollege.ca)

Problem 1: Irregular Expressions

Recommended Approach

This problem is fairly straightforward but for most solutions it will require carefully keeping track of where you are in each string using two indices. You also have to watch out for running out of characters in either the pattern or the target during the matching process and abort if that happens.

In my solution, when I encounter a bracket in the pattern, I read ahead to the next bracket and build a string of the n characters in the brackets. Then I look at the next n/2 characters in the target (rounding up) and for each one that appears in the string of characters I built, I remove it that character from the list and continue. If I find a character that doesn't belong, I stop.

The Test Cases

The sample data given in the question is very easy – the patterns are short and none of them contain more than one set of brackets. There is also a simple strategy that gets them all right – simply compute the correct length of a matching target string and accept any target string that is of the correct length. The real test data was constructed so that this strategy will fail on all but a couple of cases.

The test cases were also designed to test boundary cases. There were patterns that started with [, patterns that ended with], patterns that contained adjacent brackets (i.e. [..][..]), and patterns with no brackets at all. Non-matching strings were also designed to cover a variety of cases that might cause problems.

Solution to DATA11.txt

```
false false true true false
false false true true true
false true true true false
false true true false true
false true false true false
false false false false false
true true false true false
true false true true false
true false false true false
false false false false false
```

Solution to DATA12.txt

```
false false true false false
true false false true false
false true true false false
true true true true true
true true true false false
false true true false true
false false true true true
true false false true true
false false true true true
true false false false false
```

Credits

```
Question Design: Sam Scott
Verification & Proofreading: Greg Reid, Amlesh Jayakumar
```

Problem 2: Mutant Children

Brute Force Approach

The basic idea is to take the two parents and produce both possible children for each pair of crossover points using the method described in the question. You will need nested loops for this, and the number of children produced this way will be proportional to n^2 where n is the length of the parents. Then for each possible child, you compare it to the target child that was given and count the number of differences. This count represents the number of mutations that would have had to happen to result in the given child. The smallest number you get is the one you use to compute mutation rate.

Because the counting that happens at each step of the algorithm runs in time proportional to n, the total amount of time to compare all children is proportional to n^3. This is feasible for small problems but bogs down when n gets close to 10 000. If you use this method you will not have time to complete all the test cases.

A Better Approach

You might notice that in the above you are duplicating a lot of work. There are at least two better approaches that run in time proportional to n^2. Can you figure out an efficient solution? If you have an idea, post it to the appropriate discussion forum at compsci.ca.

Solution to DATA21.txt

```
0.15
0.06
0.07
0.33
0.37
0.24
0.31
0.22
0.42
0.40
```

Solution to DATA22.txt

```
0.17
0.31
0.34
0.40
0.26
0.32
0.16
0.06
0.34
0.08
```

Credits

```
Question Design: Sam Scott
Verification & Proofreading: Greg Reid, Amlesh Jayakumar
```

Problem 3: Go With the Flow

Recommended Approach

This can be solved with a standard backtracking path search algorithm, with a couple of twists. The first twist is that when you find a goal you have to switch targets and continue until you have found all the targets, but you still need to be able to backtrack right back to the beginning. (In my solution, I counted the targets first so I knew when I was finished.) The second twist is that before you declare victory you have to check to make sure your solution used all the grid squares. If not, you have to backtrack and continue. The boards are small and highly constrained because of the number of targets, so a properly implemented solution should finish very quickly, well within the 10 seconds allotted for each board. In fact, the boards are so constrained that one team got 7 out of 10 correct without using backtracking.

Solution to DATA31.txt

```
11111
12221
32341
33341
44441

12333
12223
11423
44423
42223

11112
13322
13444
43454
44455

111111
222231
244435
245535
242535
222555

333333
312223
311111
333334
455554
444444

112222
134442
133342
111142
444442
422222

1166666
1264446
1264346
1264346
1265346
1265346
1225366

1111666
1231646
2231646
2533646
2555546
6777546
6666666

2222222
2111112
2132222
4536666
4536333
4533373
4577773

1111112
1344432
1335532
1135332
6133311
6111116
6666666
```

Solution to DATA32.txt

```
12222
11332
41132
43332
43222

11111
12221
31111
31444
33333

11111
22324
23324
23224
22244

111111
225551
325451
355441
356641
356111

111222
133332
334452
344652
366652
222222

112222
122333
144563
114563
511563
555533

1122333
1342353
3342253
3244253
3222253
3555553
3333333

1111444
1444424
1422224
1423334
1423225
1422265
1466665

4444441
4222241
4333241
4453341
6655511
6777557
6667777

4444444
4122224
4123334
4113555
4413567
5413567
5555567
```

Credits

```
Question Design: Sam Scott
Verification & Proofreading: Sean Robertson
```

Problem 4: Tour De Force

Observations

The first thing to notice is that this problem gets a lot simpler if you ignore the first question on each card. If Pierre gets the first question wrong on a card, the card is discarded before his next turn begins. So there is always a higher score that would result from getting the first question right and the second question wrong. So you can just assume that the first questions will all be answered correctly – add up their scores as a base score and then focus on which of the second questions he gets wrong.

Exhaustive Search

The problem can be solved with a backtracking search algorithm focusing on the set of second questions on each card. At each step of the search you have two recursive calls: either you get the next question right (add its value) or wrong (subtract one point). You are looking to return the higher result from these two calls. The main wrinkle is that you have to keep track of how many cards in a row you have solved, to avoid a Tour De Force situation. If you have answered the last 4 cards correctly, you can't get the next card right.

This works great for small data sets, but it will bog down with anything above 30 or so cards. If you are trying the search using both questions from each card, it will be trickier to get right, and it will bog down above 20 or so cards.

A Better Approach

This problem is very well suited to a technique known as "Dynamic Programming". In this approach you keep track of partial solutions in a "memo" array, then at each step in the backtracking, you first check to see if you already have a solution for the next part and if you do, you don't have to continue. This speeds things up considerably and allows you to easily solve problems with 1000 question cards.

Can you figure out how to "memo-ize" this problem to construct an efficient solution? If so, post your ideas to the appropriate forum at compsci.ca.

Other Approaches

It is also possible to make some progress using a "Greedy" approach to the problem. Again, it helps to make the observation that for the maximum score, Bert will get all of the first questions right. The question is, which of the second questions on each card will he get the points for? In one Greedy approach, you start by assuming he gets all the second questions wrong. Then start marking questions as correct in order of most to least points. If adding a card will create a Tour de Force (a streak of 5), then don't add that one. Move on to another instead. This approach will get 4 of the test cases wrong on each data set.

Another approach is to start by assuming Bert got all the second questions right. Then start marking questions incorrect in order of least to most points. Continue until there are no more Tour de Force's left in the set of cards. This version of a greedy algorithm does not do as well – it only gets a couple correct on each data set.

Solution to DATA41.txt

```
68
114
114
172
163
184
341
5098
7638
10201
```

Solution to DATA42.txt

```
175
143
152
137
154
181
284
5066
7713
9975
```

Credits

Question Design: Sam Scott
Verification & Proofreading: Amlesh Jayakumar