2016 MINI COMP 0

Tony Wong

2016-02-14

M1601 Valentine Day of Week

- Use computer clock to find out the Day of Week for 2000-02-14 (Monday)
- Use a for-loop to simulate years 2000 to N-1 (note that Feb 14 is before the leap day Feb 29)
  - +1 day for non-leap years (366 % 7 = 1)
  - +2 days for leap years (366 % 7 = 2)

M1602 Valentine Roses

- First, identify the position of the centre rose (3)
  - If the rose 3 is in the correct position, A = 0
    Otherwise e.g. it is at (x, y), swap (x, y) with (3, 3) and A = 1
  - Just like Sample Test 1
- Count the number of “2”s around the perimeter, say B
  - That’s the number of swaps required to put the “2”s in the correct positions
- Overall answer = A + B
- Although here the size is fixed to be 5x5 (3 layers), think about:
  - What if there are N layers, or arbitrary size?
  - What if the pattern can be anything? (not just 1 around 2 around 3)

M1603 Valentine Kisses

- Process the messages one by one, maintaining a “state”
  - State 1: seen “>” followed by zero or more “3”s
  - State 0: otherwise (this is the default state)
- Whenever we see a “>”, switch to state 1 and initialize temp (the number of “3”s) = 0
- When we are in state 1 and see a “3”, **increase temp by 1**
- When we are in state 1 and see a “<“, **add temp to sum** and switch back to state 0
- When we are in state 1 and a character other than “3” or “<“, switch back to state 0

M1603 Valentine Kisses

		O	K	>	3	3	<	s	e	e	>	3	<
State	0	0	0	1	1	1	0	0	0	0	1	1	0
Temp	0	0	0	0	1	2	0	0	0	0	0	1	0
Sum	0	0	0	0	0	0	2	2	2	2	2	2	3

- Common Errors:
  - Failure to reset state to 0 after each message
  - Switch to switch 0 erroneously when encountering a “>” in state 1 (There should be 4 kisses in >>3333<<)

M1604 Valentine Sushi

- The first subtask can be simply solved by storing the data using a 2D (100x100) array as a list and processing the queries individually.
- The full solution requires some kind of O(N lg N) sorting (e.g. quick sort / merge sort)
- Note that for R < L, the query has to split into two parts:
  - 1 .. R, L .. N
  - (actual answer is OR of the two)

M1604 Valentine Sushi

- Although not required, you can store their favourite dishes in a single array **A** of size ∑Di by “flattening”.
- For the sample picture, the array would be:
- With these arrays, the queries become: Does number X exist in the subarray
  - A[SS[L]] .. A[SE[R] – 1] for L <= R
  - A[SS[L]] .. A[SE[N] – 1] or A[SS[1]] .. A[SE[R] – 1] for L > R

0	1	2	3	4	5	6	7	8
1	3	2	1	1	2	3	4	4

SS: Seat Start (included)	SE: Seat End (included)
0	2
2	3
3	3
3	4
4	6
6	7
7	8
8	9

M1604 Valentine Sushi

- After splitting, sort the queries by L then R (Keep the query IDs)
- Maintain M lists, one for each color
  - Initially they are all empty
  - These lists are for storing which queries are “active”
  - Each element is a pair of integers: R and ID
- Go through Seats i = 1 .. N one by one
  - Insert queries with L = i into the lists (the queries become active)
  - For each color j in seat i: go through list j and clear the list at the end
    - If the query should be expired, i.e. i > R, just ignore it
    - Otherwise, the answer for query ID is Yes
- Overall complexity: $O(N + M + \sum D_i + Q \lg Q)$

M1604 Valentine Sushi

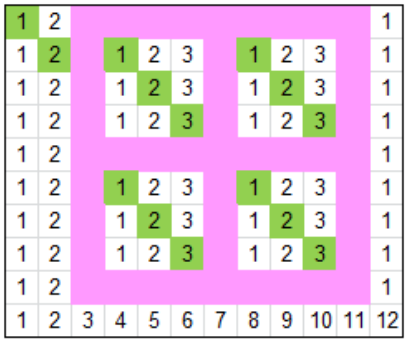
- Alternative solution using binary search tree (set)
- There will be M sets (one for each dish)
- Insert all the seat numbers for Dish i into set i
- For each query, determine whether there exist a number in L .. R / L .. N or 1 .. R in set X

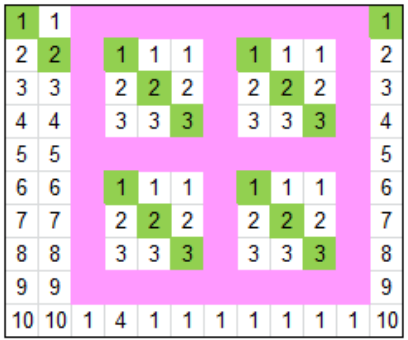
```
s[x].lower_bound(l) != s[x].upper_bound(r)
```

- Overall complexity: O(N + M + ∑Di + Q lg N)

M1605 Valentine Gift Box

- The first approach is by assuming that a dot is the bottom-right corner of the bottom right space
  - Compute distance to nearest left edge and top edge
  - Compute 2D partial sum for dots
  - For each cell, if both distances are equal (say x), check if the bottom-right space is completely filled with dots:
    - dot[i][j] – dot[i – x][j] – dot[i][j – x] + dot[i – x][j – x] == x * x
  - If so, continue checking the other spaces and finally the boundary and ribbon (# area should be 6x + 9)





M1605 Valentine Gift Box

- The second approach is to identify the pattern: 
- If processed from top to bottom, left to right, seeing this pattern would mean it is the top-left corner of a box
- Obtain the width w, height h of the empty space by looking for ##
  - You will see #. until the ribbon
- Area of the box = (2w + 1) x (2h + 1)
- Finally, you need to erase the box by replacing the whole area with some other character to avoid duplicate processing