Team Formation Test Editorial

Conveyor Belt Sushi

Setter: Theo

Prepared By: Theo

Conveyor Belt Sushi

SubTask 1&4: N<=Q

Alice won't be able to handle the same sushi again if she passed the sushi (via N)

We can imagine the situation as sushi disappears after passing

Conveyor Belt Sushi

Also, there will be no "holes" on the right side of Alice.

Therefore Block = All sushi stops

Nothing = Pass the sushi

Take = Take the sushi, then pass

Conveyor Belt Sushi

```
x = k
for i from 1 to Q
    if op == 'T' then
        output x; x = x % N + 1;
    else if op == 'N' then x = x % N + 1;
```

O(N)

Conveyor Belt Sushi

SubTask 1 & 2: N <= 1000

Direct simulate what happens on the belt, O(NQ)

Conveyor Belt Sushi

SubTask 3: Never block

To ease the problem, let’s imagine that the belt does not move, but Alice moving to the right each second.

Conveyor Belt Sushi

Then, we can use an array to handle if the sushi is taken or not, and handle the “take” operation accordingly. Nothing operation just literally do nothing.

O(Q)

Conveyor Belt Sushi

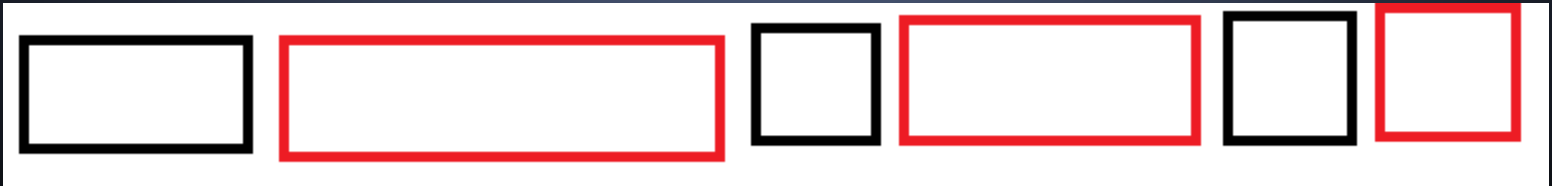
Full:

Let’s imagine, in some moment, the belt looks like “SS...S..S.”, where S are sushi disks and . are spaces (taken sushi)

Conveyor Belt Sushi

We can group continuous segment of (sushis or spaces) together, and link the neighbour ones using link.

For example, the belt will look like



Conveyor Belt Sushi

We are on some point on some block, and for each second:

if we are on a “space” block, doing N, T or B will not affect the belt (output -1 if T)

Conveyor Belt Sushi

If we are on a “sushi” block,

if next operation is N tthen we do nothing

if next operation is T then the current block will “split” into two blocks (if we take sushi from the middle of the block), OR the previous block and next block (both are “spaced” combined into one (if the length of the current block is 1) OR one of the neighbour block will increase length by 1(if we take sushi from ends of the block)

if next operation is B, then the current block will “split” into two blocks (if we block from somewhere not on the front of the block), OR the previous block will have length increase by 1 (if we block on front of the block). Also, we need to handle that the length of the next “spaced” block will decrease by 1, with possible combination of blocks.

Conveyor Belt Sushi

Notice that each operation involves O(1) number of blocks and re-pointering, therefore the complexity is O(Q).

Implementation is left as exercise

Hint: Recall how to build and maintain circular linked-list

Conveyor Belt Sushi

the same thing holds for B:

Elevator

Setter: Theo

Prepared By: LMH

Elevator - Subtask 1

Maintain an array represent all people (inside elevator)’s destinations

For each floor x from 2 to H

Letting people with destination x to leave

- Find the largest array index y of people with destination x
- Increase counter by array_size - y + 1
  - #people need to leave the elevator temporarily or permanently
- Update the array: Remove all people going to floor x

Letting them to re-enter

- Increase the counter by (array_size - y + 1) - #people left
- #People need to leave the elevator temporarily

Letting people waiting at floor x to enter

- Letting people at floor x by pushing them to the end of the array, increase the counter by one for each person

O(H * M)

Elevator - Subtask 2

Observe the fact that only O(M) floors would have people going in or out

No need to go to each floor, we should skip the floor if no one is waiting and leaving at that floor

Using discretization with similar counting approach in subtask 1

O(M^2)

Elevator - Subtask 3

Observe that in Subtask 1, it is very exhausting to:

1. Locate people going to that floor by looping through all people in the elevator
2. Update the array by removing people who just left

Elevator - Subtask 3

Following approaches could improve these performance:

1. For each destination floor from 2 to H, maintain a linked list storing all people (who are leaving at that floor)’s array indexes
2. Whenever a person gets into the elevator, push it to the end of array with state 1 (active state). Whenever a person gets out, change his/her state to 0 (inactive state)

We will talk about the exact flow in next slide

Elevator - Subtask 3

For each floor x from 2 to H

Letting people with destination x to leave

- Find the minimum array index y in the corresponding linked list by looping through the whole linked list
- Increase counter by number of state 1 people from y to array_size
- #people need to leave the elevator temporarily or permanently

Letting them to re-enter

- Loop through the linked list again, update the state of the people of those array indexes to state 0
- Increase counter by number of state 1 people from y to array_size again
- #People need to leave the elevator temporarily

Letting people waiting at floor x to enter

- Push them to the end of the array with state 1
- Increase the counter and array_size by one for each person

Required Data Structure: Binary Indexed Tree or Segment Tree (Point update, range query)

Elevator - Subtask 4

Merging the techniques used in Subtask 2 & 3

O(M log M)

Congressman Lee Sin

Setter: Theo

Prepared By: Sampson, Tony

Congressman Lee Sin

Given an edge-weighted tree, find a set of edges such that the remaining components have at most C nodes each.

Minimize the sum of edge weights in the set.

Congressman Lee Sin - Subtask 1

Exhaustion

For each edge, try destroy and NOT destroy

Check that the components has <= C nodes

If true (valid), ans = min(ans, sum of destroyed edges' weights)

O(2^N)

Congressman Lee Sin - Subtask 2

The given tree is star shaped

The center node is initially connected to N - 1 nodes

Destroying an edge reduces this by 1

Solution: Sort the weights. Answer = sum of the max(0, N - 1 - C) smallest weights.

O(N lg N)

Congressman Lee Sin - Subtask 3

Given tree is a line

We can use dynamic programming

Initialize DP[x] = infinity

x is the number of nodes still connected

DP[x] is the minimum cost to achieve that

Pick a leaf node a, initialize DP[a][1] = 0

Start from a, for node t that is directly connected to last processed node s,

we can destroy the edge(s, t): DP[t][1] = min{DP[s][i = 1 .. c]} + w(s, t)

we can also keep it: DP[t][i] = DP[s][i - 1] i = 2 .. c

Answer = min{DP[b][i = 1.. c]} where b is the other leaf node

Congressman Lee Sin - Subtask 4

C <= 2

First, let's make node 1 as the root of the tree

For each intermediate node x, we have two cases:

1. Not connected to any child node

   DP[x][1] = sum{w(x, y) + min(DP[y][1], DP[y][2])} where y is a child of x

2. To be connected to a child node z

   Then z must be a leaf node or a case 1 intermediate node

   Which node should we connect to? (How to choose z?)

   DP[x][2] = DP[x][1] - max{w(x, y) + min(DP[y][1], DP[y][2]) - DP[y][1]}

Answer = min(DP[1][1], DP[1][2])

Congressman Lee Sin - Subtask 5

For each intermediate node x, we have two cases:

1. Not connected to any child node

   DP[x][1] = sum{w(x, y) + min{DP[y][i = 1 .. c]}} where y is a child of x

2. Connect to some children

   If we choose to connect to a child z which has k = 1.. c nodes connected,

   DP[x][1 + k] = DP[x][1] - w(x, z) - min{DP[z][i = 1 .. c]} + DP[z][k]

   Starting from the second child, we have to consider all cases
   that a total of j = 1 .. c nodes (including x) have been connected to x

>> In general, DP[x][j + k] = DP[x][j] - w(x, z) - min{DP[z][i = 1 .. c]} + DP[z][k]

Note: DP[x][...] has to be updated simultaneously

Answer = min{DP[1][i = 1 .. c]}

Barcode Scanner

Subtask 1:

- Inspect the input files
- The scale of the barcode is 300% (3 pixel per bit)
- Use a for loop to read the values
- Convert (decode) values to barcode

Use floating point instead of integers

I’ll skip subtasks 2 and 3 for the moment

Barcode Scanner - Subtasks 4 - 6

Common properties of the images

- Scale
- Rotation
- Perspective (closer is larger, farther is smaller)

Design an algorithm that tackles these properties at once

Barcode Scanner - Algorithm

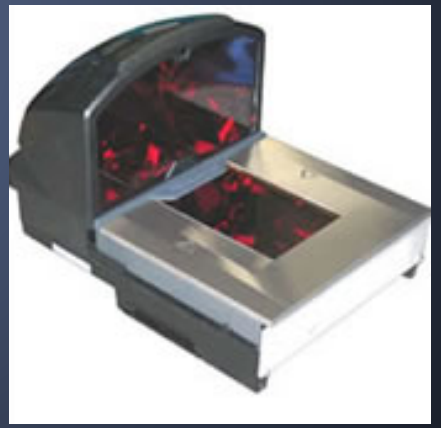
Same algorithm used by supermarket scanners

Randomly choose a starting point (x, y)

Randomly choose an angle θ

Use for loop to iterate

- x = 10, 20, .. 290
- y = 10, 20, .. 390
- θ = 0, 0.1, 0.2, … 6.2 radians



Barcode Scanner - Algorithm

Fire a line from (x, y) to angle θ, and stop until we have 60 edges

- Sampling: use a for loop to add x += dx, y += dy, where dx = sin(θ) and dy = cos(θ)
- An edge is where white > black or black > white

Store the positions of the transitions

barwidth[i] = edgepos[i + 1] - edgepos[i]

Barcode Scanner - Algorithm

barwidth[0 - 2] is the start / ending bars

barwidth[3 - 6] is the second digit

Decoding using barwidths

- sum = barwidth[3] + …[4] + …[5] + …[6]
- a = round(barwidth[3] * 7.0 / sum)
- b = round(barwidth[4] * 7.0 / sum)
- c = round(barwidth[5] * 7.0 / sum)
- d = round(barwidth[6] * 7.0 / sum)

Barcode Scanner - Decoding

v = a * 1000 + b * 100 + c * 10 + d

Lookup v from the table on the right

(Digits 2, only left column is allowed)

(Digits 8 – 13, only right column is allowed)

If v cannot be found, count as **failed digit**

Output the combination with

fewest failed digits

Bar width	
odd	**even**
3211	1123
2221	1222
2122	2212
1411	1141
1132	2311
1231	1321
1114	4111
1312	2131
1213	3121
3112	2113

Barcode Scanner - Subtasks 7 - 9

Real images have noise

220 -> 218 -> 220 (white)

Use **Schmitt trigger** edge detector

Consider it an edge only if the difference is larger than a threshold (e.g. 20)

```
52  for (int i = 1; i < l; i++) {
53    if (mode == 0) {
54      if (s[i] > maxv + THRESHOULD) {
55        dists[count++] = i;
56        mode = 1;
57        maxv = s[i];
58      } else {
59        maxv = min(maxv, s[i]);
60      }
61    } else if (mode == 1) {
62      if (s[i] < maxv - THRESHOULD) {
63        dists[count++] = i;
64        mode = 0;
65        maxv = s[i];
66      } else {
67        maxv = max(maxv, s[i]);
68      }
69    }
70  }
```

Barcode Scanner - Improving accuracy

If you find a line with 1 to 2 failed digits,

try firing lines from the region with smaller steps (e.g. 2 pixels, 0.02 radians)

Use bilinear extrapolation instead of nearest neighbor or round down

```
12 double interpolator(double x, double y) {
13   int xx = floor(x);
14   int yy = floor(y);
15   double p = a[xx][yy] * (1 + xx - x) + a[xx + 1][yy] * (x - xx);
16   double q = a[xx][yy + 1] * (1 + xx - x) + a[xx + 1][yy + 1] * (x - xx);
17   return p * (1 + yy - y) + q * (y - yy);
18 }
```

Barcode Scanner - Subtasks 2 and 3

Apply “motion blur” vertically

Average the pixels from rows 100 to 150

This can reduce the noise to a low level

Barcode Scanner - Improving scores

The length of LCS for two random strings is proportional the length and inversely proportional to the square root of alphabet size

The expected LCS length for two random strings of length = 12 and alphabet size = 10 is around 4

You can get 11 marks by simply outputting “12345689012”