2021 SBU ICPC Selection Contest Editorial

November 2, 2021

A - Baking Rivalry

Problem Setter: *Shawn Mathew,* ***Validator:*** *Yimin Zhu*

Frequency count of C and S, and output correct message corresponding to counts.

B - Lighting the Way

Problem Setter: *Allen Kim,* ***Validator:*** *Astra Kolomatskaia*

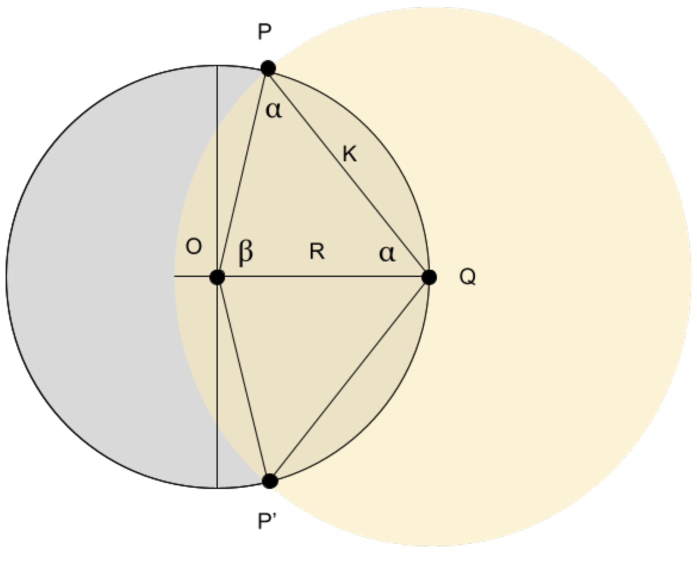

Figure 1: Diagram for Lighting the Way

We refer to Figure 1 for the solution. Our goal is to find K. First, we know the relation between α and β to be:

$$\beta = \pi - 2\alpha$$

By law of cosines, we know that

$$K = 2R\cos\alpha$$

Finally, we know we want the area to be half, so we compute the area by summing up the area of the region emanating from Q leftwards, then the area of the region emanating from β rightwards, and subtract out the area of the quadrilateral $OPQP'$ to avoid double counting. Finally, we can equate this to the half of the area. We get the following:

$$\frac{2\alpha}{2\pi}(\pi K^2) + \frac{2\beta}{2\pi}(\pi R^2) - 2\left(\frac{1}{2}RK\sin\alpha\right) = \frac{1}{2}\pi R^2$$

We can simplify and replace K with the prior expressions to get:

$$\begin{aligned}
\alpha K^2 + \beta R^2 - RK\sin\alpha &= \frac{1}{2}\pi R^2 && (1)\\
\alpha(2R\cos\alpha)^2 + (\pi - 2\alpha)R^2 - R(2R\cos\alpha)\sin\alpha &= \frac{1}{2}\pi R^2 && (2)\\
4\alpha\cos^2\alpha + \pi - 2\alpha - 2\sin\alpha\cos\alpha &= \frac{\pi}{2} && (3)\\
4\alpha\cos^2\alpha + \frac{\pi}{2} - 2\alpha - \sin 2\alpha &= 0 && (4)\\
& && (5)
\end{aligned}$$

To show we can apply binary search, we show that the expression is monotonic as the derivative of this expression is $-4\alpha\sin(2\alpha)$, which is negative in the domain of α (0 to $\pi/2$). After binary search, we get the following approximations for:

$$\begin{aligned}
\alpha &\approx 0.952848 && (6)\\
\beta &\approx 1.235897 && (7)\\
K &\approx 1.158728R && (8)
\end{aligned}$$

We also note a solution that is much simpler by observing that R and K are linearly dependent. Given a sample output, one can also just simply scale R by the sample output ratio (1.158728) to get K directly.

C - Weird Programming Contest

Problem Setter: *Tanzir Islam Pial,* ***Validator:*** *Shawn Mathew*

We can create a graph where each problem is represented by a node and a swap operation from problem a to b is represented by a directed weighted edge $W_{a,b}$. The weights are the time-penalties. Then we will need to find out the shortest path from a node u to v using at most k edges. We can pre-compute the answer for all possible pairs and all possible values of k using dynamic programming in $O(NMK)$.

$dp[i][j][k]$ will denote the shortest path from i to j using at most k edges. Then the recurrence will be:

$$dp[i][j][k] = min(dp[i][j][k-1], dp[a][j][k-1] + W_{i,a}) \tag{9}$$

where a is a direct neighbor of i.

D - By Association

Problem Setter: *Astra Kolomatskaia,* ***Validator:*** *Jiarui Zhang*

Suppose we have an expression like `(((X X) (X ((X X) X))) ((X X) X))`. This is a compound expression whose left subexpression is `((X X) (X ((X X) X)))` and whose right subexpression is `((X X) X)`. The fundamental insight in this problem is that, to left associate the whole expression, we first left associate the left subexpression and **right associate** the right subexpression. In our case, we would get `((((((X X) X) X) X) X) (X (X X)))`. We may then run the `B` command at the top level twice to bring the `X`s on the right side over to the left side, one by one, until the right side is reduced to `X`. Conversely, if we want to right associate this expression, then we would first bring it to the same form as before, except that we would conclude with running the `A` command at the topmost level five times in order to bring the `X`s on the left side over to the right side. Here is a slick Racket solution:

```racket
#lang racket

(define input (read))

(define (size expr)
  (cond
    [(cons? expr)
     (+ (size (first expr)) (size (second expr)))]
    [else 1]))

(define (left-associate expr)
  (cond
    [(cons? expr)
     (append
      (map (lambda (lst) (cons #\L lst)) (left-associate (first expr)))
      (map (lambda (lst) (cons #\R lst)) (right-associate (second expr)))
      (for/list ([i (in-range (sub1 (size (second expr))))]) '(#\B)))]
    [else '()]))

(define (right-associate expr)
  (cond
    [(cons? expr)
     (append
      (map (lambda (lst) (cons #\L lst)) (left-associate (first expr)))
      (map (lambda (lst) (cons #\R lst)) (right-associate (second expr)))
      (for/list ([i (in-range (sub1 (size (first expr))))]) '(#\A)))]
    [else '()]))

(void (map displayln (map list->string (left-associate input))))
```

E - Late Night Snacking

Problem Setter: *Shawn Mathew,* ***Validator:*** *Yimin Zhu*

Loop across each tray and keep track of minimum over the maximum sizes.

F - Help Meow-t

Problem Setter: *Michael Wolf-Sonkin,* ***Validator:*** *Tanzir Islam Pial*

Any pair of line segments, A and B, fall into one of three categories: A is vertically above B, B is vertically above A, neither is above the other. To determine this, project the line segments into the XY plane and check if their projections intersect. You also have to watch out for line segments that project onto a single point! If the projections intersect, one line segment must be above the other. To find out which one is above, compare the Z values of each line segment at their 2D intersection point.

Next, we can construct a directed graph. Each line segment is represented by a node, and an edge from A to B means line segment A is above line segment B. If there's a cycle in the graph, then it is not possible to remove all the line segments from the space. If there is no cycle in the graph, then it is possible to remove all line segments from the space.

G - Hacking ATMs

Problem Setter: *Yimin Zhu,* ***Validator:*** *Allen Kim*

We need to find a number only composed of given digits and that is a divisor of n. A list of all possible combinations of given digits could be retrieved by recursion. Go through the list and check all numbers that is $\leq a$. If $n \mod$ any of these numbers $= 0$, then it is a solution and return YES. Otherwise, NO.

Alternative is to compute all divisors of n, and check if each can be composed of the digits as well.

H - Quads

Problem Setter: *Jiarui Zhang,* ***Validator:*** *Jubayer Rahman*
Given N points in a plane, select 4 points that maximize the area of the quadrilateral with

these four points as the vertices.

First, those selected 4 points are all located at the convex hull (`https://en.wikipedia.org/wiki/Convex_hull`). From the convex hull, we can enumerate two points as a diagonal. Then we can enumerate the remaining two points at different sides of the diagonal. We can separately consider those two points. Now we focus on maximizing the triangle constructed by the selected diagonal and another point.

When we enumerate the third point in order (from left to right, or clockwise), since the length of the base is fixed, the height is a unimodal function, and the change in area is also a unimodal function. Therefore, we can use a Golden-section search to find the point that maximizes the area of the triangle. The total time complexity is $O(N^2 \log_3 N)$.

To further optimize it, we don't need to use the Golden-section search. Consider one fixed point, and enumerate the second point in clockwise order. The third point that maximizes the area of the triangle is also monotonically increased in the clockwise order. In other words, when the best solution to point a and b is point c, the optimal point c' of all b' after b in the clockwise order must not be before the clockwise order of c. Thus, the total time complexity can be $O(N^2)$.