2019 ACM-ICPC North America Qualification Contest Solution Outlines

The Judges

October 5, 2019

icpc.foundation

icpc International Collegiate Programming Contest

JET BRAINS

programming tools sponsor

2019 ICPC North America Qualifier Contest

event sponsor

D– Missing Numbers – First solved at 0:02

Description

Given an increasing sequence of numbers, identify the missing numbers.

D– Missing Numbers – First solved at 0:02

Description

Given an increasing sequence of numbers, identify the missing numbers.

Solution

Read the number of numbers n.

Keep a counter c, starting at 0.

For each of the n input numbers:

- Increment c
- Read input number x
- While $c < x$:
  - Print c
  - Increment c

A– Circuit Math – First solved at 0:03

Description

Given a description of a combinational circuit in postfix format and an input assignment find the output value.
All input circuits should be considered valid (no missing inputs to a gate, only one output).

A– Circuit Math – First solved at 0:03

Description

Given a description of a combinational circuit in postfix format and an input assignment find the output value.
All input circuits should be considered valid (no missing inputs to a gate, only one output).

Solution

This is a standard postfix calculator problem, and can be solved using a stack. Parse each input symbol from left to right:

1. If an input variable is encountered, push its value to the stack.
2. If an operator/gate is encountered, pop the last two values off the stack (one for negation) and apply the operator. Push the result back to the stack.
3. When finished, print the remaining value on the stack.

A– Circuit Math – First solved at 0:03

Solution Strategy

The problem guarantees ensure that the algorithm will work:

- Since every gate is guaranteed sufficient input, there are always enough values on the stack for an operator/gate.
- Since there is only one output (guaranteed) there is a single value on the stack after parsing.

B– Diagonal Cut – First solved at 0:04

Description

Given a $N \times M$ size lattice grid and a diagonal line which goes through the points $(0, 0)$ and (N, M), how many lattice squares are there which are cut exactly in half?

B– Diagonal Cut – First solved at 0:04

Analytic approach

1. The diagonal line equation is $y = \frac{M}{N} \cdot x$.
2. A line cuts a square into two equal-area pieces if and only if it passes through the center of the square.
3. The bottom left corner coordinates of such square is an integer pair (p, q).
4. The square center coordinates are then $(p + \frac{1}{2}, q + \frac{1}{2})$.

B– Diagonal Cut – First solved at 0:04

Analytic approach (continued)

1. Because the line goes through the square center the task is to count all integer solutions of $q + \frac{1}{2} = \frac{M}{N}(p + \frac{1}{2})$, $p < N$.
2. In the equation, express M as $m \cdot \gcd(M, N)$, N as $n \cdot \gcd(M, N)$ and cancel $\gcd(M, N)$ to obtain:
3. $(*)\,(2q + 1) = \frac{m}{n}(2p + 1)$, where $p < N$, m and n are coprime.
4. Count all integer solutions of $(*)$.

B– Diagonal Cut – First solved at 0:04

Analytic approach

Count all integer solutions of $(*)\,(2q+1) = \frac{m}{n}(2p+1)$, where $p < N$, m and n are coprime. **Case 1:** Both n and m are odd.

1. The left side of $(*)$ is an integer, the right side of $(*)$ must be an integer too.
2. Clearly, $(2p+1)$ must be a multiple of n and it also must be odd because all other factors in $(*)$ are odd. Denote the multiplication factor by k.
3. It holds, $(2p+1) = kn, (k = 1, 3, 5, \ldots)$. The condition $p < N$ yields $k = \frac{2p}{n} + \frac{1}{n} < 2\frac{N}{n} + \frac{1}{n} = 2 \cdot \gcd(M, N) + \frac{1}{n}$, or equivalently $k \leq 2 \cdot \gcd(M, N)$.
4. There are exactly $\gcd(M, N)$ odd values of k in the interval $[1, 2, \ldots, 2 \cdot \gcd(M, N)]$, meaning there are exactly $\gcd(M, N)$ integer solutions of $(*)$ and of the whole problem.

B– Diagonal Cut – First solved at 0:04

Analytic approach

Count all integer solutions of $(*)\,(2q+1) = \frac{m}{n}(2p+1)$, where $p < N$, m and n are coprime. **Case 2:** Either n or m is even (both cannot be even, they are coprime).

1. Multiply $(*)$ by n to obtain
2. $n(2q+1) = m(2p+1)$.
3. Exactly one of the four factors $n, (2q+1), m, (2p+1)$ is even, thus the parity of the left side and of the right side in the last equation is not the same.
4. Consequently, $(*)$ has no integer solutions, the answer is 0.

B– Diagonal Cut – First solved at 0:04

Pseudocode

In summary,

```
D = gcd(N, M)
N /= D
M /= D
return D if (N % 2 == 1) and (M % 2 == 1) else 0
```

B– Diagonal Cut – First solved at 0:04

Inclusion-Exclusion

Another approach is to use the principle of inclusion-exclusion. Consider a grid of size $2N \times 2M$. There are four options for the intersection of the diagonal line and a lattice point in the new grid:

1. The point would be in the center of a rectangle in the original grid.
2. The point would be on the side of a rectangle in the original grid.
3. The point would be a corner of a rectangle in the original grid.

There are two ways to be on the side. The count of each of these is the simple formula:

$$\gcd(2N, 2M) - \gcd(N, 2M) - \gcd(2N, M) + \gcd(N, M)$$

K– Summer Trip – First solved at 0:05

Description

Given a string s, count the number of intervals $[a, b]$ so that $s[a]$ and $s[b]$ do not occur anywhere else inside the string.

K– Summer Trip – First solved at 0:05

Description

Given a string s, count the number of intervals $[a, b]$ so that $s[a]$ and $s[b]$ do not occur anywhere else inside the string.

Solution

For each possible starting point a, count valid endpoints b with a linear sweep. Although seemingly $O(n^2)$, you can argue that the total time cost of the sweep is actually $O(26n)$.

K– Summer Trip – First solved at 0:05

Solution Strategy

For each starting point a:

- Sweep along the string, checking character $a + 1$, $a + 2$, ...
- If $s[a + i] = s[a]$, stop searching: no further intervals can be valid.
- Track which letters you see during the sweep. If $s[a + i]$ hasn't been seen yet, you have found a valid itinerary $[a, a + i]$. Increment a total count.

K– Summer Trip – First solved at 0:05

Solution Strategy

For each starting point a:

- Sweep along the string, checking character $a+1$, $a+2$, ...
- If $s[a+i] = s[a]$, stop searching: no further intervals can be valid.
- Track which letters you see during the sweep. If $s[a+i]$ hasn't been seen yet, you have found a valid itinerary $[a, a+i]$. Increment a total count.

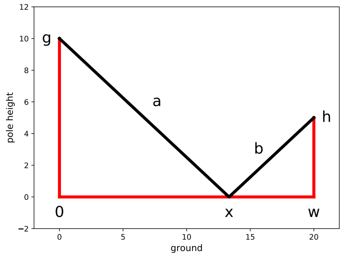
Time Complexity

- Suppose $s[a] = \ell$. Then the number of steps in the above calculation is the distance along the string to the next ℓ.
- You traverse the whole string once for each ℓ.
- Total number of iterations is therefore $26n$.

M– Zipline – First solved at 0:07

Description

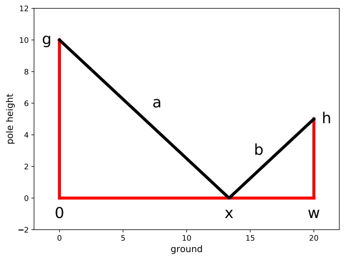
Given a description of a "zipline" hanging between two poles. How long can the line be without touching the ground?



M– Zipline – First solved at 0:07

Description

Given a description of a "zipline" hanging between two poles. How long can the line be without touching the ground?



Solution

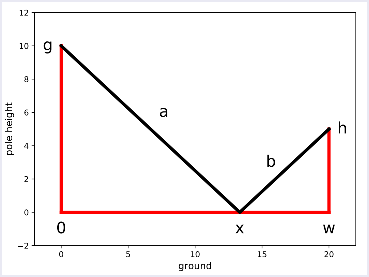
Ignore rider's height r – subtract r from pole heights g and h. Let $g \leftarrow g - r$ and $h \leftarrow h - r$.

Find $f(x)$ that gives length of a line that *just touches* the ground at horizontal position x.

We can find the x that *minimizes* this function using binary or grid search.

M– Zipline – First solved at 0:07

Solution Strategy



The length of the zipline that is connected to the poles and touches the ground at x is:

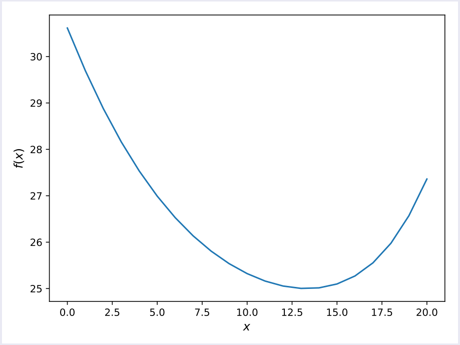
$$f(x) = a + b = \sqrt{g^2 + x^2} + \sqrt{h^2 + (w - x)^2}$$

M– Zipline – First solved at 0:07

Concave Function

$$f(x) = \sqrt{g^2 + x^2} + \sqrt{h^2 + (w - x)^2}$$

$f(x)$ is concave, so we can search it.



M– Zipline – First solved at 0:07

Solution Strategy

The function $f(x)$ represents the longest zipline at each x.

$$f(x) = \sqrt{g^2 + x^2} + \sqrt{h^2 + (w - x)^2}$$

We need to find the *minimum* of this function (the *shortest* longest length). I.e. we need to find where the derivative $f'(x) = 0$:

$$f'(x) = \frac{x}{\sqrt{g^2 + x^2}} + \frac{x - w}{\sqrt{h^2 + (w - x)^2}}$$

We can do this via binary or grid search on x.

For each x, if $f'(x) < 0$ then x is too small; otherwise x is too large.

M– Zipline – First solved at 0:07

Discussion

There's also a closed-form solution (left to the reader; search for "Heron's Problem").

A lot of submissions failed to pass due to division by zero, and printing NaN.

I– Slow Leak – First solved at 0:12

Description

Given a graph G where some nodes are designated as “filling stations”, and a maximum distance d that you can travel without visiting a filling station, what is the shortest path between a starting node (school) and a target node (home).

I– Slow Leak – First solved at 0:12

Description

Given a graph G where some nodes are designated as “filling stations”, and a maximum distance d that you can travel without visiting a filling station, what is the shortest path between a starting node (school) and a target node (home).

Solution

Calculate shortest distances between all nodes in G, then construct a new graph whose nodes are the filling stations, the start and the target nodes only. The edges give shortest paths between the nodes in the original graph. Calculate the shortest path in the new graph.

I– Slow Leak – First solved at 0:12

Solution Strategy

- Use Floyd-Warshall Algorithm to calculate the shortest path between all pairs of nodes in the graph G.
- Create a new graph G' where the nodes are only
  1. The filling stations in G.
  2. The start (1) and target (n) nodes in G
- The edge lengths in G' are the shortest paths between nodes in the original graph G with length at most d (the maximum distance you can travel).
- Use Dijkstra's algorithm (or Floyd-Warshall again) to calculate the shortest path between the start and target in G'.

G– Research Productivity Index – First solved at 0:15

Description

Given the acceptance probabilities of n papers, compute the maximum expected index $a^{a/s}$ assuming that an optimal set of papers are chosen to submit. Here s is the number of papers submitted, and a is the number of papers accepted.

G– Research Productivity Index – First solved at 0:15

Description

Given the acceptance probabilities of n papers, compute the maximum expected index $a^{a/s}$ assuming that an optimal set of papers are chosen to submit. Here s is the number of papers submitted, and a is the number of papers accepted.

Solution

Observe that:

- For a given integer k, if we are to submit k papers, we'd better submit the k papers with the highest probabilities.
- Knowing the set of papers to submit, we can compute the expected index using dynamic programming.

G– Research Productivity Index – First solved at 0:15

Solution Strategy

- Enumerate the integer k, the number of papers to submit.
- Use dynamic programming to compute the expected index when we submit the k papers with the highest probabilities.
- Take the maximum expected index across all choices of k.

Assume the papers are ordered by descending probabilities, and p_i is the ith largest probability, then the dynamic programming can be formulated as follows:

- Let $f(i, j)$ be the expected index after considering the outcome of the first i papers.
- We have the recurrence $f(i, j) = p_i f(i + 1, j + 1) + (1 - p_i) f(i + 1, j)$.
- Boundary conditions: $f(k, 0) = 0; f(k, j) = j^{j/k}$ for $j > 0$.

G– Research Productivity Index – First solved at 0:15

Solution Time Complexity

Enumerating k takes $O(n)$ time. The dynamic programming for each k takes $O(n^2)$ time. The entire solution takes $O(n^3)$ time.

Optimization (not required for AC)

Every time we add a new paper, we don't need to compute the DP table from scratch. For the ith paper, only $f(i,j), 0 \leq j \leq i$ needs to be updated. The overall complexity can be brought down to $O(n^2)$.

Incorrect Approach

Let $g(k)$ be the expected index of submitting the k papers with the highest probabilities. Note that $g(k)$ is not a monotonic function. We therefore cannot greedily stop the enumeration of k upon the first decrease of $g(k)$! Using binary search or ternary search to find the optimal value of k may also lead to wrong answers.

C– Gerrymandering – First solved at 0:16

Description

Given an allocation of voting precincts to districts and party vote totals for each precinct, determine the party that wins in each district. Finally, compute the efficiency gap over all districts.

C– Gerrymandering – First solved at 0:16

Description

Given an allocation of voting precincts to districts and party vote totals for each precinct, determine the party that wins in each district. Finally, compute the efficiency gap over all districts.

Solution

- Add up the total number of votes for A and B within each district.
- After summing all votes, compute the wasted votes for each party district, and report them.
- Sum the total number of wasted votes for A and B, and compute the efficiency gap according to the formula given in the problem.

J– Stop Counting – First solved at 0:18

Description

Given an array of integers, find the maximum attainable average after deleting some subarray.

J– Stop Counting – First solved at 0:18

Description

Given an array of integers, find the maximum attainable average after deleting some subarray.

Solution

The maximum average will always be prefix or a suffix of the array. Try all of them.

J– Stop Counting – First solved at 0:18

Proof

- Every attainable array can be considered as some prefix with average A and some suffix with average B which do not overlap.
- Say A has x elements and B has y elements ($x + y \leq n$).
- Then the weighted average is $\frac{xA+yB}{x+y}$.
- WLOG, assume $A \leq B$.
- Then $\frac{xA+yB}{x+y} \leq \frac{xB+yB}{x+y} = \frac{(x+y)B}{x+y} = B$.

Runtime: $O(n)$.

H– Running Routes – First solved at 0:26

Description

Find the maximum number of straight lines that can be drawn between vertices of a polygon such that no lines intersect, and each line connects two vertices that are "connectable" according to a provided matrix.

H– Running Routes – First solved at 0:26

Description

Find the maximum number of straight lines that can be drawn between vertices of a polygon such that no lines intersect, and each line connects two vertices that are "connectable" according to a provided matrix.

Solution

Find the maximum number of lines that can be drawn for each set of three consecutive vertices, then expand this to increasing numbers of consecutive vertices using the previously calculated results.

H– Running Routes – First solved at 0:26

Solution Strategy

1. Create an $n \times n$ array. The number at (i,j) is the max number of lines that can be drawn among the j vertices starting at index i.
2. The value for any set of three consecutive vertices is easily found from the matrix by checking each option.
3. The value for n consecutive vertices starting at i is the maximum of the following. For each possible connection between vertex n and a vertex j with $i < j < n$, sum the maximum (previously found) for vertices one through $j - 1$ and the maximum for vertices $j + 1$ through $n - 1$ and add one if vertex j is connectable to vertex n. If the maximum for vertices 1 through $n - 1$ is greater, simply store that.
4. Use the previous step to fill in the rest of the table, calculating all sets of three vertices starting before the last two, all sets of four vertices except starting at the last three, and so on. The maximum for all n vertices is the answer.

E– NVWLS – First solved at 1:03

Description

Given a dictionary of words and a message without vowels and spaces, reconstruct the message using only dictionary words. Break ties by largest number of vowels.

Bounds

Message length: $L = 300,000$, total number of characters in dictionary $S = 100,000$, no specific limit on number of words.

E– NVWLS – First solved at 1:03

Description

Given a dictionary of words and a message without vowels and spaces, reconstruct the message using only dictionary words. Break ties by largest number of vowels.

Bounds

Message length: $L = 300,000$, total number of characters in dictionary $S = 100,000$, no specific limit on number of words.

Solution

Preprocess dictionary to remove vowels, then identify all matches of dictionary entries in the message, compute optimal word break using DP based on number of vowels, then reconstruct message.

E– NVWLS – First solved at 1:03

Solution Strategy

- Implementing this strategy requires the use of an efficient string matching strategy. Worst case: $O(L\sqrt{S})$ matches.
- Processing each match must be done in $O(1)$.
- (1) use Aho-Corasick: complexity is $O(|\text{matches}|)$
- (2) use hashing: complexity is $O(L|\text{different word lengths}|)$
- Both are bounded by $O(L\sqrt{S})$.
- DP using number of vowels is relatively straightforward.

L– Traveling Merchant – First solved at 1:09

Description

Given the prices of buying and selling goodies at $n \leq 10^5$ towns arranged in a row, answer $q \leq 10^5$ range queries. Each range query gives a starting town s and a destination town t, and asks about the maximum profit a merchant can earn when he travels from s to t at a speed of one town per day and buys and sells at most one goodie on the way. Each town's price fluctuates daily according to a weekly schedule.

L– Traveling Merchant – First solved at 1:09

Description

Given the prices of buying and selling goodies at $n \leq 10^5$ towns arranged in a row, answer $q \leq 10^5$ range queries. Each range query gives a starting town s and a destination town t, and asks about the maximum profit a merchant can earn when he travels from s to t at a speed of one town per day and buys and sells at most one goodie on the way. Each town's price fluctuates daily according to a weekly schedule.

Solution

Observe that each price has a cycle of seven days, we can precompute $f(s, t, d)$, the maximum profit the merchant can earn if he travels from s to t, and arrives at s on weekday d. There are only seven possible choices of d.

L– Traveling Merchant – First solved at 1:09

Solution Strategy

- For the simplicity of the discussion, we assume the merchant travels left to right in this editorial. The other direction is symmetric.
- We cannot afford computing $f(s, t, d)$ for every s, t. We can use a range data structure to help us maintain a subset of the ranges that are useful to answer any queries. One good data structure is the *segment tree*.
- Let $minPrice(l, r, d)$ and $maxPrice(l, r, d)$ be the minimum and maximum price the merchant can encounter if he travels from l to r, starting on weekday d. These values can be precomputed using standard segment tree construction for range min/max queries (RMQ).

L– Traveling Merchant – First solved at 1:09

Solution Strategy

- For a segment tree node that manages the range $[l, r]$, we precompute the value $f(l, r, d)$ for every d. Let $m = (l + r)/2$, the middle point of the range. Let d_r be the weekday on which the merchant will arrive at town $m + 1$. We have:

$$f(l, r, d) = max \begin{cases} maxPrice(m + 1, r, d_r) - minPrice(l, m, d) \\ f(l, m, d) \\ f(m + 1, r, d_r) \end{cases}$$

- We can construct the tree in $O(7n \log n)$ time.

L– Traveling Merchant – First solved at 1:09

Solution Strategy

For each query range $[s, t]$, the segment tree has $O(\log n)$ nodes that are entirely covered by $[s, t]$ with a parent that is not entirely covered. These are the *hit nodes* of the segment tree. We need to retrieve the following values from hit node i (numbered from left to right) that manages $[l_i, r_i]$: $minValue(l_i, r_i, d_i)$, $maxValue(l_i, r_i, d_i)$, $f(l_i, r_i, d_i)$. Here d_i is the weekday on which the merchant arrives at town l_i. The answer to the query is therefore:

$$max \begin{cases} \max_i \{f(l_i, r_i, d_i)\} \\ \max_{i<i'} \{maxValue(l_{i'}, r_{i'}, d_{i'}) - minValue(l_i, r_i, d_i)\} \end{cases}$$

Querying the segment tree and finding the above max values both take $O(\log n)$ time. Answering each query is thus $O(\log n)$.

F– Prospecting – First solved at 3:33

Description

Given a tree representing a mine, with tunnel lengths y_i and rewards x_i at nodes, and a special "motherlode" leaf node, compute the amount of money you need to start with to find the motherlode for the worst and best case tunnel digging order.

F– Prospecting – First solved at 3:33

Description

Given a tree representing a mine, with tunnel lengths y_i and rewards x_i at nodes, and a special "motherlode" leaf node, compute the amount of money you need to start with to find the motherlode for the worst and best case tunnel digging order.

Solution

Both halves of the problem solved separately, using two different recursive strategies.

F– Prospecting – First solved at 3:33

Solution Strategy: Worst Case

Worst-case behavior: waste as much money as possible digging into side branches of the mine that are not on the path to the motherlode. Then dig one step towards the motherlode. Repeat.

F– Prospecting – First solved at 3:33

Solution Strategy: Worst Case

Worst-case behavior: waste as much money as possible digging into side branches of the mine that are not on the path to the motherlode. Then dig one step towards the motherlode. Repeat.

Computing Wasted Money

How much money $\mathrm{waste}(i)$ can be wasted exploring the mine behind tunnel i? Two options:

- Don't break into the ore chamber: waste $y_i - 1$ dollars
- Break into the ore chamber: waste

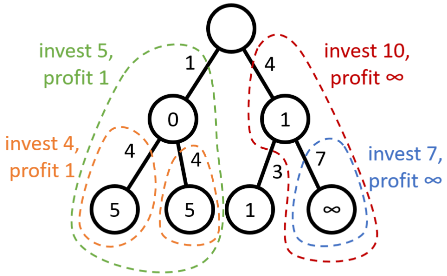
$$y_i - x_i + \sum_{\text{children } j} \mathrm{waste}(j).$$

- $\mathrm{waste}(i)$ is the max of these options. Compute recursively.

F– Prospecting – First solved at 3:33

Solution Strategy: Best Case

Define a subtree of edge *i* to be *optimal* if (a) exploring the subtree yields a profit and (b) the initial money required to explore the subtree is no greater than for any other profitable subtree.



All optimal subtrees of this mine are circled.

F– Prospecting – First solved at 3:33

Solution Strategy: Best Case

Define a subtree of edge i to be *optimal* if (a) exploring the subtree yields a profit and (b) the initial money required to explore the subtree is no greater than for any other profitable subtree.

Key Claim

It is always beneficial to explore the optimal subtree T of the mine entrance with least investment cost.

- After exploration, you end up with more money than you started.
- Either T includes the motherlode, or exploring T requires less investment than reaching the motherlode (since any subtree involving the motherlode is always profitable).

F– Prospecting – First solved at 3:33

Computing Optimal Subtrees

- If $x_i > y_i$, already found an optimal subtree (with profit $x_i - y_i$).
- Otherwise, compute optimal subtrees for all child edges.
- Until the subtree is profitable, explore child optimal subtrees, in order of increasing initial money required.
- After exploring a child optimal subtree, reparent any orphaned portions of the child: edges encountered in the child that weren't explored now have node i as parent.
- Use priority queues to keep the whole calculation $O(n \log n)$.