China Collegiate Programming Contest 2020
Weihai Site
Tutorial

Problem Setters

Nanjing University

A. Golden Spirit

Problem

There are n guys on each side of the bridge. Each guy wants to go across the bridge, relax, finally go across the bridge again. Everyone needs t minutes to go across the bridge and x minutes to relax. You are going to help these guys, and you want to know the minimum time.

A. Golden Spirit

Solution

Assume that you start on the left side of the bridge. There are four steps:

1. It takes $2nt$ minutes to help all $2n$ guys go across the bridge, after which you still on the left side.
2. Make a decision to wait on the left or the right side, and if you choose the right side, it takes another t minutes.
3. You need to wait for the first guy to finish relaxing on the side you chose.
4. It takes another $2nt$ minutes to help all guys go across the bridge again.

A. Golden Spirit

Solution

Assume that you start on the left side of the bridge. There are four steps:

1. It takes $2nt$ minutes to help all $2n$ guys go across the bridge, after which you still on the left side.
2. Make a decision to wait on the left or the right side, and if you choose the right side, it takes another t minutes.
3. You need to wait for the first guy to finish relaxing on the side you chose.
4. It takes another $2nt$ minutes to help all guys go across the bridge again.

Therefore the answer is simply $\min(2nt + \max(2nt, 2t + x), 2nt + \max(2nt + t, t + x))$, considering the choice of the second step and taking the minimum.

B. Labyrinth

Problem

Given an $n \times m$ grid graph with at most k vertices removed. Answer q queries on the shortest path between two vertices.

B. Labyrinth

Problem

Given an $n \times m$ grid graph with at most k vertices removed. Answer q queries on the shortest path between two vertices.

Observation

Given query between (x_1, y_1) and (x_2, y_2) (assume $x_1 \leq x_2$ and $y_1 \leq y_2$),

B. Labyrinth

Problem

Given an $n \times m$ grid graph with at most k vertices removed. Answer q queries on the shortest path between two vertices.

Observation

Given query between (x_1, y_1) and (x_2, y_2) (assume $x_1 \leq x_2$ and $y_1 \leq y_2$),

- if there is no black hole in $[x_1, x_2] \times [y_1, y_2]$, then the shortest path is $|x_1 - x_2| + |y_1 - y_2|$;

B. Labyrinth

Problem

Given an $n \times m$ grid graph with at most k vertices removed. Answer q queries on the shortest path between two vertices.

Observation

Given query between (x_1, y_1) and (x_2, y_2) (assume $x_1 \leq x_2$ and $y_1 \leq y_2$),

- if there is no black hole in $[x_1, x_2] \times [y_1, y_2]$, then the shortest path is $|x_1 - x_2| + |y_1 - y_2|$;
- otherwise, there must exist a shortest path passing an adjacent vertex of some black hole.

B. Labyrinth

Solution

Preprocess compute single source shortest paths $d_v(\cdot)$ for every key vertex v adjacent to some black hole (at most $4k$ such vertices).

Query test if any black hole is in $[x_1, x_2] \times [y_1, y_2]$.

- If so, return $|x_1 - x_2| + |y_1 - y_2|$.
- Otherwise, return $\min_v\{d_v(x_1, y_1) + d_v(x_2, y_2)\}$ where v is a black hole.

Time complexity: $O(kmn)$ for preprocessing, $O(k)$ for each query.

C. Rencontre

Problem

Given a weighted tree T and three lists of nodes. Choose u_1, u_2, u_3 from the three lists uniformly and independently, answer the expected value of

$$f(u_1, u_2, u_3) = \min_{v \in T}(dis(u_1, v) + dis(u_2, v) + dis(u_3, v))$$

.

C. Rencontre

Problem

Given a weighted tree T and three lists of nodes. Choose u_1, u_2, u_3 from the three lists uniformly and independently, answer the expected value of

$$f(u_1, u_2, u_3) = \min_{v \in T}(dis(u_1, v) + dis(u_2, v) + dis(u_3, v))$$

.

Observation

For each tuple (u_1, u_2, u_3), there exists a unique node v such that $dis(u_1, v) + dis(u_2, v) + dis(u_3, v)$ is minimized, such that

$$f(u_1, u_2, u_3) = \frac{1}{2}(dis(u_1, u_2) + dis(u_1, u_3) + dis(u_2, u_3))$$

By the linearity of expectation, we can calculate the expected value of $dis(u_1, u_2), dis(u_1, u_3), dis(u_2, u_3)$, respectively.

C. Rencontre

Problem (Reformulated)

Given a weighted tree and two lists of nodes. Choose u_1, u_2 from the two lists uniformly and independently, calculate the expected value of $dis(u_1, u_2)$.

Solution

Calculate the contribution of each edge respectively. Assume edge e with weight w_e divides the tree T into 2 parts T_1, T_2, the contribution of e is

$$w_e \cdot (Pr[u_1 \in T_1] \cdot Pr[u_2 \in T_2] + Pr[u_1 \in T_2] \cdot Pr[u_2 \in T_1])$$

It can be accumulated by a depth-first search on the tree.
Time complexity: $O(n)$.

D. ABC Conjecture

Problem

Given a positive integer c, determine if there exists positive integers a, b such that $a + b = c$ and the product of distinct prime divisors of abc is less than c.

D. ABC Conjecture

Problem

Given a positive integer c, determine if there exists positive integers a, b such that $a + b = c$ and the product of distinct prime divisors of abc is less than c.

Solution

- If c is square free, then $\text{rad}(abc) \geq \text{rad}(c) = c$, hence there don't exist such a, b;
- If c contains a square factor, namely $c = p^2 q$ $(p > 1)$, then we have $a = pq$ and $b = p(p-1)q$, such that $a + b = c$ and $\text{rad}(abc) = \text{rad}(p^4(p-1)q^3) \leq \text{rad}(p(p-1)q) < p^2 q = c$.

D. ABC Conjecture

Problem

Given a positive integer c, determine if there exists positive integers a, b such that $a + b = c$ and the product of distinct prime divisors of abc is less than c.

Solution

- If c is square free, then $\text{rad}(abc) \geq \text{rad}(c) = c$, hence there don't exist such a, b;
- If c contains a square factor, namely $c = p^2 q$ $(p > 1)$, then we have $a = pq$ and $b = p(p-1)q$, such that $a + b = c$ and $\text{rad}(abc) = \text{rad}(p^4(p-1)q^3) \leq \text{rad}(p(p-1)q) < p^2 q = c$.

To check if c is square free, check p^2 factors for p up to $\sqrt[3]{c}$, and check if c/p is a square integer for p up to $\sqrt[3]{c}$. Overall time complexity is $O(\sqrt[3]{c})$ per case.

E. So Many Possibilities...

Problem

Given n integers $a_1, ..., a_n$. An operation is to choose an index j with positive a_j uniformly, subtract a_j by 1.
Calculate the expected number of zeroes after taking m operations on array a.

Solution

For a series of operations, let s_i be the index chosen by the ith operation, then we get an operation sequence $s = (s_1, s_2, ..., s_m)$. Let $cnt_s(i)$ be the number of occurrences of i in the sequence s.
A sequence is valid if and only if:

$$\forall 1 \leq i \leq n, cnt_s(i) \leq a_i$$

Let Γ be the set of all valid operation sequences $(s_1, ..., s_m)$.

E. So Many Possibilities...

Solution

Fix an operation sequence s with length l, let $\sigma_i^s (1 \leq i \leq l)$ be the number of nonzero integers before the ith operation, and σ^s be the number of nonzero integers after l operations. Define the conditional probability $p(s)$:

$$p(s) = \prod_{k=1}^{l} \frac{1}{\sigma_i^s}$$

The probability of a valid operation sequence s occurs equal to $p(s)$, thus the answer equal to

$$\sum_{s \in \Gamma} p(s) \cdot (n - \sigma^s)$$

E. So Many Possibilities...

Solution

Define $\Gamma_{l,S}(0 \le l \le m, S \subseteq [n])$ as the set of sequences$(s_1, ..., s_l)$ which satisfy the following conditions:

- $\forall 1 \le i \le l, s_i = 0$ or $s_i \in S$
- $\forall j \in S, cnt_s(j) = a_j$

Here $s_k = 0$ denotes the index of the k-th operation $\notin S$.
Apply a bitmask dp, define $dp(l, S)(0 \le l \le m, S \subseteq [n])$ as

$$dp(l, S) = \sum_{s \in \Gamma_{l,S}} p(s)$$

The bitmask dp can be done in $O(2^n \cdot nm)$.

E. So Many Possibilities...

Solution

Since for any $s \in \Gamma_{m,S}$, the $cnt_s(0)$ is same, and we need to color each 0 by an index $x \notin S$ with ensuring for each color j, its occurrence is strictly less than a_j.

Let $c(S)$ denotes the number of ways to color the 0s. Calculating $c(S)$ is a traditional dynamic programming problem, which can be computed in $O(nm^2)$.

Apply the dynamic programming for each subset of $[n]$ requires $O(2^n \cdot nm^2)$ time, but with some smart implementation, it can be solved in $O(2^n \cdot m^2)$ totally.

Finally, the answer can be accumulated as

$$\sum_{S \subseteq [n]} dp(m, S)c(S) \cdot |S|$$

Time complexity: $O(2^n \cdot (n+m)m)$.

F. Skeleton Dynamization

Problem

Factorize a given graph as the Cartesian product of a path P and another graph G.

The Cartesian product (V, E) of two graphs (V_1, E_1) and (V_2, E_2) is defined as

- $V = V_1 \times V_2$;
- $E = \{((u, v_1), (u, v_2)) : u \in V_1, v_1 v_2 \in E_2\} \cup \{((u_1, v), (u_2, v)) : u_1 u_2 \in E_1, v \in V_2\}$.

F. Skeleton Dynamization

Problem

Factorize a given graph as the Cartesian product of a path P and another graph G.

The Cartesian product (V, E) of two graphs (V_1, E_1) and (V_2, E_2) is defined as

- $V = V_1 \times V_2$;
- $E = \{((u, v_1), (u, v_2)) : u \in V_1, v_1 v_2 \in E_2\} \cup \{((u_1, v), (u_2, v)) : u_1 u_2 \in E_1, v \in V_2\}$.

In this problem, the Cartesian product of a path P and a graph G is a layered graph, where each layer is a copy of G and the corresponding vertices of adjacent layers are interconnected by edges.

F. Skeleton Dynamization

Problem

Factorize a given graph as the Cartesian product of a path P and another graph G.

The Cartesian product (V, E) of two graphs (V_1, E_1) and (V_2, E_2) is defined as

- $V = V_1 \times V_2$;
- $E = \{((u, v_1), (u, v_2)) : u \in V_1, v_1 v_2 \in E_2\} \cup \{((u_1, v), (u_2, v)) : u_1 u_2 \in E_1, v \in V_2\}$.

In this problem, the Cartesian product of a path P and a graph G is a layered graph, where each layer is a copy of G and the corresponding vertices of adjacent layers are interconnected by edges.
Without loss of generality, we assume the size of P is at least 2.

F. Skeleton Dynamization

Solution Sketch

1. Find a vertex u with the minimum degree, which must be in the first layer;
2. Enumerate which incident edge uv to be cross-layer edge;
3. Perform a two-source (from u and v) breadth-first search; the corresponding vertices in the two layers have the same distance and are connected by an edge. Hence we can construct the first two layers in this step.
4. Iteratively construct all remaining layers by extending the cross-layer edges of the top layer. Check the graph of the constructed layer is isomorphic to the first layer.

If no conflict is found in step 2 or 3, then we obtain a valid factorization. Note that step 2 and step 3 can be implemented in $O(m)$ time.

Note that the maximum degree of u is $O(\sqrt{M})$, so the total time complexity is $O(M\sqrt{M})$.

G. Caesar Cipher

Problem

Maintain an array of integers which supports the following operations:

- Increase every element in $[l, r]$ by one, modulo 65536;
- Ask if two subarrays $[x, x + l]$, $[y, y + l]$ are the same.

G. Caesar Cipher

Problem

Maintain an array of integers which supports the following operations:

- Increase every element in $[l, r]$ by one, modulo 65536;
- Ask if two subarrays $[x, x + l]$, $[y, y + l]$ are the same.

Solution

- Note that overflow occurs at most $O(nq/65536)$ times (about 5×10^6), hence we may process every single overflow.
- Use a segment tree to maintain rolling hash $b^n a_n \bmod p$; hence the rolling hash of any subarray can be computed by a range sum query.
- Use another segment tree to support query on the maximum of a range; this is used to detect overflow. Once overflow detected on some element, modify the hash value in the segment tree.

The total time complexity is $O(nq \log n/65536 + q \log n)$.

G. Caesar Cipher

There is also a modulus-agnostic solution. Let $b_i = a_i - a_{i-1} \bmod 65536$, then

- query of type 2 asks if $a_x = a_y$ and $(b_{x+1}, b_{x+2}, \cdots, b_{x+l-1}) = (b_{y+1}, b_{y+2}, \cdots, b_{y+l-1})$;
- query of type 1 only updates b_{l+1} and b_{r+1}.

Using rolling hash to maintain $\{b_i\}$, each can be done in $O(\log n)$ time. The total time complexity is thus $O(q \log n)$.

H. Message Bomb

Problem

There are n people and m groups and 3 types of events occur in order:

1. Person u enters group v.
2. Person u leaves group v.
3. Person u sends a message in group v.

We need to calculate the total number of messages each person receives.

Observation

For a person u staying in a group v between time $[L, R]$, we use cnt_v^t to denote the total number of messages in group v before time t. The message he receives equal to

$$cnt_v^R - cnt_v^L - (\text{the number of messages he sends during } [L, R])$$

H. Message Bomb

Solution

For each group v, set a counter cnt_v denotes the total messages. Scan all the events in group v:

- Once person u enters, subtract ans_u by cnt_v.
- Once person u leaves, add ans_u by cnt_v.
- Once person u sends a message, add cnt_v by 1 and subtract ans_u by one.

Finally for each person u stay in group v in the end, add ans_u by cnt_v.
Time complexity: $O(n + m + q)$.

I. Sean the Cuber

Problem

Given two states of $2 \times 2 \times 2$ Rubik cube, find the minimum number of steps to transform the first to the second.
Every quarter twist of a half-plane is counted as one step, and rotating the entire cube doesn't count into the number of steps.

I. Sean the Cuber

Problem

Given two states of $2 \times 2 \times 2$ Rubik cube, find the minimum number of steps to transform the first to the second.
Every quarter twist of a half-plane is counted as one step, and rotating the entire cube doesn't count into the number of steps.

Observation

We may fix a corner of the cube. For example, by only allowing rotating the right, back, bottom half-planes from the solved state, the left upper front corner is fixed. The number of states, given one corner fixed, is 3674160.
We may rotate the entire cube of the initial and the final states such that the left upper front corner is identical to the fixed one. Now we only have to consider the operations of rotating the right, back, bottom half-planes.

I. Sean the Cuber

If we perform breadth-first search of every test case, the total time complexity is $O(T|G|)$, where $|G| = 3674160$. Even if we use bidirectional search, the complexity is still not acceptable.

I. Sean the Cuber

If we perform breadth-first search of every test case, the total time complexity is $O(T|G|)$, where $|G| = 3674160$. Even if we use bidirectional search, the complexity is still not acceptable.

Observation 2

Let A be the initial state, and B be the final state in $2 \times 2 \times 2$ Rubik cube group, then the shortest operation from A to B is equal to the shortest path from the solved state to $A^{-1}B$. So we just have to preprocess the distance from the solved state to every state.

I. Sean the Cuber

If we perform breadth-first search of every test case, the total time complexity is $O(T|G|)$, where $|G| = 3674160$. Even if we use bidirectional search, the complexity is still not acceptable.

Observation 2

Let A be the initial state, and B be the final state in $2 \times 2 \times 2$ Rubik cube group, then the shortest operation from A to B is equal to the shortest path from the solved state to $A^{-1}B$. So we just have to preprocess the distance from the solved state to every state.

But how to compute $A^{-1}B$? In other words, how to compute the inverse and composition of group elements?

I. Sean the Cuber

If we perform breadth-first search of every test case, the total time complexity is $O(T|G|)$, where $|G| = 3674160$. Even if we use bidirectional search, the complexity is still not acceptable.

Observation 2

Let A be the initial state, and B be the final state in $2 \times 2 \times 2$ Rubik cube group, then the shortest operation from A to B is equal to the shortest path from the solved state to $A^{-1}B$. So we just have to preprocess the distance from the solved state to every state.

But how to compute $A^{-1}B$? In other words, how to compute the inverse and composition of group elements?
We may represent each group element as a 24-order permutation, describing how the 24 small squares (4 per face) are permuted. Then we may compute the inversion and composition on the permutations.

J. Steins;Game

Problem

Given n piles of stones, each colored in black or white. Two players take turns to apply one of the following operations:

- Remove some positive number of stones from any white pile.
- Remove some positive number of stones from the smallest black pile.

The player who can't make a move loses.

Calculate the number of ways to color each pile of stones so that the second player wins under the optimal strategy for both players.

J. Steins;Game

- If all piles are white, then the problem reduces to Nim.
- By Sprague-Grundy theorem, one only needs to find out the Grundy value represented by a set of black piles.

J. Steins;Game

- If all piles are white, then the problem reduces to Nim.
- By Sprague-Grundy theorem, one only needs to find out the Grundy value represented by a set of black piles.

Theorem

For a set of black piles, let m be the number of stones in the smallest pile, and c_m be the number of piles with m stones. The Grundy value for the set of black piles is equal to

$$m - ((c_m + [\text{all the black piles have same size}]) \mod 2)$$

J. Steins;Game

- If all piles are white, then the problem reduces to Nim.
- By Sprague-Grundy theorem, one only needs to find out the Grundy value represented by a set of black piles.

Theorem

For a set of black piles, let m be the number of stones in the smallest pile, and c_m be the number of piles with m stones. The Grundy value for the set of black piles is equal to

$$m - ((c_m + [\text{all the black piles have same size}]) \mod 2)$$

- The theorem can be found through computing the Grundy value for certain cases and observing the pattern. The proof can be done through either induction or handwaving.

J. Steins;Game

Solution Sketch

1. Sort all piles by the number of stones, enumerate the size m of the smallest black pile (from smallest to largest), the number of such piles, and whether all the black piles have the same size.
2. Since all the piles smaller than m must be painted white, calculate the number of ways to color the piles larger than m to make the xor sum of Grundy values equal to 0. This can be efficiently done by maintaining the linear basis of piles larger than m.
3. The overall time complexity is $O(n \log n + n \log \max a_i)$

K. Tree Tweaking

Problem

Given a permutation $p_1, p_2, \ldots, p_n$ and $1 \leq l \leq r \leq n$. One may arbitrarily permute the numbers with indices in the interval $[l, r]$. Minimize the sum of depths of every node in the binary search tree built according to the new permutation.

K. Tree Tweaking

Problem

Given a permutation $p_1, p_2, \ldots, p_n$ and $1 \leq l \leq r \leq n$. One may arbitrarily permute the numbers with indices in the interval $[l, r]$. Minimize the sum of depths of every node in the binary search tree built according to the new permutation.

Problem(Subtask)

Given a permutation $p_1, p_2, \ldots, p_n$. Calculate the sum of depths of every node in the binary search tree built according to the new permutation.

K. Tree Tweaking

Problem

Given a permutation $p_1, p_2, \ldots, p_n$ and $1 \leq l \leq r \leq n$. One may arbitrarily permute the numbers with indices in the interval $[l, r]$. Minimize the sum of depths of every node in the binary search tree built according to the new permutation.

Problem(Subtask)

Given a permutation $p_1, p_2, \ldots, p_n$. Calculate the sum of depths of every node in the binary search tree built according to the new permutation.

Solution(Subtask)

Building the binary search tree directly leads to $O(n^2)$ time complexity. Note that the binary search tree partitions all numbers to be added into some disjoint intervals at each step. Maintaining these intervals with some proper data structure(e.g., std::set) leads to $O(n \log n)$ time complexity.

K. Tree Tweaking

Solution Sketch

1. Build the binary search tree for numbers with indices in the interval $[1, l-1]$. This step takes $O(n \log n)$ time. Note that now the remaining numbers are partitioned into some disjoint intervals, and each interval can be solved independently.
2. Now consider how to solve for each interval. Let $dp_{l,r}$ be the best answer we can get for the interval $[l, r]$. With proper preprocessing, the dynamic programming can be calculated in $O(k^3)$ time.
3. The overall time complexity is $O(n \log n + k^3)$.

L. Clock Master

Problem

Given a positive integer b, find positive integers $x_1, x_2, \cdots, x_n$ such that $x_1 + x_2 + \cdots + x_n \leq b$ and $\text{lcm}(x_1, x_2, \cdots, x_n)$ is maximized.

L. Clock Master

Problem

Given a positive integer b, find positive integers $x_1, x_2, \cdots, x_n$ such that $x_1 + x_2 + \cdots + x_n \leq b$ and $\mathrm{lcm}(x_1, x_2, \cdots, x_n)$ is maximized.

Observation

In optimal solution, every x_i is a prime power.

Proof: for prime powers p^i and q^j $(p \neq q)$, $\mathrm{lcm}(p^i, q^j) = p^i q^j$ whereas $p^i + q^j < p^i q^j$.

L. Clock Master

Problem

Given a positive integer b, find positive integers $x_1, x_2, \cdots, x_n$ such that $x_1 + x_2 + \cdots + x_n \leq b$ and $\mathrm{lcm}(x_1, x_2, \cdots, x_n)$ is maximized.

Observation

In optimal solution, every x_i is a prime power.

Proof: for prime powers p^i and q^j $(p \neq q)$, $\mathrm{lcm}(p^i, q^j) = p^i q^j$ whereas $p^i + q^j < p^i q^j$.

Problem (Reformulated)

Let $P = \{p^i : p \in \text{Prime}, i \in \mathbb{Z}^+\}$ be the set of prime powers. Find a set $S \subset P$ such that $\sum S \leq b$ and $\mathrm{lcm}\, S = \prod S$ is maximized.

L. Clock Master

Problem

Given a positive integer b, find positive integers $x_1, x_2, \cdots, x_n$ such that $x_1 + x_2 + \cdots + x_n \leq b$ and $\operatorname{lcm}(x_1, x_2, \cdots, x_n)$ is maximized.

Observation

In optimal solution, every x_i is a prime power.

Proof: for prime powers p^i and q^j $(p \neq q)$, $\operatorname{lcm}(p^i, q^j) = p^i q^j$ whereas $p^i + q^j < p^i q^j$.

Problem (Reformulated)

Let $P = \{p^i : p \in \text{Prime}, i \in \mathbb{Z}^+\}$ be the set of prime powers. Find a set $S \subset P$ such that $\sum S \leq b$ and $\operatorname{lcm} S = \prod S$ is maximized.

This is a standard knapsack problem. Since $|P \cap [1, b]| = O(b/\log b)$, it can be solved in $O(b^2/\log b)$ time.