University of São Paulo

Institute of Mathematics and Statistics

USP Try-Outs 2023

Editorial

Problem	Author
A - Metaverse Real Estate	Victor Lamarca
B - Maracas	Renzo Gómez
C - Quasi-Palindrome	Pedro Sousa
D - Supermarket queue	Enrique Junchaya
E - Long Live Mexico	Gabriel Morete
F - Goalkeeper 7 games (or less)	Renzo Gómez
G - Choice hero	Victor Lamarca
H - The infinite festival	Renzo Gómez
I - Help the Aztecs	Thiago Oliveira
J - Indiana Jiang and the Temple of Kukulkan	Enrique Junchaya
K - Missing Cyan	Enrique Junchaya
L - Tourist circuits	Renzo Gómez
M - Chavos's Barrel	Pedro Sousa

Table 1: Contest problems.

A. Metaverse Real Estate.
Tags : Mathematics, Lagrange Interpolation.
Author : Victor Lamarca.

With some basic combinatorics we can get the formula of the sum of hypervolumes of the cube B and the number of distinct hypercubes B. The answer is the division of one through the other. The sum of hypervolumes can be described by:

$$\text{SumHyper} = \sum_{i=1}^{n} i^k \cdot (n-i+1)^k$$

With some algebra this can be reduced to the sum of powers. Consider the following function:

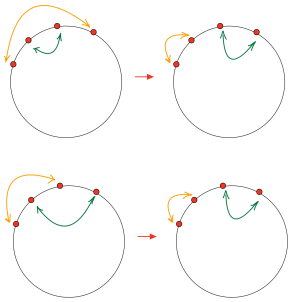
$$f(x) = \sum_{i=1}^{x} i^k$$

SumHyper is a polynomial of degree $2k+1$. With some algebra, *SumHyper* can be expressed as a linear combination of $f(x)$. (Disclaimer: You don't need to use algebra to reduce the first formula, this is only good if you know the result that the second formula is a polynomial and, therefore the first formula is also a polynomial of degree $\mathcal{O}(k)$, for a proof see the editorial of the problem linked below). To get the polynomial, we can calculate the first $2k+2$ values of it (for n from 0 to $2k+1$) and use Lagrange interpolation to get the coefficients of the polynomial and evaluate it at the value n (given in the input), as described in this similar problem (CF 622 F).

B. Maracas.
Tags : Ad-hoc.
Author : Renzo Gómez, Lucas Harada.

We say that a position is *odd* if it contains an odd number of maracas. The first observation is that the number of odd positions has to be even, otherwise there is no solution. Note that every solution can be seen as a matching between odd positions. Moreover, it is sufficient to transfer one maraca between each pair of positions in this matching. Thus, it is enough to consider $m_i \mod 2$, for $i = 1, \ldots, N$.

As the time to move to a certain position is zero, the time to transfer one maraca between positions i and j is $\min\{A, B\} \times |i-j|$. This observation implies that any optimal matching has no crossings (see the following figure).



Therefore, for a given odd position, there are only two possible solutions: match it to the next odd position to the right or to the next odd position to the left. Since once we make this decision, the whole matching is determined, we can solve this problem in $\mathcal{O}(N)$.

C. Quasi-Palindrome.
Tags : Strings.
Author : Pedro Sousa.

For every possible center of a quasi-palindrome, that is for each character and for each center between 2 characters in the given string, we can use hashing and binary search to determine the biggest quasi-palindrome centered there: we use one binary search to find the farther you can go until you find a mismatch and then do this process again starting from the first mismatch until you find another.

You can also solve the problem by using an adaptation of Manacher's algorithm.

D. Supermarket queue.
Tags : Implementation.
Author : Enrique Junchaya, Gustavo Carlos.

We can simulate the entry and exit of the customers using an array of queues. Every time a costumer exits, we could verify if she's sad by iterating for all the costumers that already left the supermarket and checking if any of them entered after him. Of course, this would exceed the time limit.

Notice, however, that is easy to implement the same idea efficiently. Instead of maintaining a list of all the customers that already left, we could just keep the latest entry time among all the costumers that already left. With that, we can check in $O(1)$ if a customer will be sad after leaving the supermarket. Also, notice that updating the latest entry time of a costumer that already left is also $O(1)$ per update.

E. Long Live Mexico.
Tags : Mathematics.
Author : Gabriel Morete.

First and foremost, notice that since the cost function is squared, every coordinate is independent. That means we can decide for each coordinate independently. Therefore, we merely need to solve the uni-dimensional case.

Moreover, notice that the cost function for a single servant drone is convex. Since the sum of convex functions is also convex, the cost function is convex for each dimension. Hence, we may use a ternary search to find the point that minimizes the cost. To test a point, one only needs to evaluate the function in linear time.

We can avoid the ternary search by taking the derivative of the cost function. Differentiating for x results in

$$\frac{\partial}{\partial x} \sum_{i=1}^{n} \text{dist}(0, j)^2 w_j = \sum_{i=1}^{n} 2w_i(x_i - x_0).$$

Hence, the point that minimizes for x is given by

$$x_{min} = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}.$$

Lastly, we must test the neighbor integer points to find the lexicographically smallest that minimizes the cost. Notice that by the convexity of the cost function, it suffices to test at most three points. Hence, the problem can be solved in either $O(n)$ or $O(n \log(max\_coord))$, where max_coord is the maximum value a coordinate assumes.

F. Goalkeeper of 7 games (or less).
Tags : Data structures.
Author : Renzo Gómez, Lucas Harada.

The statement asks to solve the following problem: given an array $A[1, \ldots, N]$, answer/apply the following requests.

- $1\,\ell\,r$: decide whether there exist pairs of indices (a, b) and (c, d) such that $A[a] = A[b]$ and $A[c] = A[d]$
- $0\,i\,x$: make $A[i] = x$

Moreover, we have to attend this requests online.

First, for each position i, let $S[i] = \min\{j : j > i,\ A[i] = A[j]\}$ (if such position j does not exist let $S[i] = N + 1$). We can construct the array S in $\mathcal{O}(N \lg N)$. After that, we create a segment tree M (for minimum queries) over S.

In the case of the first request, if $A[b] = A[c]$, we will suppose that $b < c$. The following observation is important to answer the first request. If there exist such pairs of indices (a, b) and (c, d) in the interval $[\ell, r]$, then there is a solution where b is minimum possible. Since if there is a pair (a', b') such that $A[a'] = A[b']$ and $b' < b$, then we can exchange (a', b') with (a, b). Note that if $A[b] = A[c]$, then $b' < b < c$.

Thus, to answer the first request, we query M over interval $[\ell, r]$ to find the pair (a, b). Finally, we query three intervals $[\ell, a - 1]$, $[a + 1, b - 1]$ and $[b + 1, r]$ to check the following cases:

- $c < a < b < d$
- $a < c < b < d$
- $a < b < c < d$

To implement the second request, we can maintain an auxiliary array

$$P[i] = \max\{j : 1 \leq j < i,\ A[i] = A[j]\}$$

and a collections of sets to maintain the positions where each value occurs. The overall solution has complexity $\mathcal{O}(N \lg N)$.

G. Choice hero.
Tags : Greedy.
Author : Victor Lamarca.

The problem can be solved with a simple greedy strategy: at any level the hero should defeat the strongest monster it can defeat. The stronger the monster he defeats, the stronger he gets, which is better for later levels. By using this optimal strategy we can check if we can clear all the n levels.

H. The infinite festival.
Tags : DP, Divide and Conquer.
Author : Renzo Gómez.

We can model this problem with a digraph where each vertex represents a state (i, j), there is an arc from (i, j) to $(i, j+1)$ representing going to the next day, and from (i, j) to $(i+1, j)$ representing going one level up. Observe that this is like a circular grid (see the Figure 1a). First, to get rid

(a) (b)

Figure 1: Modeling Problem H with a digraph.

of the circularity, we duplicate the digraph (see Figure 1b). In this digraph, the problem reduces to finding the shortest path among N pairs of vertices (marked with colored squares in Figure 1b). Finding each of these paths is a classical DP problem. However, the complexity of such solution is $\mathcal{O}(N^2M)$. To reduce this time complexity we have to look into the structure of those paths.

Let P be a shortest path from $(1, j)$ to $(N, j + M - 1)$. Similarly, let Q be a shortest path from $(1, k)$ to $(N, k+M-1)$ such that $k > j$. We say that Q *crosses* P if i) there exist vertices $(a, \ell) \in P$ and $(c, \ell) \in Q$ such that $c > a$; or if ii) there exist vertices $(\ell, b) \in P$ and $(\ell, d) \in Q$ such that $b > d$. As $k > j$, if Q crosses P, then P and Q have at least to vertices in common. An example is shown in Figure 2a. Since P and Q are shortest paths, the two subpaths between the intersection vertices have the same cost. Therefore, there exists a collection of optimal paths that do not cross (note that these paths may have vertices or arcs in common).

Thus, if we find a shortest path P from $(1, j)$ to $(N, j + M - 1)$, and a (non-crossing) shortest path Q from $(1, k)$ to $(N, k + M - 1)$ where $k > j$, it suffices to consider the vertices between the borders defined by P and Q to find a shortest path from $(1, \ell)$ to $(N, \ell + M - 1)$, for every ℓ, $j < \ell < k$. This situation is depicted in Figure 2b.

To find this collection of N non-crossing shortest paths, we can follow a Divide and Conquer strategy similar to the construction of a segment tree. First, we calculate a shortest path starting

(a) (b)

Figure 2: (a) Two crossing paths P and Q; and (b) the valid region defined by two non-crossing paths.

in $(1, 1)$. Note that, this is also an optimal path starting in $(1, N + 1)$ (by mirroring). This is like considering the interval $[1, N + 1]$. Let $m = \lfloor (N + 1)/2 \rfloor$. Then, we find a shortest path starting in $(1, m)$ having the previous two paths as borders. Now, we break the interval $[1, N + 1]$ into intervals $[1, m]$ and $[m, N + 1]$. We solve these subproblems recursively always carrying over the new borders we obtain over this process. Considering the recursion tree, the total work needed to calculate one level is $\mathcal{O}(NM)$ (the borders partition the grid). Therefore, the complexity of this algorithm is $\mathcal{O}(NM \lg N)$.

I. Help the Aztecs.
Tags : Graph algorithms.
Author : Thiago Oliveira.

The first step is to note that we do not need to answer the queries in the order they were given. So reverse all queries from the input, now we can either be asked to find a good cut or to add one vertex. We say that a cut is good if there are more good roads than bad ones, note that this is equivalent to saying at least half of the roads are good. Suppose that we already have a partition of the vertices such that there are more good roads than bad ones. When adding a new vertex, add it to the side of the partition that it has the smallest number of neighbors. This way we will add more good roads than bad ones, and since we started with a good cut we will have a good cut after adding the new vertex.

Our problem now is to find an initial good cut, so suppose we have a graph and want to find one good cut. We can start with the empty graph and add the vertices one by one with the same strategy previously described. When we add all vertices we will have a good cut. The final complexity is $O(qn + m)$.

There are alternative solutions. For example, you could start with a random partition of the vertices (sample until one is good). And start removing the vertices, if at one point the partition is not good anymore you random sample until it is good again. Although the worst case complexity of this solution is $O(q(n + m))$ it works fast enough in practice.

J. Indiana Jiang and the Temple of Kukulkan.
Tags : Graphs, ad-hoc.
Author : Enrique Junchaya.

Let $G = (V, E)$ be the graph where each vertex represents a symbol and each edge $uv \in E$ represents a lever that its connected with the symbols represented by u and v. Also, it's a bit simpler to think the problem the other way around: we have a fixed pattern as the initial symbol sequence and we want to choose a set of levers such that the final sequence is just formed by suns (zeroes). Then, let's label the vertices with a 1 if we want it to be a moon or 0 if we want it to be a sun. With this, the problem is reduced to finding a set of edges S such that, if for every $uv \in S$ we flip the labels of vertices u and v, we obtain a graph with just vertices labeled zero. If such a set of edges exists, we say that the graph has a solution.

Let's call a connected component of the graph **odd** if the sum of the labels of its vertices is odd and **even** otherwise. Since any arbitrary graph contains a set of connected components, a graph has a solution if and only if all of its components have a solution. Therefore, we just need to know how to solve the problem for a connected component.

First, notice that odd components have no solution because we always flip labels an even amount of times. This implies that the parity of the component is preserved and it's not possible to obtain a graph with vertices just labeled with zeroes.

Now, we'll show how to find a solution for an even component. Let T be any spanning tree of an even component P. We'll show by induction on $|P|$ that a solution for P that uses only edges of T exists.

If P has only one vertex, the solution for P is the empty set.

Now, assume that there exists a solution for an even component with $n \geq 1$ vertices that only uses the edges of any spanning tree of it. We'll build a solution S to an even component P with $n + 1$ vertices that only uses the edges of T, an arbitrary spanning tree of P.

Initially S is empty. There exists a leaf v of T because every tree has at least two leaves. Let e be the only edge adjacent to v. We add e to S if and only if v has label 1. In that case, we flip the endpoints of e. Notice, then, that the component $P - v$ is also even. Since $P - v$ is an even component with n vertices, there is a solution S' to $P - v$ that only uses edges of $T - v$ by induction hypothesis. Add the edges of S' to S. Finally, S is a solution to P that uses only edges of T.

K. Missing Cyan.
Tags : Implementation.
Author : Enrique Junchaya, Thiago Oliveira.

As each team needs one printed copy of the contest, we should minimize the number of teams. In order to do so, we should maximize the number of teams with 3 members. The maximum number of teams with 3 member is $\lfloor \frac{n}{3} \rfloor$ and the (possible) remaining contestants should form one more team. Therefore, the answer to the problem is $\lceil \frac{n}{3} \rceil$

L. Tourist circuits.
Tags : Graph Theory, DP
Author : Renzo Gómez.

Let $G = (V, E)$ be the graph where each vertex represents a tourist attraction, and each edge $uv \in E$ indicates that there exists a street linking the attractions represented by u and v. We denote by $d_G(u, v)$ the length of a shortest path between vertices u and v in G. Under this setting, the problem asks to find a collection of circuits $C_1, \ldots, C_r$ such that

- $V(C_i) \cap V(C_j) = \emptyset$, for $i \neq j$,
- $\bigcup_{i=1}^{r} V(C_i) = V(G)$,

and r is minimized, or report that V does not admit such partition. Since this problem generalizes the Hamiltonian cycle problem, which is NP-complete, it is expected that the structure of G plays an important role. Our first observation is the following.

Lemma 1. *Let C be a cycle in G, then $V(C)$ induces a clique.*

Proof. By induction in $k := |V(C)|$. If $k = 3$, then C is clique. In case $k = 4$, let $C = \langle u, x, v, y \rangle$. The condition

$$d_G(u, v) + d_G(x, y) \leq \max\{d_G(u, x) + d_G(v, y),\, d_G(u, y) + d_G(v, x)\}$$

implies that C is a clique. So suppose that $k \geq 5$. Now, we distinguish two cases.

Case 1: $\exists u, v \in V(C)$ such that $d_G(u, v) < d_C(u, v)$.

Choose one such pair of vertices $u, v \in V(C)$ that minimizes $d_G(u, v)$. Note that C can be seen as the union of two (edge-disjoint) paths P and Q between u and v. Next, let R be a shortest path in G between u and v. Observe that the internal vertices of R do not belong to C, otherwise, we contradict the way we chose u and v. Now, let $C_1 = P \cup R$ and $C_2 = Q \cup R$. Note that $|C_i| < |C|$, for $i = 1, 2$. Then, by the induction hypothesis, $V(C_i)$ induces a clique. Finally, for any pair of vertices $x \in V(P) \setminus \{u, v\}$ and $y \in V(Q) \setminus \{u, v\}$, we have that $\langle u, x, v, y \rangle$ is a cycle in G. By induction hypothesis, the edge $xy \in E(G)$ and, therefore, $V(C)$ induces clique in G.

Case 2: $\forall u, v \in V(C)$, $d_G(u, v) = d_C(u, v)$.

Let $u, v \in V(C)$ be vertices that maximize $d_C(u, v) = d_G(u, v)$. Note that $d_G(u, v) = \lfloor k/2 \rfloor$. Let P and Q be the paths between u and v, in C, such that $P \cup Q = C$. Observe that there exist vertices $x \in V(P) \setminus \{u, v\}$ and $y \in V(Q) \setminus \{u, v\}$ such that $d_G(x, y) = d_G(u, v) = \lfloor k/2 \rfloor$. Finally, note that

$$d_G(u, v) + d_G(x, y) > \max\{d_G(u, x) + d_G(v, y),\, d_G(u, y) + d_G(v, x)\},$$

which is a contradiction. □

Since, for any two vertices inside a nontrivial block, there is a cycle containing those vertices, the previous result implies the following.

Corollary 1. *Every block of G is a clique.*

Now, the problem reduces to deciding how to assign the cut-vertices to the blocks of G in order to construct the desired cycles. To solve this, we apply DP over the block-cut tree of G.

Let T be the block-cut tree of G. First, we root the tree at any vertex. Let $N(v)$ denote the set of children of v in T, for $v \in V(T)$. Moreover, let T_v be the subtree of T rooted at v. We observe that there are two kinds of vertices in T, the vertices that represent blocks of G, and the cut-vertices of G. To describe the DP recurrence for block vertices, we will use the variables X_v and $\overline{X}_v$ with the following meaning:

$$\begin{aligned} X_v &= \text{minimum cycle decomposition of } T_v \text{ given that } v \text{ is using its parent,} \\ \overline{X}_v &= \text{minimum cycle decomposition of } T_v \text{ given that } v \text{ is not using its parent.} \end{aligned}$$

We say that a block vertex v *uses* a cut-vertex u if u is assigned to the block represented by v. In a similar way, we define variables Y_v and $\overline{Y}_v$ for cut-vertices. Finally, whenever a certain state is infeasible, we let $X_v = +\infty$, $\overline{X_v} = +\infty$, $Y_v = +\infty$ or $\overline{Y_v} = +\infty$.

First, we consider the case in which v is a cut-vertex. Then, if v has already been used by its parent, we have

$$Y_v = \sum_{u \in N(v)} \overline{X}_u = \overline{X}(N(v)).$$

In the other case, we have to assign v to one of its children. Therefore,

$$\begin{aligned} \overline{Y}_v &= \min\left\{\overline{X}(N(v) - u) + X_u : u \in N(v)\right\}, \\ &= \min\left\{\overline{X}(N(v) - u) + X_u + (\overline{X}_u - \overline{X}_u) : u \in N(v)\right\}, \\ &= \overline{X}(N(v)) + \min\left\{X_u - \overline{X}_u : u \in N(v)\right\}, \\ &= Y_v + \min\left\{X_u - \overline{X}_u : u \in N(v)\right\}. \end{aligned}$$

Now, we consider that v is a block vertex. First, we observe the following.

a) to form a cycle, we need at least 3 vertices

b) if a block contains a non cut-vertex, we must construct a cycle with its vertices,

c) otherwise, we may assign all of its vertices to other blocks

In what follows, given a subset $U \subseteq N(v)$, we denote by $\overline{U} := \{u \in N(v) : u \notin U\}$. Then, if a block vertex v uses its parent, we have to construct a cycle and, thus

$$\begin{aligned} X_v &= 1 + \min\left\{Y(U) + \overline{Y}(\overline{U}) : U \subseteq N(v)\right\}, \\ &= 1 + \min\left\{Y(U) + \overline{Y}(\overline{U}) + (Y(\overline{U}) - Y(\overline{U})) : U \subseteq N(v)\right\}, \\ &= 1 + Y(N(v)) + \min\left\{\overline{Y}(\overline{U}) - Y(\overline{U}) : U \subseteq N(v)\right\}, \\ &= 1 + Y(N(v)) + \min\left\{\overline{Y}(U) - Y(U) : U \subseteq N(v)\right\}. \end{aligned}$$

To calculate $\min\left\{\overline{Y}(U) - Y(U) : U \subseteq N(v)\right\}$, we can create an auxiliary array D with the differences $\overline{Y}_u - Y_u$, for each vertex $u \in N(v)$. Then, we sort D in nondecreasing order. Note that to find the desired minimum, it suffices to consider each prefix sum of D.

Finally, to calculate $\overline{X}_v$, we have also to consider the case in which we assign all of the cut-vertices of v to other blocks (if it satisfies *c*)).

$$\overline{X}_v = \min\left\{\overline{Y}(N(v)), 1 + Y(N(v)) + \min\left\{\overline{Y}(U) - Y(U) : U \subseteq N(v)\right\}\right\}.$$

The overall complexity of the algorithm is $\mathcal{O}(N \lg N)$.

M. Chavo's Barrel.
Tags : Geometry
Author : Pedro Sousa.

We can use binary search to find the circle's radius. Given a guess for the radius r, you can shrink all circles by this amount. If the intersection of all three shrunken circles is not empty then r is less or equal to the correct answer. You can verify if this is the case by checking if any intersecting point of a pairs of circles is inside the third circle for every pair of circles.