ACM ICPC 2017 - Kharagpur Regional

Presentation of solutions

1.	Uniform Strings	Easy
2.	Spam Classification using Neural Net	Easy
3.	Generating a Permutation	Easy-medium
4.	Taxi Making Sharp Turns	Easy-medium
5.	Number Game	Medium
6.	Non Overlapping Segments	Medium
7.	Spanning Tree	Medium
8.	SAD Queries	Medium
9.	Chef and XOR Queries	Med-Hard
10.	Science Fair	Hard
11.	Black Discs	Hard

Problem 1
Uniform Strings

Accepted: 67

First solved by: **kgeccoders_1**
Kalyani Government Engineering College

00:03:29

Author: Praveen Dhinwa

Problem 2
Spam Classification Using Neural Net

Accepted: 67

First solved by: **Tesla**
International Institute of Information Technology, Hyderabad

00:13:05

Author: Praveen Dhinwa

P2 - Spam Classification Using Neural Net

- Calculate everything modulo 2
- $(Y_{i+1} = W_i * Y_i + B_i \ )\text{mod } 2$
- Only depends whether Y_0 is even or odd
- Count number of even and odd numbers in [minX, maxX]

Problem 3
Generating a Permutation

Accepted: at least 44

First solved by: **ground floor**
Indian Institute of Technology Madras

01:01:30

Author: Archit Karandikar

P3 - Generating a Permutation

Minimum and maximum possible values of f(P)?

Minimum: 1, 2, 3, , ... , n

Maximum: 1, n, 2, (n - 1), …

Let's take the permutation P = 1, 2, 3, … , n

If we swap n and (n - 1), we get:
P' = 1, 2, 3, … , **n**, **n - 1**.

Note that f(P') = f(P) + 1

P3 - Generating a Permutation

Now, swap **n** with **(n - 2)**.
We get f(P") = f(P') + 1.

Key observation:
If you move n from the last position to the first position, you can see the f values increase by 1 for every move.

For each number, you can determine their positions uniquely.

Complexity: O(N)

Problem 4
Taxi Making Sharp Turns

Accepted: at least 7

First solved by: **De_Dana_Dan**
National Institute of Technology Silchar
02:07:49

Author: Praveen Dhinwa and Arjun Arul

P4 - Taxi Making Sharp Turns

- Check if there are any sharp turns - O(N)
- If you find a sharp turn at point i, then one of i-1, i or i+1 have to be changed.
- 50^2 options for each of them
- O(N) to check for each possibility

Problem 5
Number Game

Accepted: at least 10

First solved by: **NDTM**
Indian Institute of Technology Guwahati

01:36:00

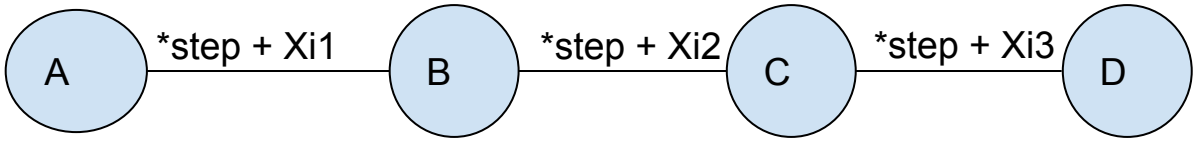
Author: Balajiganapathi

P5 - Number Game

- X_i = The number A with i^{th} digit removed
- Consider k concatenation of some X_is
- $X_{i1}X_{i2}..X_{ik}$
- $X_{i1}*10^{k(N-1)} + X_{i2}*10^{(k-1)(N-1)} +..+ X_{ik} = 0$ **mod M**
- $X_{i1}*step^{k} + X_{i2}* step^{(k-1)} +..+ X_{ik} = 0$ **mod M**

P5 - Number Game

- $X_{i1}*step^{k} + X_{i2}* step^{(k-1)} +..+ X_{ik} = 0$ **mod M**
- $((X_{i1}*step + X_{i2}) * step + X_{i3})*step + X_{i4} \ldots = 0$



- **A * step + X_i = B mod M for some i**
- Then add a directed edge from A to B
  0 < A, B < M

P5 - Number Game

- Atmost M distinct X_i
- For A in [0..M):
  For x in distinct(X_i)
  B = (A * step + x) mod M
- Find all the nodes from where 0 is reachable
- $O(M^2)$ per test

Problem 6
Non Overlapping Segments

Accepted: at least 2

First solved by: **LongTimeNoC**
Indian Institute of Technology Kanpur

01:29:08

Author: Archit Karandikar

P6 - Non Overlapping Segments

Let ***dp(i, j)*** denote the

- ***maximum number of extra segments that can still be accommodated***
- if we fix **j** segments in the first **i** segments with **i^{th}** segment fixed)

P6 - Non Overlapping Segments

Time Complexity: $O(N^3)$

dp[i][j] = min(dp[k][j-1] + distance(i to k)/R)

If **dp[i][j] > N-j** then we can fix j segments. Find max such j.

Problem 7
Spanning Tree

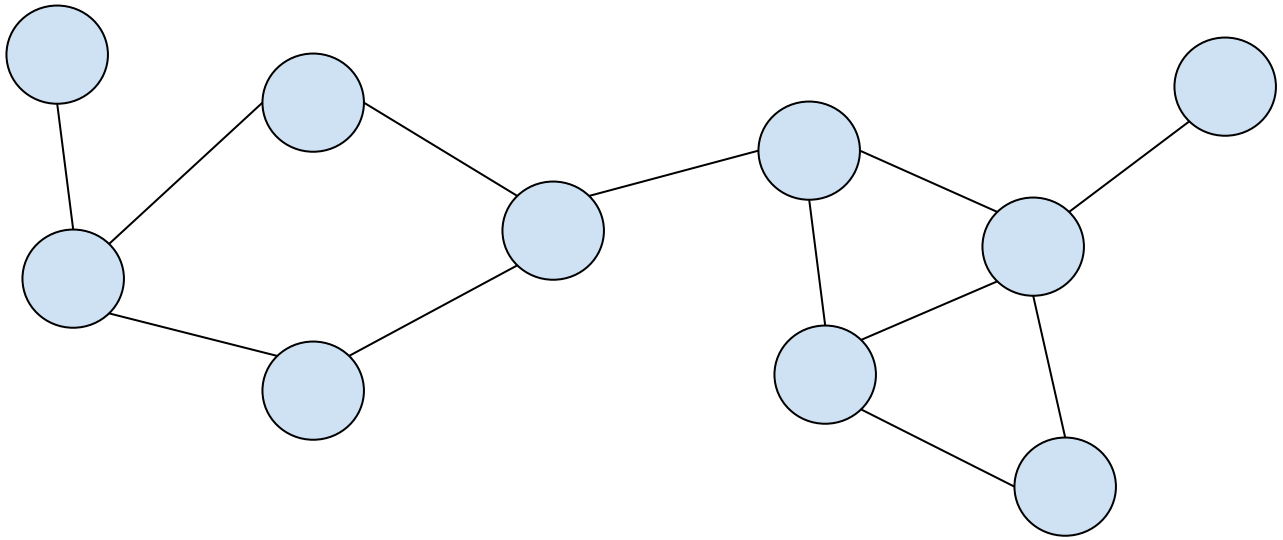
Accepted: at least 4

First solved by: **DU_Inception**
University of Dhaka

01:56:08

Author: Sidhant Bansal

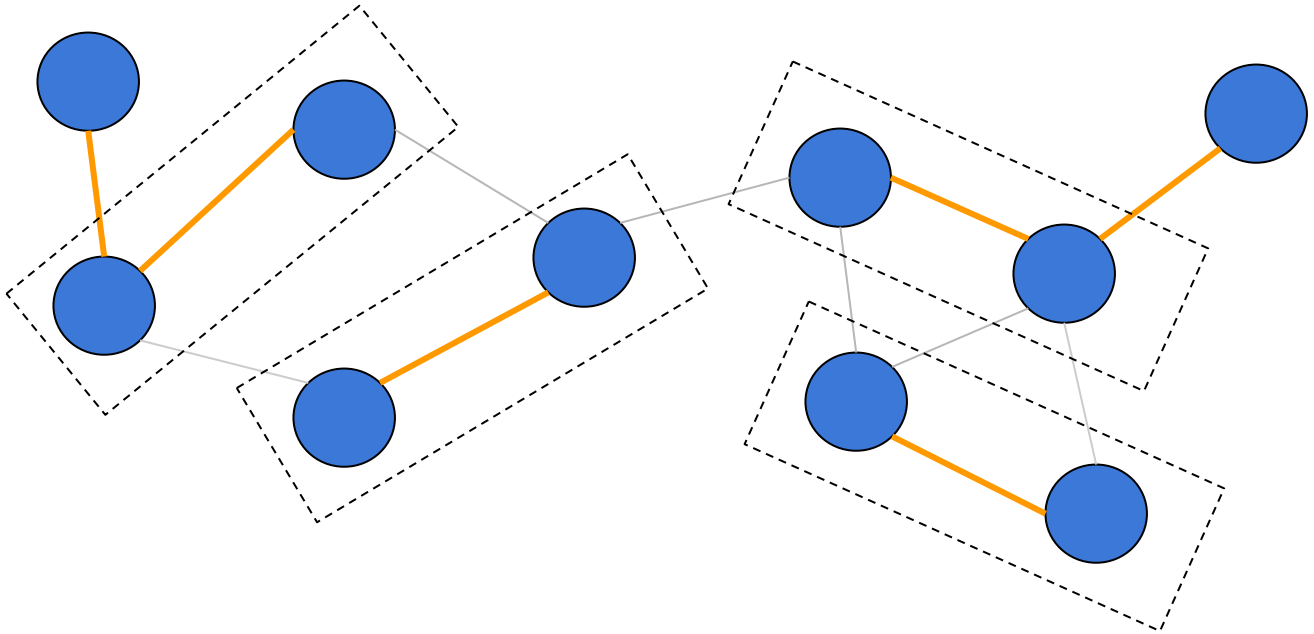
P7 - Spanning Tree



1. Let's find the nearest node for each node.

   Query: **A = {u}**, **B = {all except u}**

P7 - Spanning Tree



- We get components of size=2.
- Use DSU to maintain components.

P7 - Spanning Tree

- For the available component, we keep on querying:
  {component} vs {all except component}
- After every merge, the size of a component gets **doubled**.

How many times does each node appear in a query?

O(log(N))

Cost incurred by one appearance of a node?

1

Total cost = N*log(N) < 10^4

Problem 8
Chef and XOR Queries

Accepted: at least 4

First solved by: **Plumbus**
IIT Roorkee

01:47:43

Author: Utkarsh Saxena

P8 - Chef and XOR Queries

- Do you need the tree ?
- X_i = **XOR** of all edges from **root** to node **i**
- **F(u, v) =** XOR of path from u to v = $\mathbf{X_u \wedge X_v}$
- **Problem: Given xor values of some pair of numbers. Can you infer xor of some other pairs**

P8 - Chef and XOR Queries

- **Problem: Given xor values of some pair of numbers. Can you infer xor of some other pairs**

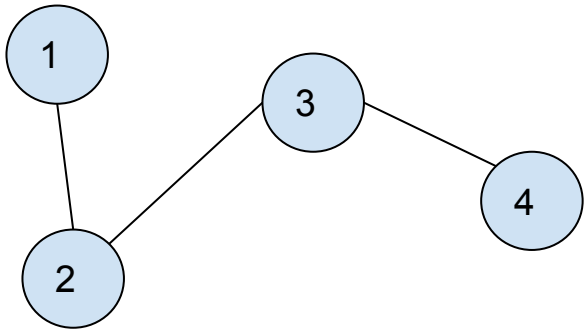
- X1 ^X2 = Y12
  X2 ^X3 = Y23
  X3 ^X4 = Y34

- What is
  X1 ^ X3? Y12^Y23
  X1 ^ X4? Y12^Y23^Y34

P8 - Chef and XOR Queries

- **Problem: Given xor values of some pair of numbers. Can you infer xor of some other pairs**

- X1 ^X2 = Y12
  X2 ^X3 = Y23
  X3 ^X4 = Y34

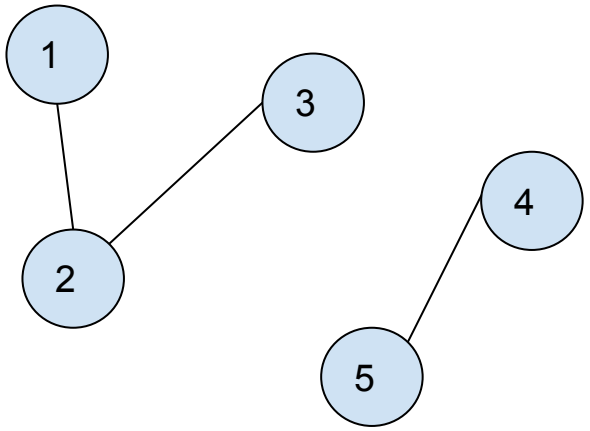


- What is
  X1 ^ X3? Y12^Y23
  X1 ^ X4? Y12^Y23^Y34

P8 - Chef and XOR Queries

- **Problem: Given xor values of some pair of numbers. Can you infer xor of some other pairs**

- X1 ^X2 = Y12
  X2 ^X3 = Y23
  X4 ^X5 = Y45



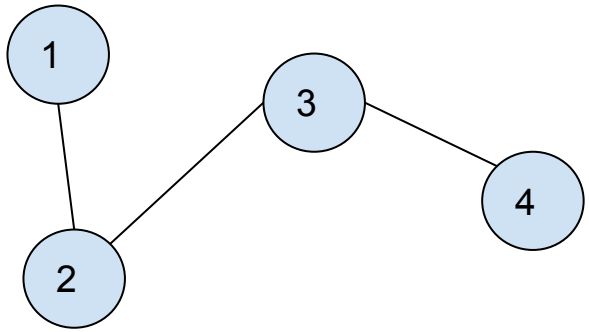
- What is
  X1 ^ X3? Y12^Y23
  X1 ^ X4**? ?? ??? You can't say**

P8 - Chef and XOR Queries

- **Problem: Given xor values of some pair of numbers. Can you infer xor of some other pairs**
- When you have information of $X_u \wedge X_v$: add an edge from **u** to **v**
- You can tell answer $\mathbf{X_i \wedge X_j}$ if and only if there is a path from **i** to **j** in this graph.
- Maintain all connected components.
- You can answer queries belonging to the same component.

P8 - Chef and XOR Queries

- **Problem:** Given xor values of some pair of numbers in same **connected component**. Can you infer xor of some other pairs.



- X1 ^X2 = Y12
  X2 ^X3 = Y23
  X3 ^X4 = Y34
- What if we assume X1 = 0.
- X2 = Y12
  X3 = X2 ^ Y12
  X4 = X3 ^ Y34
- Consistent with the given information

P8 - Chef and XOR Queries

- **Problem:** Given xor values of some pair of numbers in same **connected component**. Can you infer xor of some other pairs.

- Fix value of one node in every component(say root of dsu)

- Let Xi denote the assumed value of i^{th} node in its component.

- Query **a, b: Xa^ Xb** iff a and b in the same component.

P8 - Chef and XOR Queries

- The information stored inside a component is still consistent if we change **X[i] = X[i] ^ r** for all i in the component.

- X[u] ^ X[v] does not change if we change
  - X[u] = X[u] ^ r
  - X[v] = X[v] ^ r

P8 - Chef and XOR Queries

- **Deal with new information**

- New information only when xor of two disconnected nodes is given
- Given F(u, v) = r & u and v are disconnected

- Merge component of u and v

- Need to X[u] so that X[u] ^ X[v] = r
  Change X[u] to r ^ X[v]

- But what about the initial information in the component of u???
- Change **X[i] = X[i] ^ (r^X[v])** for all **i** in the component of **u**.

P8 - Chef and XOR Queries

- **Complexity ?**
- When we merge two components: Size of smaller component atleast doubles.
- O(N*logN + Q)

Problem 9
SAD Queries

Accepted: at least 23

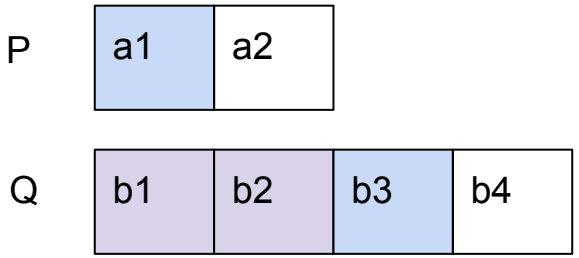
First solved by: **NDTM**
IIT Guwahati

00:53:14

Author: Archit Karandikar

P9 - SAD Queries

- Preprocess:
  - Sort all arrays
  - Maintain prefix sums for each.
- For each P[i], find largest Q[j] less than P[i]



- Q is monotonic
  - Binary search to find Q[j].
- Key observation:
  - Always keep P as the smaller array and Q as the larger array.

P9 - SAD Queries

- Contribution to total sum by P[i], Q[1..j]?
  - j * P[i] - prefix_sum_Q[j]
- Contribution to total sum by P[i], Q[j+1..s_Q]?
  - suffix_sum_Q[j+1..s_Q] - ((s_Q- j) * P[i])

- Time Complexity per query?
  - O($s_P$log(s_Q))

But will this not TLE?

P9 - SAD Queries

- Arrays of size >= sqrt(N) -> ***heavy***
- Arrays of size < sqrt(N) -> ***light***

- Key observation:
  - There can be at most O(sqrt(N)) heavy arrays.

- 3 types of queries:
  - Light-Light
  - Light-Heavy
  - Heavy-Heavy

P9 - SAD Queries

- Light-Light
  - Max possible size of P: O(sqrt(N))
  - Max possible size of Q: O(sqrt(N))
  - Worst-case complexity: O(sqrt(N)log(sqrt(N)))
- Light-Heavy
  - Max possible size of P: O(sqrt(N))
  - Max possible size of Q: O(N)
  - Worst-case complexity: O(sqrt(N)log(N))
- Heavy-Heavy
  - Max possible size of P: O(N)
  - Max possible size of Q: O(N)
  - Worst-case complexity: ??

P9 - SAD Queries

- How many times can a heavy array play the role of array P?
  - No. of possible heavy arrays Q = O(sqrt(N))
- Therefore, every element in that heavy array (P) is iterated O(sqrt(N)) times.

- Heavy-Heavy
  - Max possible size of P: O(N)
  - Max possible size of Q: O(N)
  - Worst-case complexity: O(Nsqrt(N)log(N))

- What if there are 2 heavy arrays of size N/2 each?
  - Memoize solution for each query

Problem 10
Science Fair

Submits: 0
Accepted: 0

First solved by: 0

00:00:00

Author: Utkarsh Saxena

P10 - Science Fair

- Modified Travelling Salesman Problem (***TSP is a prerequisite***)
- Cost({S1, S2,..Sk}) = TSP + [Talk(S1) * Talk(S2) * ..* Talk(Sk) * Talk(E1) * Talk(E2) .. mod 1e9+7]
- The only catch: The driver picks up every child on its path even if not mentioned(E1, E2,..) on the list.
- Cost changes if you visit an unexpected node.

P10 - Science Fair

- Sp[i][j] = shortest path between i^{th} student and j^{th} student.
- Calculate TSP dp on this matrix.
- Dp[mask] = Shortest path covering all students with bits ON in mask
- Cost of a trip(mask) = Dp[mask] + talkativeness[mask]

P10 - Science Fair

- Sp[i][j] = shortest path between i^{th} student and j^{th} student.
- Calculate TSP dp on this matrix.
- Dp[mask] = Shortest path covering all students with bits ON in mask
- Cost of a trip(mask) = Dp[mask] + talkativeness[mask]

Wrong

P10 - Science Fair: **Avoid unwanted student**

- Sp[i][j] = shortest path between i^{th} student and j^{th} student <u>THAT DOES CONTAIN ANY OTHER STUDENT</u>.

- Dp[mask][i] = Shortest path to cover only those students mentioned in mask and end the trip at i^{th} student.

- ***dp[mask|2^a][a] = min(dp[mask][b] + sp[b][a], dp[mask|2^b][a])*** over all b in mask

- Can't solve this using **TSP DP**. Contains cycle dependency.

- Solve it using **Dijkstra**

P10 - Science Fair: **How to deal unwanted students**

- cost[mask] = dp[mask] + talk[mask]

- **Given a mask of student: The driver**
  - Might want to cover more students to reduce cost
  - Might be forced to cover some other student
  - He might end up covering an optimal mask ***opt(mask)*** for this mask

- **Mask is always a subset of opt(mask)**

P10 - Science Fair: **How to deal unwanted students**

- cost[mask] = dp[mask] + talk[mask]
- **Mask is always a subset of opt(mask)**
- Random observation: *opt(opt(mask)) = opt(mask)*
- Assume omask = opt(mask)
  Given omask as the list to driver:
  - Will not cover any unwanted student.
  - cost_trip(omask) = cost[omask] = dp[omask] + talk[omask]
- cost_trip(mask) = min(cost_trip(mask2)) where *mask is a subset of mask2*

P10 - Science Fair: **How to deal unwanted students**

- cost[mask] = dp[mask] + talk[mask]
- **Mask is always a subset of opt(mask)**
- Random observation: *opt(opt(mask)) = opt(mask)*
- Assume omask = opt(mask)
  Given omask as the list to driver:
  - Will not cover any unwanted student.
  - cost_trip(omask) = cost[omask] = dp[omask] + talk[omask]
- cost_trip(mask) = min(cost_trip(mask2)) where *mask is a subset of mask2*

SOS DP

Problem 11
Black Discs

Submits: 0
Accepted: at least 0

Author: Triveni Mahatha

P11 - Black Discs

- Idea: Monte-carlo search
  - Select a bounding box (B)
  - Generate N random points within the box
  - Check how many points lie within the curve who's area is to be estimated. Let's say K of the points lie within the curve.
  - The estimated area is: (K/N) * area(B)

P11 - Black Discs

- High-level approach
  - Maintain a set of arcs - the max value of y for each x
  - For each query circle
    - Iterate over arcs
    - Add area of intersection of query circle and arc to total area
- How to uniformly generate points within a circle of radius R?
  - Let's define uniform(L, R) as a function that gives a random real number in [L..R] with uniform probability
  - Generate theta - uniform(0, 360)
  - Generate r = R * sqrt(uniform(0, 1))

Thanks!