**INOI Analysis:**

*Solutions:*<https://github.com/JVJplus/IOIPractice/blob/master/INOI%20Previous%20Papers/>

*Syntax:*

*Year:*

*Problem No: Problem Type*

*\*Comment i1*

*\*Comment i2*

2002:

2003:

2004:

1.) Recursion

\*Path backtracking in 2d array.

To identify a path, enhance the recursive search by recording the direction from which you came. For example, if your search takes you one square down, store the character 'u' at the new position to say that you reached this square from the up direction. In the end, you can follow the directions back from the bottom right square to reach the top left square.

2.) Sorting

\*Mathematical Analysis And proofing required. Evaluating/Simplifying Expression

What if a group of wrestlers have the same number of wins? How should we arrange such a group in the queue? A little analysis shows that this cannot happen. If wrestler A beats wrestler B and wrestler B beats wrestler C, then A also beats C!

To see this, let sA, sB and sC denote the strengths of A, B and C and let rA, rB and rC denote the power of their rings, respectively. If A beat B, we know that sA + rA\*sB > sB + rB\*sA. If we collect the sA terms on the left and the sB terms on the right, we have sA(1-rB) > sB(1-rA), or sA/(1-rA) > sB/(1-rB).

Thus if A beats B and B beats C, we have sA/(1-rA) > sB/(1-rB) and sB/(1-rB) > sC/(1-rC). It is immediate that sA/(1-rA) > sC/(1-rC), so A beats C.

From this it follows that we can directly sort the wrestlers in descending order by the value s/(1-r), without computing the outcomes of all N(N-1)matches.

2005:

1:) DP

2:) Sorting, Inclusion/Exclusion tech., 2 Pointers, Greedy, Encoding/Decoding

\*Tricky to come up with solution.

\*If *I < J* belong to the same set, the encoding of *I* will be smaller than the encoding of *J*. Sort the encoded values in each set *S*. For *i = 1,2,…*, let *S[i] = X* be the value at position *i* in *S* after sorting. Find a connection between the decoding of *S[i]* and *X*.

2006:

1:) Hashing, Searching, Binary Search, 2 Pointers

\*Looks like DP, but since a[i] upper bound is not mentioned so cant used dp array of unknown size.

2:) Greedy, Activity Selection Type

2007:)

1:) Full Hashing

\*Good problem to check student hashing knowledge.

\*Counting frequency can also be done by sorting.

\*https://discuss.codechef.com/questions/85945/smsdict-editorial

2:) DP, Maximum submatrix sum.

\*0 case makes it tricky, which can be overcomed by setting 0 to –INF

\*Spent very much time debugging and thinking.

2008:) //I find this year both questions really good

1:) Greedy, Merge Sort, 2 Pointers, Tricky(Not to use pair instead use 2 different array and merge)

\*Can be done by hashing/prefix sums if memory limit allows it.

\*https://www.geeksforgeeks.org/find-the-point-where-maximum-intervals- overlap/

\*https://discuss.codechef.com/questions/63350/culpro-editorial

\*coordinate or pair values problems are usually solvable by Greedy methods.

\*wow, in this problem we have converted lazy/prefix problem to 2 pointers.

2:) DP

\*At the 1st instance the problem seemed quiet difficult, but later you’ll realize the problem is just a variation of 201X ZCO Tiles Problem.

\*https://discuss.codechef.com/problems/DOMSOL

2009:)

1:) Merge Sort, D&C, Inversion Counting

\*Great Problem, How this problem is broken into problem of finding just the inversions (Inversion Count).

\*Merge Sort Implementation, so long and bugging.

\*Constraints are not given on the values of array, else I would rather had solved this using BIT/Fenwick Tree.(Easy to code!).

//UPD: Can use coordinate compression to use BIT!

\*Array needed for counting Inv. Can be generated by hashing or Binary Search

\*https://discuss.codechef.com/questions/63608/reacar-editorial

2:) DP, Game Theory?

\*At 1st instance I though this to be a game theory problem, but later realized it a DP.

2010:)

1:) Greedy

\*Problem seems to be DP, but due to high constraints its easy to prove Greedy will work.

2:)DP

\*At 1st the problem seemed very difficult.

\*But after observing the 2nd path relies on the 1st, Here you go with simple Recursion with memorization.

\*Try to think of rotation of matrix.

\*<https://www.codechef.com/problems/TWINRO>

\*https://discuss.codechef.com/questions/89963/pls-help-with-this-question

2011:)

1:) Hashing/Sorting

\*Basic Maths required ie(Permutation And Combinations)

\*No need to consider Pythagoras Theorem here as not mentioned in problem, just consider parallel to X-axis and Y-axis.

\* http://asd-justthat.blogspot.com/2011/01/inoi-2011-solutions.html

2:) DP(Hard as compared to previous papers DP), Kadene’s Algo Variation

\*Read the question properly, we need best continuous segment, don’t write recursion before proving the correctness of recurrence, this led me 2 waste 1.5hrs debugging which can be very costly during real exam.

\*Took lot of time, recursive solution consists to 4 condition and 2 base cases.

I don’t know still if some simpler solution available for this.

\*the intresting part is how to decide when to make segment end. (when further ie f(i+1) gives ans<arr[i].)

\*Malvika’s Github code is wrong. Don’t compare the outputs.

2012:)

1:) Greedy, Sorting

\*Simple Greedy Approach.

\*Good use of STL.

2:) ++Observations, Segment Tree/Square Root Decomposition, Prefix Maximum

\*It is very well explained by Sunny Aggarwal: <https://discuss.codechef.com/questions/60311/inoi-2012-problem-2-tablesum>

\*Its quiet difficult to come to above solution, so its better if know any range based data structures beforehand!

2013:)

1:) DP, Observations

\*” The game consists of one forward phase followed by one backward phase.”

Read this until you fully understand the meaning of **one forward** and **phase**.

\*The question is language is written in such a way that makes it difficult to understand at first 3-4 instances.

2:) DFS/BFS, Binary Search, STL

\*Good problem to test knowledge of DFS with binary search.

\*We can also use hashing and create a graph and run bfs.

\*or simply we can dfs starting from king and on each call count the no of elements similar with binary search to reduce time complexity.

2014:)

1:) DP

\*4 WA’s due just bcz of ignoring output in mod 20011.

\*decide base cases carefully!

2:) Graphs (Floyd-Warshall Algorithm)

* “choose a pair of cities on the network for which the cost of the cheapest route is as high as possible” understand the language meaning properly.
* This suggest we need to compare all pair and find the maximum cost among them in the cheapest path possible. So, we need nested loop I and J, and find cheapest route between I and J. So we can precalculate and find the cheapest route in 0(1)!
* Use Adjacency Matrix to represent Graph, Don’t use Adjacency List otherwise it would be quiet difficult to implement the algorithm.

2015:) //Really Difficult Problems :(

1:) ++ Observations, Very Tricky, Prefix Sums, Precalculations, Remixing/Expanding of Equations

* See here, explain very well by Udit! <https://www.commonlounge.com/discussion/840ad9acd98247d89ad5979b3a146627>

2:) Modular Exponentiation(can be skipped), DP, Inclusion/Exclusion, Observations

* Really Interesting Problem. But very scary at 1st  few instance.
* due to integer overflows and –ve in mod wasted a lot of time debugging.
* use long long everywhere in modular function, add +MOD when doing %MOD to overcome –ve
* We can overcome modular exponentiation by precalculating the values of 2^i till N. i.e: 2^i=2\*2^i(-1)
* I didn’t able to calculate the complexity, but got AC :P

2016:)

1:)