TFT: Tips and Past Problems

Alex Tung

23 April 2016

Topic change

- ~~Ad-Hoc Problems~~
- We already have:
  - Constructive Algorithm (27 Feb)
  - Interactive, Output-only and Communication tasks (9 Apr)

- So, it will be boring if I talk about Ad-Hoc Problems again today

Part 1: TFT General Tips

Basic information

- Date: 7 May
- Time: 1400 – 1900
- Venue: SHB 924 (same as HKOI Final venue)

- Please arrive at 1330 for registration and machine testing

What to expect --- Tasks

- “The paper shall consist of several programming tasks. Each task may carry different points. Each task will be further divided into one or more subtasks. Each subtask may carry different points.”

What to expect --- Tasks

- From past experience, the paper will most likely consist of 4 tasks, each carrying 100 points
- So, most likely 4 tasks, 5 hours
- For the past five years,
  - 3 x Batch problems
  - 1 x Non-batch problem

What to expect --- Tasks

- Difficulty: TFT >> HKOI Senior
- What does the above mean?

  1. It requires knowledge on many more topics to get a respectable score/position in TFT

  2. The average strength of contestants is much higher; getting an award (i.e. being an IOI/NOI HK representative) is much harder

  3. The non-batch tasks in past TFTs are generally very hard [Fact: T134 has a mean score of 0.00]

  4. There won’t be any easy tasks by HKOI standard

What to expect --- Judging

- Similar to that in HKOI 2015/16 Final

- Batch test cases (unless otherwise specified)
- Full feedback

What to expect --- Environment

- Linux, not Windows

- We will offer help on that day if you have trouble using Linux to do basic things, e.g. compile your code

Topics

- We generally follow the IOI syllabus and what we have taught in the trainings

- http://hkoi.org/en/schedule-2016/
- https://people.ksp.sk/~misof/ioi-syllabus/ioi-syllabus.pdf

- NOI participants will learn some extra algorithms (e.g. max flow) during team training

Topics

- The AM training topics are more OI-oriented and, hence, more likely to appear in TFT

Popular topics:

- Data structures (T114, T142, T151, T152)
- Dynamic programming (T112, T123, T141*, T153*)
- Graph (T111, T121, T141*, T143, T153*)

  (*): Tree DP

Topics

Other topics:

- Greedy algorithms
- Mathematics in OI
- Machine learning (probably only in non-batch tasks)
- Optimization (more like a ‘skill’ rather than a ‘topic’; same for Exhaustion, Recursion, and D&C)

- There won’t be problems testing your understanding of Miscellaneous CS Topics (IV)

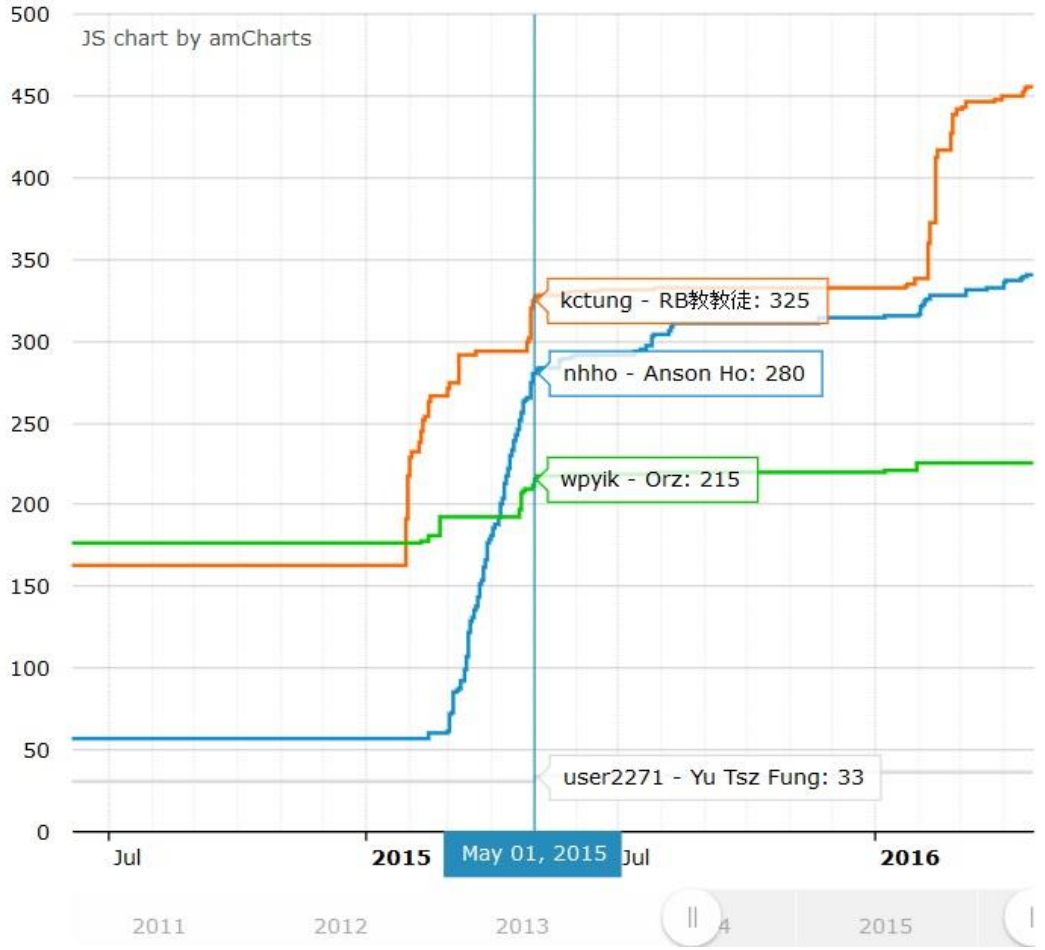
Topics

- Note that the popular/other classification is only based on the batch tasks in the past 5 years

- It does NOT predict the future (e.g. TFT 16)

What to do --- Before TFT

- Study, at least, all the AM topics
- Practise A LOT
  - Not the D- problems, of course
  - I- (IOI past papers)
  - M- (Mini-comp past papers)
  - T- (TFT past papers)
  - NOI past papers [in PEG], if you are strong enough
  - Anything (CF, USACO, COCI, CCC, ...)



What NOT to do --- Before TFT

- Work overnight the day before TFT
- Just study and not practice
- Have wrong expectations

What NOT to do --- Before TFT

Have wrong expectations, e.g.

- Expecting to ‘lose’ from the very beginning
  - NOI cutoff is ‘reachable’ with ‘good’ strategy
- Expecting to ‘win’ from the very beginning
  - Luck and condition play a part in TFT
- Expecting to get a very high score, like in HKOI
  - In 2013 and 2015, only the top 3 got a passing score (>= 200/400)
  - Even just getting 25% is not bad in the last few years

What to do --- At the start of TFT

- Carefully read ALL problems, **including the samples and subtasks**
- Ask questions if you do not understand a certain part of the problem statements

- Decide on the order of attempting subtasks
  - So, unless you are tourist, instead of 1 -> 4 -> 2 -> 3, you may need to decide on something like 1:1 -> 1:2-5 -> 4:1-3 -> 2:1 -> 4:4 -> 2:2 -> 2:3-5 [which, of course, is subject to change as the contest goes on.]

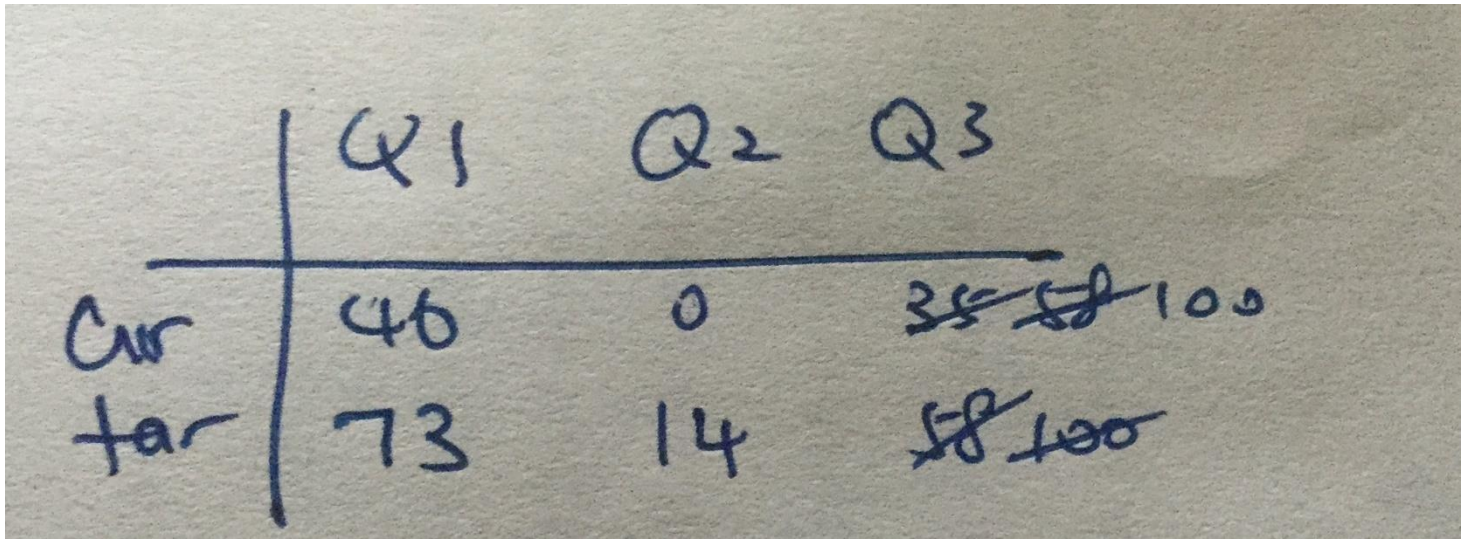
What to do --- During TFT

- In each of the last 4 TFTs (those are the only ones that I know), there exists at least a task that no one got 100 in
- Therefore, you may write something like:

```
if (n <= 1000) sub1();
else if (k == 1) sub2();
else if (k == 2) sub3();
else if (s[1] == 0) sub4();
else sub5();
```

What to do --- During TFT

- Always remember that the aim is score maximization
- For all big competitions (APIO, IOI, TFT) I did in the past three years, I made a table consisting of current score and target score on the rough work sheet



What NOT to do --- during TFT

- printf(“0\n”); and expect a positive score
- Submit the same random program many times to expect a great score increase
- Directly attempt the full solution, unless you know that the task is easy
- ‘Cheat’

What to do --- after TFT

- Stay for solutions and refreshments
- Ask around for scores and discuss ideas

- If you are in the team, you will be notified right after TFT

Any questions concerning TFT?

- You may also ask us on Facebook

Part 2: TFT Past Problems

- Today we will focus on TFT 2011 problems

- T111 – Mars Exploration
- T112 – Tetrisudoku
- T113 – Stones Rearrangement
- T114 – Current Flow

Read the problems first

- How would you order the tasks (and the subtasks) by difficulty?

- Which topic does each task cover?

The tasks, in general

- All are of medium difficulty
  - Perhaps Q3 slightly easier
- 1, 2, 4 are standard

- T111: Graph (+ data structure)
- T112: DP
- T113: Ad-Hoc (and non-batch)
- T114: Data structures

T111 – Mars Exploration

- The more experienced trainee will immediately identify this as an MST problem

- Two squares are connected by an edge of weight W = |s[x1][y1] – s[x2][y2]| iff they are adjacent and W <= K

For each query, you need to:

- Identify the number of components
- Determine the weight of the MST in each component

T111 – Mars Exploration

- Partial solution:

- Generate a new graph for each query
- Identify each component by DFS
- Run MST algorithm (Prim, Kruskal, etc.)

T111 – Mars Exploration

- Full solution:

- First, sort the queries in ascending order of K and the edges by W

- Notice that the graph in the i^{th} query and that in the $(i+1)^{th}$ query are closely related
- To go from one to another, we only need to add some edges to the original graph

T111 – Mars Exploration

- Here, we have to rely on Kruskal's algorithm, with the disjoint set union speedup
- Add edges one by one and maintain the necessary information (number of components, max. component MST weight)
- When current W = K_i, update answer for query i

- Time complexity: $O(N^2 \log N + Q \log Q)$

T112 - Tetrisudoku

- 20%: Exhaustion
  - In TFT, remember to attempt even the 20% subtask if you have no idea for the full solution
- 60%: DP
- Notice that score change only depends on the numbers on the top
- Let dp[a][b][c][d][e][f] = max. score with abcdef being the top values
- Originally, dp[0][0][0][0][0][0] = 0
- Time complexity: $O(N * 7^6)$

T112 - Tetrisudoku

- 100%: DP, with a clever optimization
- Again, let dp[a][b][c][d][e][f] = max. score with abcdef being the top values
- Notice that at least three of a,b,c,d,e,f are fixed
- Store previous reachable states in a queue
- Process one by one to update the DP values

- Time complexity: $O(N * 7^3)$

T113 – Stones Rearrangement

- Set 1: Try all permutations
- Sets 2-4: Perhaps something like bubble sort

- Ultimate target: Find the correct permutation within N guesses

T113 – Stones Rearrangement

- Observation: If we compare the number of inversions of x P and P x, where x is a number and P is a length-(N-1) sub-permutation, we immediately know the position of x
- Simply guessing
  - 1 2 3 ... N
  - 2 3 ... N 1
  - ...
  - N 1 2 ... (N-1)
- We can guess the correct permutation in N+1 guesses

T113 – Stones Rearrangement

- We still need to make one less guess to get full!
- We first guess (N-1) times
- 1 2 ... N
- ...
- (N-1) N ... (N-2)

- Thus fixing 1, 2, ..., (N-2)

T113 – Stones Rearrangement

- There are only two positions left for (N-1) and N
- Which is correct?

- Let P be the correct permutation
- Notice that we know the number of inversions required to reach P from (1 2 ... N)!
- Then, compute, for the two possible placements, the number of inversions to reach (1 2 ... N)
- The two numbers match <=> correct permutation

T113 – Stones Rearrangement

- Why the two inversion numbers match <=> correct permutation?
- <= direction: obvious
- => direction: swapping two numbers changes the **parity** of inversion. Therefore, one of them must not have the same inversion number as that from P to (1 2 ... N).

T114 – Current Flow

- You may notice that the given graph is a forest of rooted trees

- You need to find the T^{th} ancestor of Q nodes
- Notice that putting T = min(T, NM) does not change the query results; HKOI must ‘stabilize’ before time (NM)

- 50%: simulation, stop when HKOI stabilizes

T114 – Current Flow

- 100%: Using a standard data structure
- Let f[i][j] denote the (2^i)-th ancestor of node j

- f[i][j] = f[i-1][f[i-1][j]]
- All values can be found in O(NM log (NM))

- With this tool, each query takes O(log (NM))
- Time complexity: O((NM+Q) log (NM))

The end

- Work hard and good luck!