2015 ACM ICPC Asia Singapore Preliminary Contest (Online)
Regional Contest Director Report

Dr Steven Halim
School of Computing, National University of Singapore

Saturday, 12 September 2015, 1-6PM SGT

1 Introduction

This file is to document the preparation and execution of the 2015 ACM ICPC Asia Singapore **Preliminary Contest** that is conducted on Saturday, 12 September 2015, 1-6PM SGT. It will later be followed by another file to report the follow up 2015 ACM ICPC Asia Singapore **Regional Contest** later in December 2015.

WARNING: SPOILER INSIDE. For present/future ACM ICPC teams in any part of the world who want to use the 2015 ACM ICPC Asia Singapore Preliminary Contest problem set for training (available at `https://open.kattis.com`), please skip Section 5 until you finish your training.

2 Registration and Publicity Campaign

The registration period was started right after the RCD attended the RCD workshop in Marrakech, Morocco, on Monday, 18 May 2015. The registration period lasted for 3.5 months until Saturday, 29 August 2015.

From early July until end of August, we launched Facebook advertising campaign about our contest webpage `http://www.comp.nus.edu.sg/~acmicpc` to targeted Computer Science students in South East Asia aged 16-24[1]. A total of 5.6 Million people viewed our advertisements and the contest webpage received about 120000 hits during this registration period.

Meanwhile, the RCD also contacted coaches from established Universities who have joined ACM ICPC Jakarta 2014, Kuala Lumpur 2014, Vietnam National Round 2014, and Philippines National Round 2014 before about the re-appearance of Singapore site in Pacific & Indochina Peninsula sub-region this year (after 8 years of absence[2]). The RCD wishes to thank his Jakarta, Malaysia, Vietnam, and Philippines counterparts for sharing these coaches information.

[1]Unless given special permission, students that can join ACM ICPC this year must be born in year 1992 or later.
[2]The last time Singapore hosted an ACM ICPC regional was back in December 2007.

The RCD also used NUS School of Computing network of alumni to reach out to Myanmar and Brunei Darussalam Universities.

3 Accepted Teams

3.1 Overview

211 teams from 11 countries[3] are accepted for this Preliminary Contest after the registration deadline is over. The breakdown of these 211 teams are as follows:

Country	# Universities	# Teams	RCD Remarks
Brunei Darussalam	1	3	First participation in ICPC
Cambodia	1	1	First participation in ICPC
China	1	2	Outside Pacific & Indochina Peninsula sub-region
India	1	17	Outside Pacific & Indochina Peninsula sub-region
Indonesia	16	54	One University with first participation
Malaysia	6	14	
Myanmar	1	6	First participation in ICPC
Philippines	6	19	
Singapore	2	54	
Taiwan	2	2	
Vietnam	13	40	
Total	50	211	Met the target of ≥ 200 accepted teams

3.2 New Universities and Countries

We are happy to report that there are FOUR (4) new Universities joining ACM ICPC for the very first time[4]. Welcome to the ICPC family. Moreover, three (3) of them come from countries that have not been represented in ACM ICPC before. These Universities, in order of acceptance are:

1. Institut Teknologi Brunei (Brunei Darussalam), coach: Prof Poon Sheung Hung
2. University of Information Technology, Yangon (Myanmar), contact person: Prof Saw Sanda Aye
3. Zaman University (Cambodia)
4. Sampoerna University (Indonesia), coach: Mr Teddy Mantoro

We wish all these new Universities all the best in our contest and hope that you can help spread ICPC interest in you home country.

[3]There are 4 more countries in Pacific & Indochina Peninsula sub-region: Japan, Laos, South Korea, and Thailand with no representative University in Singapore site.

[4]How does the RCD determine 'New University'? A University is considered a new University if the coach cannot easily register the team in ICPC system @ Baylor University as that University name has never been stored in ICPC system before. Only coaches of Universities who contacted the RCD about this issue and subsequently the RCD asked ICPC manager to add that new University name to ICPC system are eligible.

4 Warmup Contests

We use Kattis (`https://open.kattis.com/`), the official online judge used in the recent ACM ICPC World Finals as the judging software for Singapore site.

Kattis is not well known yet prior to Singapore site this year in this Pacific & Indochina Peninsula sub-region. Therefore, we decided to run two warm up contests using past North American Qualifiers 2014 and 2013.

1. `https://open.kattis.com/contests/asiasg15prelwarmup`
2. `https://open.kattis.com/contests/asiasg15prelwarmup2`

The reception is positive with many official (and non-official) teams practising on Kattis during these two warmup contests. These two warm up contests have ≈ 150 and ≈ 230 teams joined (mostly from 211 accepted teams in Singapore site, but there are others too). These warm up contests help us iron out the technical glitches that may possibly happen during the actual Preliminary Contest.

5 Problem Set

5.1 Overview

The problem set is created with the standard ICPC problem set goals in mind:

1. All teams solve at least one problem
2. All problems are solvable by at least one team
3. No team solves all problems

5.2 Problem Analysis, Pre-Contest

The table below contains Scientific Committee (SC) **prediction** of the difficulty rating of each problem (from trivial, easy, medium, to hard) **prior** to the actual Preliminary Contest.

ID	Problem Name	Problem Type	SC Expectation
A	Tutorial	Ad hoc, Giveaway	All teams solve this problem
I	Pivot	Data Structure, Easy	Best teams solve this before B
B	2048	2D Array, Easy Tedious	(Slightly) harder to code than I
C	VisuAlgo	SSSP, DAG, DP	Easy and classic for the best teams
H	Panda Chess	DP, LCS/LIS, optz	Medium 'classic' DP problem with optz
J	Animal Classification	Parsing, Hashing	Tedious to parse, hashing is risky
F	Triangles	Math, Big Integer	Need mathematician(s)
G	Ski	Binary Search ans, DS	Segment Tree, hard to code
E	Rectangle Land	Geometry, Sweep, DS	Segment Tree again, hard
D	Grid MST	Rectilinear MST	Algorithmically hardest to derive

5.2.1 A - Tutorial

Author: Dr Steven Halim (NUS); Tester: Harta Wijaya.

This is the giveaway problem (see Section 1.2.3 of [1]) in this problem set in order to have the number of teams solving at least one problem is as close as possible to the number of accepted teams in this Preliminary Contest (211 teams). Simple if-else checks is enough to solve this problem. The only challenge is that the team has to notice that 13! and 2^{30} are both already greater than the largest $m = 10^9$. 'TLE' or 'WA' will occur if the team fails to notice this and implement factorial or power-of-two naïvely. The SC predicts the best team to solve this problem in under 10 minutes, or maybe in under 5 minutes.

5.2.2 I - Pivot

Author: Dr Steven Halim (NUS).

This is another very easy problem, especially for the more experienced teams. This problem has a bit of data structure flavor, but the solution is actually not hard. Given an array $A[0..n-1]$ with n distinct integers, a valid pivot at index i must be greater than all integers in subarray $A[0..i-1]$ (possibly empty) and smaller than all integers in subarray $A[i+1..n-1]$ (possibly empty).

We can implement this check naively in $O(n^2)$ but that will get 'TLE' as n is up to 100000. We can also use fancy Range Minimum/Maximum Query data structures (Sparse Table/Section 9.33 of [1] or Segment Tree/Section 2.4.3 of [1]) to achieve $O(n \log n)$ or $O(n)$ time complexity, respectively, but both are overkill.

A simple algorithm that performs one pass from left-to-right (keeping the running maximum) and performs another one pass from right-to-left (keeping the running minimum) is enough to answer the required Range Minimum/Maximum Query in $O(n)$.

5.2.3 B - 2048

Author: Dr Steven Halim (NUS); Tester: Harta Wijaya.

This is yet another easy problem, but attempting this problem before problem I is likely an inferior contest strategy as it is more complex to implement. The problem is simply a 2D array manipulation problem. Use symmetry to reduce the problem complexity by a factor of four (rotate the 4×4 matrix by 90, 180, or 270 degrees) so that we just need to handle one case instead of four cases.

The SC predicts that 3 problems: A, I, B, mostly in that order, will be targeted by most teams in the first one hour of the contest. After teams get 3 ACs, things will get much more challenging :)... unless they are the better teams who feel that the next few problems are still 'easy'.

5.2.4 C - VisuAlgo

Author: Dr Steven Halim (NUS); Tester: Irvan Jahja, Jonathan Irvin Gunawan.

This is a medium-hard problem for typical Computer Science students, but a very easy classic problem for experienced competitive programmers: We are asked to count the number of shortest paths from a source vertex s to a target vertex t in a standard directed weighted graph.

In most Computer Science classes, students are told that running Shortest Paths algorithms like $O((V+E)\log V)$ Dijkstra's (see `http://visualgo.net/sssp.html` and Section 4.4 of [1]) will produce a Shortest Path Spanning Tree. We want of update this notion a bit. Running Dijkstra's (with a bit of tweak) may produce a Shortest Path Spanning DAG if there are multiple shortest paths to reach a certain vertex from the source vertex s.

After we have this DAG, the problem reduces into a problem of Counting Number of Paths in a DAG with a classic DP solution (see Section 4.7.1 of [1]).

Note that one can code the solution without decomposing the problem into two like described above by simply editing Dijkstra's routine to merge the two ideas. Team that has good algorist and fast coder may actually solve this problem faster than problem B above.

5.2.5 H - PandaChess

Author: Gan Wei Liang (SG IOI); Tester: Dr Seth Lewis Gilbert, Nathan Azaria.

This is a hard problem for teams without team member that is (very) good with Dynamic Programming (DP) technique (see Section 3.5, 6.5, and 8.3 of [1]) as having only basic knowledge of DP is probably not enough to fully solve this problem.

Basically, the list of M matches will determine a total order ranking, let's call it r_{new}. This r_{new} can be found using a simple topological sort of the DAG that describes these M matches (see Section 4.2.5 of [1]). This r_{new} is likely different from the N ranking list r_{cur} (but can be equal). Note that both rankings have indices $[1..N]$, regardless of those Panda IC numbers. Your task is to transform r_{cur} into r_{new} using as minimum edits as possible and report this minimum number of edits. A delete or an insert operation is counted as one edit. Those who are familiar with Longest Common Subsequence (LCS) problem (see Section 6.5.2 of [1]) will realize that once we find the length l of the Longest Common Subsequence between r_{cur} and r_{new}, i.e. $l = LCS(r_{cur}, r_{new})$, the required answer is simply $(N - l) * 2$ edits.

The problem is how to find $LCS(r_{cur}, r_{new})$ faster than the textbook $O(n^2)$ LCS algorithm as $2 \leq N \leq 100000$. Direct implementation of textbook LCS algorithm will get TLE.

The key to pass this TLE is to realize that this ranking is a permutation. We can transforms the LCS problem into a Longest Increasing Subsequence (LIS) problem. We can solve LIS problem not just in $O(n^2)$ but in $O(n \log n)$ with help of binary search (or you can say $O(n \log k)$ where k is the length of the LIS).

5.2.6 F - Triangles

Author: Mark Theng Kwang Hui (SG IOI); Tester: Nathan Azaria.

This is a mathematics problem. First, notice that the area of a triangle can be calculated in terms of the coordinates of the vertices. This can be derived, for example, using the cross product (see Section

7.3.3 of [1]). But implementing this formula naïvely in $O(n^3)$ is impossible as $1 \leq n \leq 1000000$.

By some algebraic manipulation, we can convert the formula so that it can be calculated in $O(n)$.

Let
$a = (sum(x^2) * sum(y^2) - sum(x^2y^2)) * (n-2)$,
$b = (sum(x^2) * sum(y) * sum(y) - sum(x^2) * sum(y^2) - 2 * sum(x^2y) * sum(y) + 2 * sum(x^2y^2)$,
$c = (sum(y^2) * sum(x) * sum(x) - sum(x^2) * sum(y^2) - 2 * sum(xy^2) * sum(x) + 2 * sum(x^2y^2)$,
$d = (n-2) * (sum(xy) * sum(xy) - sum(x^2y^2))$,
$e = 2 * (sum(xy) * sum(x) * sum(y) - sum(x^2y) * sum(y) - sum(xy^2) * sum(x) - sum(xy) * sum(xy) + 2 * sum(x^2y^2))$

Then the result is $(a - b - c - d + e)/4$.

That's not all though. The last part is that the computation requires Big Integer. We test both the C++ and Java BigInteger versions during preparation. The SC predicts that strong teams with at least one very good mathematician will be able to solve this problem earlier in the contest.

Update post-contest: Alternative formula from Khor Shi-Jie (HalimArmyPlatoon1):
$S_x = \sum_{i=1}^{n} x_i$
$S_y = \sum_{i=1}^{n} y_i$
$S_{xy} = \sum_{i=1}^{n} x_i y_i$
$S_{x^2} = \sum_{i=1}^{n} x_i^2$
$S_{y^2} = \sum_{i=1}^{n} x_i^2$
$Solution = \frac{1}{4}(nS_{x^2}S_{y^2} - nS_{xy}^2 + 2S_xS_yS_{xy} - S_{x^2}S_y^2 - S_{y^2}S_x^2)$

5.2.7 J - Animal Classification

Author: Professor Sung Wing Kin, Ken (NUS); Tester: Jonathan Irvin Gunawan.

The parsing part is already quite tedious that may deter team without good coder. However, it is one of the easiest way to describe the tree input. After that, this is just a hashing problem.

The perfect hashing that guarantee no collision may be a bit difficult to think. However, any good hash function can work as long as we make it robust enough to eliminate collision (at least for the test data set up in the online judge). As this is a programming contest, you can resubmit using more optimized hashing strategy if the earlier one does not work.

Here we outline two hashing strategies:

1. Relabel the leaves of the two trees T_1 and T_2 such that, for every subtree of T_1, the set of leaves is a consecutive interval $[i..j]$.

2. For every subtree t, we get a hash value $H(t)$.
   For example, $H(t) =$ (sum of leaf-label $x \in$ subtree $t)^2 \% M$, where M is the hash table size.

Notice that in a binary tree with N leaves, there are $N-1$ internal vertices (subtrees).

3. Identify subtree $t1$ in T_1 and subtree $t2$ in T_2 such that $H(t1) = H(t2)$.
4. For each ($t1$, $t2$) where $H(t1) = H(t2)$,
   (a) Get min label, max label, and size for $t1$
   (b) Get min label, max label, and size for $t2$
   (c) If all three parameters are the same for $t1$ and $t2$, we know that the two subtree has the same set of leaves and we increase the count by one.
5. Report the count :).

Here is another possible hashing strategy: For each node in the tree, we can compute the set of animals in the subtree. We can convert a set of animals to a binary representation, with the i-th number is 1 if animal i is inside the subtree, or 0 otherwise. Therefore, for each node, we can compute the set of animals in the subtree and represent it by an integer, converting the binary representation to a base-10 integer. Let's call $f(x)$is a base-10 representation of the set of animals in the subtree where x is the root.

We can first process the first tree, and compute the $f()$s for each node. We then store all values of $f()$ using C++ map. After that, we can process the second tree and compute $f()$s for each node. For each node x in the second tree, we compute how many times $f(x)$ appears in the first tree, and we take the sum for all x.

However, the value of $f(x)$ can be as big as 2^{100000}. It is impossible to store this value even using C++ long long. It will be too slow to store this value using Java BigInteger. Therefore, we need to hash this value so it will be smaller. To reduce the probability of collision, using more than one hash function, with each one having different MOD key is a good approach.

5.2.8 G - Ski

Author: Hubert Teo Hua Kian (Stanford University); Tester: Harta Wijaya.

Note that since going eastward via ski lift is free and one is allowed to return a cabin at position X, the actual budget when $T = 1$ will be the total cost from slope 0 to slope $X + K$.

The brute force solution takes $O(N)$ for each query when $T = 1$. This is similar to the maximum sum subarray problem (see Section 3.5.2 of [1]).

The optimal solution can be divided into 3 subproblems, i.e.:

1. Update the cumulative cost when the query type $T = 0$.
   This can be done by using either Binary Indexed Tree (BIT) or Segment Tree data structure (see Section 2.4.3 or 2.4.4 of [1]). Every update takes $O(\log N)$.
2. Use binary search to find the farthest segment one can travel given the budget constraint (consider the actual budget instead of K) (see Section 8.4.1 of [1]).

3. Once we know the farthest segment one can travel, use segment tree to query the subsegment with the maximum fun value.

SC remarks: The implementation is not easy though.

5.2.9 E - Rectangle Land

Author: Loh Bo Huai, Victor (Facebook); Tester: Nathan Azaria.

This is a Computational Geometry + Data Structure problem. Brute force solution with naïve 2D array will only work if the range of x and y is just approximately 1 Million cells: 1000×1000 (so that we can use 2D array of size 1000×1000). In this problem, the range is 2000001×2000001...

One way to find out the maximum signal coverage is to use a line sweep algorithm. We can sweep from the left end to the right end, keeping track of the current signal strengths for each row. Note that we only need to update these signal strengths when we encounter a left edge or a right edge of a rectangle.

Now, what should we use to store the current signal strength? First, let's see what operations do we need. When we encounter a left edge, we need to add the rectangle's signal to the affected rows. When we encounter a right edge, we need to subtract the rectangle's signal from the affected rows. After we do all the update, we need to check the maximum signal strength, and update the current maximum accordingly. With Segment Tree (see Section 2.4.3 of [1]), all three operations can be done in $O(\log maxrow)$ or $O(\log N)$ with coordinate compression.

The final complexity would be $O(N \log maxrow + maxcol)$, or $O(N \log N)$ with coordinate compression. Either solution is accepted. That is, we can actually make this problem even harder if we had chosen this problem for the actual Regional Contest.

The SC predicts that problem G and E are quite tedious to implement correctly. The SC purposely put such problem in this online contest to minimize plagiarism cases. As the code will be relatively long, submissions from two teams that are significantly too similar will be very suspicious.

The SC also predicts that teams that choose to solve any of these two problems may likely have no time to do both and/or to also solve the hardest problem D below.

5.2.10 D - Grid MST

Author: Mark Theng Kwang Hui (SG IOI); Tester: Dr Steven Halim, Harta Wijaya.

This is one of the hardest problem in the set, designed to challenge the teams such that only the likely winner of this Preliminary Contest can solve it during contest time. The algorithmic idea is quite hard to derive on the spot but the implementation is arguably easier than problem G and E.

In short, this is a Rectilinear MST problem on a **small 2D planar grid of size** $1001 \times 1001 = 1M$ **cells**. Textbook MST algorithms (e.g. Kruskal's, Section 4.3 of [1]) will produce correct answer but will suffer from $O(n^2 \log n)$ runtime as there are $O(n^2)$ possible edges of the complete graph that describe these n points of this small 2D planar grid. As n is up to 100000, this textbook MST algorithms will be too slow.

To solve the problem, the main idea is to somehow reduce the number of edges considered as part of MST. We need to make several observations to do this. Details is purposely omitted until about next week so that contestants can think about it first.

6 Post Contest Remarks

This section is updated after the Preliminary Contest is over.

6.1 Contest Statistics

The distribution of number of solved problems is shown in the table below:

Problems Solved	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
# teams	109	47	41	53	37	30	19	16	6	5	2

This statistics is a combined statistics between official and non-official teams (as we know that the best ranked official team only solved 9 problems with 1035 penalty minutes, i.e. no official team solves all 10 problems).

There is a total of **3734** submissions throughout the Preliminary Contest. More than **eighty percent** of these submissions use C/C++ language, about **fifteen percent** of these submissions use Java, and only a mere **9** Accepted submissions using Python 2 or 3 (this language will be available in future ACM ICPC World Finals) this time.

The actual order of problems solved and the frequency of solve in this Preliminary Contest is shown in the table below. Please compare it with the Scientific Committee prediction in Section 5.2.

Time	**ID**	**Problem Name**	**Fastest to Solve**	**# AC**
5	B	2048	Azure Dragon 77	209
7	I	Pivot	HalimArmyPlatoon1	188
9	A	Tutorial	RRwatameda	232
27	C	VisuAlgo	Budweiser	117
68	J	Animal Classification	SlowBro	41
76	H	Panda Chess	BIT	71
94	E	Rectangle Land	MEGABYTE	17
104	D	Grid MST	JustBrowsing	36
133	F	Triangles	bcw0x1bd2	5
201	G	Ski	OSU	9

6.2 Teams Advancing to 2015 Asia Singapore Regional Contest

There is no plagiarism case with strong black-and-white proof found among official contestants. Therefore, the results as listed in Kattis become official results. Coaches of teams in top 20 local and top 40 foreign are invited to officially declare their intention to the RCD and pay registration fee of 120 USD per team by **12 October 2015**, i.e. one month from this Preliminary Contest. If the registration fee payment is not received by deadline, the team's place will be forfeited and the RCD will give that

place to the next highest ranked team in Preliminary Contest (or other invited teams) until all 60 slots are filled.

All advancing teams will receive an autographed copy of 'Competitive Programming 3' book [1] onsite. You can notice that the book is mentioned several times in this RCD report on purpose :).

6.3 Top 20 Local Singapore Teams

Local Singapore teams need to solve at least **4** problems with not more than **605** penalty minutes to be the top 20 local teams and advance to the regional contest. The special scoreboard is shown in this URL: `https://open.kattis.com/contests/asiasg15prel?filter=190`.

6.4 Top 40 Foreign Teams

Foreign teams need to solve at least **4** problems with not more than **513** penalty minutes to be the top 40 foreign teams and advance to the regional contest. That is, the level of local Singapore teams and foreign teams around the cut-off rankings are relatively similar. The special scoreboard is shown in this URL: `https://open.kattis.com/contests/asiasg15prel?filter=191`.

6.5 Contest Photos

The photos submitted by various team coaches can be found in this public Facebook album: `https://www.facebook.com/media/set/?set=a.10153104090657304.1073741832.725577303&type=1&l=e1aaa7e959`. Coaches and contestants are free to tag and/or share this album.

6.6 Live Commentary

The Preliminary Contest is covered live by the commentary team (Jonathan Irvin Gunawan, Nathan Azaria) using Facebook `https://www.facebook.com/jonathanirvings/posts/10204768945849459`. This post is boosted via paid advertisement to reach wider audience during the 5 hours contest.

We will do the same with the onsite Regional Contest in December and we will also discuss the problem set live as in ACM ICPC World Finals when the official contestants are competing in the contest floor with restricted Internet access. We simply cannot do so this time as the Preliminary Contest is an **online** contest.

6.7 Live Contest for Coaches and Technically Gifted Spectators

The RCD decided to open the Preliminary Contest to any other Competitive Programmers out there who wants to join or sample the problem set without any obligation or ICPC rules restrictions. The non-official scoreboard can be found in `https://open.kattis.com/contests/asiasg15prel/standings?filter=189` which has much more than 211 teams inside.

We will do the same for the onsite Regional Contest in December. That is, this time the coaches (in coach room) or any other technically gifted spectators can also try to solve the same problemset while the real 2015 ACM ICPC Singapore Regional Contest is running, same as with ACM ICPC World Finals.

6.8 Final Remarks and Acknowledgements

The RCD encourages all advancing teams to practice hard in these interim 3 months before the actual 2015 Asia Singapore Regional Contest on 9-11 December 2015. The RCD wishes everyone all the best and see you in Singapore this December.

For the rest of the teams, we hope that you keep continue practising on this problemset (now available at Kattis for further practice) and various other ICPC problemsets. Who knows it is your turn to advance next time?

For this Preliminary Contest, the RCD wishes to thank especially NUS SoC Corporate Relations (Tien, Wati) for the publicity efforts, Ket Fah for secretariat issues (registration, all those announcements and clarifications, collection of photos), all the problem authors/scientific committee/testers for preparing this nice problem set (details in Section 5), NUS SoC IT support (Musa), and Kattis team (especially Fredrik Niemelä, Per Austrin) for letting us use World Finals online judge for our Singapore site contests and their very professional support, an ACM ICPC World Finals standard :).

Lots more work will be done by the other members of the `http://www.comp.nus.edu.sg/~acmicpc/#committees` to bring these 60 advancing teams onsite to Singapore this coming December 2015.

References

[1] Steven Halim and Felix Halim, Competitive Programming 3: The New Lower Bound of Programming Contests, Lulu, 3rd edition, 2013.