2018 ACM ICPC Asia Singapore Preliminary Contest (Online) Regional Contest Director Report

Dr Steven Halim
School of Computing, National University of Singapore

Saturday, 15 September 2018, 10AM-3PM SGT

1 Introduction

This file is to document the preparation and execution of the 2018 ACM ICPC Asia Singapore **Preliminary Contest** that is conducted on Saturday, 15 September 2018, 10AM-3PM SGT. It will later be followed by another file to report the follow up 2018 ACM ICPC Asia Singapore **Regional Contest** later in December 2018.

WARNING: SPOILER INSIDE. For present/future ACM ICPC teams in any part of the world who want to use the 2018 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 soon after the RCD attended his second RCD workshop in Beijing, China, on Tuesday, 01 May 2018. The registration period lasted for a rather long period of 4.5 months until Saturday, 08 September 2018. But obviously not many registrations happened around typical University (Summer) holiday period: May-June-July-early August 2018.

From late August 2018 until early September 2018, due to rather low number of official team registration at that point of time (only 27 official teams registered by 27 August 2018 from initial target of 200+ teams), we launched another Facebook paid advertising campaign about our contest webpage `https://www.comp.nus.edu.sg/~acmicpc` to targeted Computer Science students in South East Asia aged 16-24[1].

Meanwhile, the RCD also contacted coaches from established Universities who have joined ACM ICPC Singapore previously, and coaches that he know in Indonesia, Vietnam, Malaysia, and Philippines about the re-appearance of Singapore site in Asia Southeast and Pacific Contests region this year (after 3 years of absence[2]).

[1]Unless given special permission, students that can join ACM ICPC this year must be born in year 1995 or later.
[2]The last time Singapore hosted an ACM ICPC regional was back in December 2015.

Maybe due to those aggressive approaches or simply because many Universities register their team(s) near deadline, we saw a spike of registrations around registration deadline day (Initially Saturday, 8 September 2018 but eventually extended a bit to Monday, 10 September 2018).

3 Accepted Teams

3.1 Overview

As of Tuesday, 11 September 2018, 158 teams from 33 different Universities and 9 different Countries have been Accepted[3] for this Preliminary Contest. The breakdown of these 158 teams are as follows (note that Indian and Chinese teams are **outside** the Asia Southeast and Pacific Contests region and thus will not affect the World Finals 2019 quota for this region):

Country	**# Universities**	**# Teams**	**RCD Remarks**
Singapore	2	23	Choose top 15 local teams
Indonesia	11	38	Biggest foreign teams participation
Philippines	5	18	Rotated with Singapore site
Malaysia	4	8	
Vietnam	3	8	
Taiwan	3	7	Clash with Taipei site Prelim
Thailand	1	3	Clash with Thai exams back in 2015
India	2	50	Outside Asia Southeast and Pacific
China	2	3	Outside Asia Southeast and Pacific
Myanmar	0	0	6 teams back in 2015
South Korea	0	0	1 (winning) team back in 2015
Brunei Darussalam	0	0	1 team back in 2015
Cambodia	0	0	1 team back in 2015
Japan	0	0	0 team back in 2015
Laos	0	0	0 team back in 2015
Total	33	158	Lower than target of 200 Accepted teams

4 No Warmup Contests

We use Kattis (`https://asiasg18-prelim.kattis.com/`), the official online judge used in the recent ACM ICPC World Finals as the judging software for Singapore site.

Unlike in year 2015, Kattis is now a well know online judge in Asia. NUS, the host of ACM ICPC Asia Singapore, is currently ranked first in `https://open.kattis.com/ranklist/universities`. To prepare new contestants, we decided to just encourage Accepted teams to practice on `https://open.kattis.com/problem-sources/ICPC%20SG%20Preliminary%20Contest%202015`.

[3]Unfortunately, about 12 teams have incomplete crucial registration details by deadline in such a way that the RCD cannot determine the eligibility of the teams. These 12 teams were cancelled.

5 Problem Set

5.1 Overview

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

1. ~~All~~ Most[4] teams solve at least one problem
2. All problems are solvable by at least one team
3. No team[5] 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. For this problem set, we make problem ID 'A' to be the easiest and problem ID 'J' to be the hardest (on paper). This is an experiment (we will not do the same for the actual regional contest) to see if actual Accepted submissions from the teams in the actual contest follow this pattern or not.

ID	Kattis ID	Problem Type	SC Expectation
A	knightsearch	Backtracking/DP, Knight Jump	Most teams solve this problem
B	swaptosort	CC/UFDS; Pairing	Observation; CC/UFDS to avoid TLE
C	makepalindromes	Classic DP String variant + counting	DP palindrome
D	caveexploration	Bi-connected component containing 0	Classic but rare graph theory problem
E	gridgame	Binary Search ans; Perfect MCBM	Easy and classic for the best teams
F	modulodatastructures	Sqrt Decomposition	Data Structure Problem
G	foolingaround	Pre-calculate the answers	Some teams give up with this problem
H	beehouseperimeter	Flood fill, modified DFS/BFS	Tricky to code, not Geometry
I	classicalcounting	Combinatorics, Modulo	Needs Mathematician
J	unicycliccount	Graph Theory, Counting, Modulo	The hardest problem in this contest

The authors and the testers primary programming language is C++, thus all 10 problems in this Preliminary Contest can definitely be solved with C++. However, during testing we also try our best to ensure that all problems are solvable with Java. We do not make any guarantee with Python but at least one problem is clearly solvable with Python. For each problem, we also estimate what is the expected solving time for a 'potential World Finalist' level team and the expected solving time for an 'average team'. The table below summarizes our testing process as of Friday, 31 August 2018.

[4]With all due respect, we are aware that some teams register to this prelim site with wrong expectation about the contest. Unfortunately we cannot lower the standard of the problemset and thus we are bracing for impact that the easiest problem in the set (knightsearch) may not be solvable by all teams during contest time...

[5]Actually this will be hard as we know that there are a few very strong teams in this preliminary contest; we are OK if these teams clean sweep all the 10 problems.

ID	Kattis ID	C++	Java	Python	Fastest to AC	Average to AC
A	knightsearch	AC	AC	??	5m	20m
B	swaptosort	AC	AC	??	10m	40m
C	makingpalindrome	AC	AC	??	20m	40m
D	caveexploration	AC	AC	??	20m	60m
E	gridgame	AC	AC	??	20m	60m
F	modulodatastructures	AC	AC	??	20m	80m
G	foolingaround	AC	AC	AC	30m	Not AC
H	beehouseperimeter	AC	AC	??	30m	Not AC
I	classicalcounting	AC	AC	??	45m	Not AC
J	unicycliccount	AC	AC	??	100m	Not AC
				Total time	300m (10 AC)	300m (6 AC)

5.2.1 A - Knight Search (knightsearch)

Author: Dr Steven Halim (NUS); Tester: Dr Suhendry Effendy (NUS).

This is the giveaway problem 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 (158 teams). **UNFORTUNATELY THIS DID NOT HAPPEN, as only 72 teams solved this problem out of 158 Accepted teams..., or only 46%.** This problem is a kind of String Matching in a 2D Grid (see Section 6.4.3 of [1]) but using Knight Moves (see Section 9.16 of [1]). As there are only 1 fixed starting character 'I' and 9 other characters "CPCASIASG", one can either bet that a recursive backtracking with pruning (essentially 'Depth Limited Search (see Section 8.2.5 of [1])) will pass or realize that Dynamic Programming with state (row, column, length_of_matched_char) is sufficient (distinct states is just $100 \times 100 \times 10 = 10^5$).

Note that in this problem, a cell/letter can be used more than once. In "ICPCASIASG", there are several repetitions, but the important repetitions are only the 'C's and 'I's.

1. There are 2 'A's and 2 'S's, their distance is 3 characters: "A**A" and "S**S". So there is no valid knight moves where the first 'A' will be used twice.

2. There are 2 'C's and their distance is 2 characters: "C*C". A possible test case is:

```
XSXXX
ISXAP
GXCXX
AXIXX
XXXXX
```

3. There are 2 'I's and their distance is 6 characters: "I*****I". A possible test case is:

```
XXAXX
IXAXP
XSCXX
XSXCX
GXXXX
```

5.2.2 B - Swap to Sort (swaptosort)

Author: Sean Pek Yu Xuan (NUS); Testers: Dr Steven Halim (NUS), Dr Suhendry Effendy (NUS).

First, we have to make an observation that in order to be able to say "`Yes`" (note that we purposely ask user to print out "`YES`" vs "`NO`" in knightsearch, but not here in swaptosort :O; some teams may get accidental Wrong Answer and +20 minutes penalty because of this - this is intended), the first integer must be swap-able with the last integer (via a single edge or via a path/multiple edges), second with second last, third with third last, and so on. This gives rise to a naïve $O(\frac{N}{2} \times N) = O(N^2)$ DFS/BFS algorithm. But this is TLE as N is up to 10^6.

To speed up the solution, one has to combine all CCs in the underlying graph with a DFS/BFS pre-processing routine (see Section 4.2.3 of [1]) or using the simpler Union Find Disjoint Sets (UFDS) data structure (see Section 2.4.2 of [1]) in $O(N+K)$. Then, we can run the $O(\frac{N}{2} \times 1)$ checks afterwards. This is AC.

5.2.3 C - Make Palindromes (makingpalindromes)

Author: Sean Pek Yu Xuan (NUS); Testers: Dr Suhendry Effendy (NUS).

We can construct a dynamic programming solution for this problem. Let $f(L, R, K)$ be the number of palindromic strings of length $R - L + 1 + K$ which contains $S_{L..R}$ as its subsequence. The original problem can be solved with $f(1, N, N)$, i.e. the number of palindromic strings of length $2N$ which contains $S_{1..N}$ as its subsequence. To solve $f(L, R, K)$, we have to analyze several cases similar to cases in *longest common subsequence* dynamic programming solution. If $S_L == S_R$, then $f(L, R, K) = f(L+1, R-1, K) + 25 * f(L, R, K-2)$, i.e. either take both characters (which are the same) or put another pair of characters (25 possibilities). If $S_L \neq S_R$, then $f(L, R, K) = f(L+1, R, K-1) + f(L, R-1, K-1) + 24 * f(L, R, K-2)$, i.e. either take one of the character (and put a matching one) or put another pair of characters (24 possibilities).

The base cases in which you need to pay attention to are when $S_L == S_R$ (there is one character left to take) and when $S_L > S_R$ (there are no more character left to take). In both cases, we have to check the remaining K first. If $S_L == S_R$, then K should be odd; if $S_L > S_R$, then K should be even; otherwise, simply output 0. In $S_L == S_R$ case, we have $\lceil K/2$ characters to choose to fill the remaining $K+1$ slots. Beware that $S_{L=R}$ should appear at least once among these. In $S_L > S_R$ case, we have $K/2$ characters to choose to fill the remaining K slots; it is simply $26^{K/2}$.

This dynamic programming solution has $O(N^3)$ states, and to solve a state we need $O(1)$-time, thus, total time-complexity for this solution is $O(N^3)$.

5.2.4 D - Cave Exploration (caveexploration)

Author: Dr Steven Halim (NUS); Tester: Dr Suhendry Effendy (NUS), Dr Felix Halim (Google).

This is a classic but actually rare graph problem with a rare modified DFS algorithm (see Section 4.2.8 of [1]). In summary, this problem is about finding the size of bi-connected component that contains vertex 0. We can simply run $O(V + E)$ Tarjan's algorithm to identify the bridges, and then run another $O(V+E)$ DFS/BFS from vertex 0 to count all vertices reachable from that vertex 0 if we ignore all the bridges.

We reckon that some strong teams will immediately recognize this and solve this problem in approximately 20m (most of the time is about re-typing Tarjan's DFS code without bug[6]). However, non trained teams may not be able to solve this problem at all (using $O(E \times (V+E))$ checks to determine if which edge(s) is/are bridge will be TLE)...

5.2.5 E - Grid Game (gridgame)

Author: Sean Pek Yu Xuan (NUS); Testers: Dr Steven Halim (NUS), Dr Suhendry Effendy (NUS).

This problem can be challenging for non trained teams, but doable for experienced teams. In Section 8.4.1 of [1], we discuss the Binary Search the Answer technique. Apparently, this problem can be solved that way. If we binary search the answer X, we can then construct classic bipartite graph of a grid (left set = rows; right side = columns) with edges going from left set to right set if the edge weight $\geq X$. Now we try if we can find a perfect MCBM (see Section 4.7.4 of [1]). If the answer is possible, we know that X is a valid answer and we can try reducing the answer X to a lower number. If the answer is not possible, we know that X is not a valid answer and we have to increase X to higher number. This is binary searchable. The time complexity is $O((V \times E) \log 10^6))$ in the worst case which is very doable with the given constraints.

Note that the MCBM part can been speedup from $O(V \times E)$ to $O(\sqrt{V} \times E)$ (or using greedy pre-processing), but this optimization is not necessary for this problem. We can also use Max Flow based solution to find the MCBM value, but this is much longer to code. Alternatively, we can also use some kind of backtracking with pruning to test the possible/not possible status of answer X, but this is a bit harder to analyze.

5.2.6 F - Modulo Data Structures (modulodatastructures)

Author: Sean Pek Yu Xuan (NUS); Tester: Dr Felix Halim (Google), Dr Steven Halim (NUS).

Implementing the solution verbatim (do queries of type $T=1$ in $O(N)$ and do queries of type $T=2$ in $O(1)$) is TLE as it can be made to run in $O(Q \times N)$ by having many type $T=1$ queries.

However, if one knows the square-root decomposition technique (not yet written in [1] but will be added in CP4), this problem becomes easy. We decompose the array Arr into $\sqrt{N} \times \sqrt{N}$ buckets. For the largest $N = 200\,000$, $\sqrt{200\,000}$ is just 447 (note that N does not have to be necessarily a perfect square number).

For each query of type $T=1$ with $B \leq \sqrt{N}$, we just update one cell: `bucket[B][A] += C` in $O(1)$. Otherwise if $B > \sqrt{N}$, we do `Arr[j] += C` for each $j \in [A, A+B, A+2B, ...]$ and stop when $j > N$ (as $B > \sqrt{N}$, this loop will be just $O(N/\sqrt{N}) = O(\sqrt{N})$, which is a major improvement compared to the verbatim implementation above).

[6]Note that copy pasting a well written Tarjan's DFS code and modify it to suit this poblem will save about 5m of typing time compared to official teams who (are encouraged to) retype everything. This is very hard to check in an online contest like this.

Now, we can answer each query of type $T = 2$ also in $O(\sqrt{N})$ time by combining values from `Arr[D]` (this is $O(1)$) and sum of `bucket[B][D%B]` for each $B \in [1..\sqrt{N}]$ (this is $O(\sqrt{N})$). Notice the reverse time complexity compared to query of type $T = 1$. We will get the correct answer again and have a fast enough solution.

The SC predicts that the first 6 problems: A, B, C, D, E, F, mostly in that order, will be targeted by most (strong) teams in the first two hours of the contest. After teams get 6 ACs, things will get much more challenging :)... Unless they are the stronger teams who feel that the next few problems are still 'easy'.

5.2.7 G - Fooling Around (foolingaround)

Author: Sean Pek Yu Xuan (NUS); Tester: Dr Felix Halim (Google), Dr Steven Halim (NUS).

The title of this problem is actually a hint... :).

One needs to actually run a rather slow generator algorithm (ours terminate in a few minutes, but not in a few seconds): Sieve of Eratosthenes to generate primes below 10^9, generate list of 'one less than a prime number', and to realize that there are only 379 values of N where Bob wins. So, we can precompute this :).

Note that as the final answer is just a precomputation, even a slow Python code will pass the required time limit.

5.2.8 H - Bee House Perimeter (beehouseperimeter)

Author: Dr Steven Halim (NUS); Tester: Dr Felix Halim (Google), Dr Suhendry Effendy (NUS).

This is probably not as widely known, but there is actually a nice and simple mapping from a hexagonal grid into a 2D grid, see below for the transformation of $R = 3$. Notice that we give sentinels (-1) to first/last row/column to simplify this problem. Do you notice the required pattern by observing the transformation below?

```
-1 -1 -1 -1 -1 -1 -1
-1  1  2  3 -1 -1 -1
-1  4  5  6  7 -1 -1
-1  8  9 10 11 12 -1
-1 -1 13 14 15 16 -1
-1 -1 -1 17 18 19 -1
-1 -1 -1 -1 -1 -1 -1
```

After you reach that transformation (other ways may exist), then the next step is just a matter of counting how many times a cell from 'out-of-house' touches the boundary of a cell of Alice's house. The number of such occurrences is the required perimeter that we have to count. We can simply do $O(R \times R)$ DFS/BFS (to 6 directions per cell) from any one of the sentinel (-1/out-of-house).

5.2.9 I - Classical Counting (classicalcounting)

Author: Sean Pek Yu Xuan (NUS); Tester: Dr Suhendry Effendy (NUS).

First, let us ignore the fact that the modulus in this problem (i.e. $10^6 + 7$) is NOT a prime number; we will come back to that later. This is a counting problem which can be solved with *k-combination* ("n choose k") and *k-multicombination* ("n choose k" with repetition) combined with the *inclusion-exclusion principle.* Let say M is very large (e.g., ∞) or there is no bound on how many copies can be taken from an object, then this problem is simply a k-multicombination $\left(\!\binom{N}{K}\!\right)$ which can be solved with k-combination, i.e. $\left(\!\binom{N}{K}\!\right) = \binom{N+K-1}{N-1}$.

To handle the bound M, we can use inclusion-exclusion principle. Start with $\left(\!\binom{N}{K}\!\right)$. Substract the result with the number of combination in which one object is chosen $> M$ times. Add back to the result the number of combination in which two objects are chosen $> M$ times. Substract the result with the number of combination in which three objects are chosen $> M$ times. Add back to the result ..., and so on. The number of combination in which i objects are chosen $> M$ times can be computed with $\binom{N}{i} \times \left(\!\binom{N}{K-i\cdot(M+1)}\!\right)$. The first term corresponds to the number of choosing i out of N objects, while the second one corresponds to the number of choosing $K - i \cdot (M + 1)$ out of N objects with repetition—the first $i \cdot (M + 1)$ objects are already "distributed" to the selected i objects, causing them to exceed the M bound. Therefore, in general, the solution is:

$$\sum_{i=0..\infty} -1^i \times \binom{N}{i} \times \left(\!\binom{N}{K - i \cdot (M+1)}\!\right)$$

This equation can be computed with:

$$\sum_{i=0..\infty} -1^i \times \binom{N}{i} \times \binom{N + K - 1 - i \cdot (M+1)}{N-1}$$

The next challenge is how to quickly compute $\binom{N}{i}$ and $\binom{N+K-1-i\cdot(M+1)}{N-1}$ for all i. Observe that M can be as low as 1, causing i to have $O(N)$ different values, thus, computing **each** binomial in $O(N)$-time will cause our solution to be *TLE* with large N. Notice that in both terms, there is a fixed value, i.e. N in $\binom{N}{i}$, and $N - 1$ in $\binom{N+K-1-i\cdot(M+1)}{N-1}$. We can exploit this to compute the value of both terms for **all** i in $O(N + K)$ and store them in an array. Hints: $\binom{n}{k} = \binom{n}{k-1} \times (n - k + 1)/k$ and $\binom{n}{k} = \binom{n-1}{k} \times n/(n - k)$.

Now, back to the modulus issue. The given modulus $10^6 + 7$ is a composite number ($29 \times 34\,483$) while the constraint for N and K can be as large as 10^5. We can combine *chinese reminder theorem (CRT)* with *Lucas theorem* to handle the *modular multiplicative inverse* in the equation above. Break down the problem into two prime moduli, 29 and 34 483, solve them separately (meaning, you have to do all the above twice, each with a different modulus), and then combine the result with CRT to get the result modulo 1 000 007. As the moduli are smaller than the largest N and K, the number whose modular multiplicative inverse we want to compute may not be coprime with a modulus, e.g., when $N - 1 \geq 29$. Note that Euler's theorem requires the numbers to be coprime. To handle this, we can break down the k-combination which we want to compute with Lucas theorem, i.e.

$\binom{n}{k} \equiv \prod_i \binom{n_i}{k_i} (\bmod\ p)$, where n_i and k_i are the expansion of n and k in base-p.

The time-complexity for this approach is $O((N+K)\log(N+K))$. Note that this complexity is not tight, but it is not worse than this.

5.2.10 J - Unicyclic Count (unicycliccount)

Author: Sean Pek Yu Xuan (NUS); Tester: Dr Suhendry Effendy (NUS).

First, we observe that the a unicyclic graph consists of N vertices and N edges. Moreover, the edges can be oriented such that each vertex has one outgoing edge. This is done by first considering all the vertices of degree 1 in the unicyclic graph. The edges connecting to them can only point away from the "leaf". Repeating this procedure, we will result in orientiering all the non-cycle edges. The cycle can be oriented in exactly two ways (provided the cycle has at least 3 edges). Henceforth, we shall call all graphs with N vertices and N edges a candidate graph.

Next, we would like to compute the number of connected candidate graphs. This is done via a dynamic program on subsets of vertices. Let $F[S]$ and $G[S]$ be the number of connected candidate graphs and (not-necessarily connected) candidate graphs using the vertices in the bitmask S. Then, from the previous paragraph discussion, $G[S]$ is the product of degree of vertices in induced subgraph of S. This can be done effectively via bitmasks. We have to be more careful regarding $F[S]$ to avoid double counting. Let the smallest labelled vertex in S be u, then

$F[S] = \sum_{u \in T \not\subseteq S} F[T] * G[S \setminus T]$.

Now that we have the connected candidate graphs, there is another obstruction. It is possible for the cycle to be of length 2, hence the same edge chosen twice. This is a tree. We count the number of trees via Kirchoff Matrix-Tree Theorem (or Dynamic Programming). Each tree can occur $N-1$ times as each edge in the tree can be doubled. Subtracting this, we have double the answer as our cycle can be oriented "clockwise" or "anti-clockwise".

All in all, by carefully doing the operations modulo M (this affects the division at the end as well as the finding of determinant of matrix) and ensuring the no redundant states in our dynamic programming on subsets (this is done via bit manipulations). We result in an algorithm of time complexity $\Theta(N^3 + 3^N)$.

We reckon that this is likely the hardest problem in this prelim contest.

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 (will be updated after result is unfrozen):

Problems Solved	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
# teams	86 :(	16	13	11	7	5	4	7	4	4	1

This statistics is a combined statistics between official and non-official teams.

There is a total of **1235** submissions throughout the Preliminary Contest. Most of these submissions use C/C++ language. We have about 16 Java AC submissions and only 4 Python 2/3 AC submissions.

The actual order of problems solved and the frequency of solve in this Preliminary Contest is shown in the table below. As the order is not 'A', 'B', ..., 'H', 'I', 'J'. But at least the first two 'A', 'B' are quite correct and last three 'H', 'I', 'J' are in correct order. Please compare it with the Scientific Committee prediction in Section 5.2. This table will be updated after result is unfrozen:

Time	**ID**	**Kattis ID**	**Fastest to Solve**	**# AC**
8m	B	swaptosort	I See the One (ITB)	61
8m	A	knightsearch	ITBNUS (Binus)	72
11m	F	modulodatastructures	Pandamiao (NUS)	28
17m	E	gridgame	3body2 (NUS)	33
35m	D	caveexploration	414 (NUS)	45
47m	G	foolingaround	Pandamiao (NUS)	9
51m	C	makingpalindromes	THREE (VNU)	20
52m	H	beehouseperimeter	3NationsIOI (NUS)	29
57m	I	classicalcounting	Bitset (VNU)	13
188m	J	unicycliccount	Platelet (SJTU)	5

6.2 Teams Advancing to 2018 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 15 local and top 45 foreign are invited to officially declare their intention to the RCD and pay registration fee of 250 SGD per team by **15 October 2018**, 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, 2018b revised edition' book [2] onsite. You can notice that the book is mentioned several times in this RCD report on purpose :).

6.3 Top 14 Local Singapore Teams

Local Singapore teams need to solve at least **2** problems with not more than **116** penalty minutes to be the top 14 local teams and advance to the regional contest.

Total	Rank	University Name
10	2,3,4,7,12,13,14,15,19,21	National University of Singapore
3Sophomores, Send Bobs to Alice, Pandamiao, 3body2, power harder Sultan Halim, ReFreshPHD, 3NationsIOI, scrub++, NEWBIE (6 other NUS teams at rank 22, 23, 29, 30, 48, 51 are not sent: 414, Illumina, lorem ipsum, ICanProgramC++, LGD, BLS		
4	10,34,41,47	Nanyang Technological University
1T, ntunoobs II, H1N4, NTU Walnut (2 other NTU team at rank 66, 68 are not sent: Code_Age, PlusUltra		

6.4 Top [30..40]?? Foreign Teams

Due to only 75 out of 158 teams solved at least 1 problem in this preliminary contest and a few top Universities only sending their best (out of 3/4/7/16 qualified teams), then foreign teams need to solve only **1** problem to be the top [30..40]? foreign teams and advance to the regional contest... This leaves the RCD with about 10 wildcard slots that can be given to anyone else who will register to Asia Singapore 2018 before deadline.

Total	Rank	Country	University Name
2	1,11	China	Shanghai Jiao Tong University
Platelet, Skyrim (both)			
1	6	Vietnam	Vietnam National University
Bitset (4 other VNU teams at rank 5,8,17,18 are not sent: ONE, THREE, TWO, FOUR)			
4?	9,16,20,28	Taiwan	National Tsing Hua University
Teletubbies, NTHU_5734k, Made in Abyss(Jinkela), NTHU_leporidae			
1	24	Indonesia	Bina Nusantara University
ITBNUS (6 other BINUS teams at rank 38, 40, 43, 46, 50, 54 are not sent: Aeroflot, YoBiLiNUS, BeNUS, PPTI_SABER, Lucero, DoReMi)			
2?	25,55	Malaysia	International Islamic University Malaysia
Fellas, Tyros			
2	26,37	Taiwan	National Chiao Tung University
NTCU_Jaguar, NCTU_Kemono			
3?	27,45,60	Philippines	De La Salle University
Convex Hull, Blackjack, Panic			
5?	31,32,53,56,69	Philippines	Ateneo de Manila University
L^3, Nutritious Nilaga w/ Saba, Happy BST Friends, $\lvert CBN\rvert^2$, O(b^2)			
2?	33,35	Indonesia	Institut Teknologi Bandung
I See the One, Arurange Code Party			
1	36	Indonesia	Gadjah Mada University
Sayata Kid Prim's Sieve			
2?	39,64	Philippines	University of the Philippines - Diliman
Quiwarriors 1, Quiwarriors 2			

Total	Rank	Country	University Name
1	52	Taiwan	National Taiwan Normal University
Love and Peace			
2?	57,70	Indonesia	STMIK Mikroskil
Gae Bolg, Numpang Lewat			
3?	58,71,75	Vietnam	Eastern International University
CDK, ADN, PSV			
2?	59,67	Thailand	Kasetsart University
CSKU-PITLORD, CSKU-SIMANGKALO			
2?	61,74	Indonesia	Universitas Multimedia Nusantara
Qubit, WeBareBears			
1	≈64	Indonesia	Parahyangan University
Anonymous Wombat (submit code to asiasg18-prelim-open instead)			
2?	62,72	Malaysia	University of Malaya
tacocat, IM26C4U			
1?	65	India	RMK Engineering College
TryToBeatUs			
1?	73	Philippines	Malayan Colleges Laguna
MCLCCIS2018B			

6.5 Contest Photos

The photos submitted by various team coaches (only show teams who solve at least 1 problem) can be found in this public Facebook album: `https://www.facebook.com/media/set/?set=a.10155746944362304&typ` `1&l=a511e1d72f`. Coaches and contestants are free to tag and/or share this album.

6.6 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/asiasg18-prelim-open`.

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 2018 ACM ICPC Singapore Regional Contest is running, same as with ACM ICPC World Finals.

6.7 Final Remarks and Acknowledgements

The RCD encourages all advancing teams to practice hard in these interim 3 months (15 September 2018 - 13 December 2018) before the actual 2018 Asia Singapore Regional Contest on 12-14 December 2018. 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 open 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 the problem authors/testers/scientific committee for preparing this nice problem set (details in Section 5), NUS SoC Corporate Relations

(Tien, Wati, Christine) for the publicity efforts, Dean's Office (Mrs Ho) for the local food @ NUS, NUS SoC IT support (Musa), and Kattis team (Greg Hamerly, Fredrik Niemelä) for letting us use World Finals online judge for our Singapore site contests (again).

Lots more work will be done by the other members of the `https://www.comp.nus.edu.sg/~acmicpc/#committees` to bring these 50+ advancing teams onsite to Singapore this coming December 2018.

References

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

[2] Steven Halim and Felix Halim, *Competitive Programming 3: 2018b Revised Edition*, Not published yet, 3.18b, 2018.