Overview of the PAN@FIRE 2020 Task on the Authorship Identification of SOurce COde

Ali Fadel[a], Husam Musleh[a], Ibraheem Tuffaha[a], Mahmoud Al-Ayyoub[a], Yaser Jararweh[a], Elhadj Benkhelifa[b] and Paolo Rosso[c]

[a]*Jordan University of Science and Technology, Jordan*
[b]*Staffordshire University, UK*
[c]*Universitat Politècnica de València, Spain*

Abstract

Authorship identification is essential to the detection of undesirable deception of others' content misuse or exposing the owners of some anonymous malicious content. While it is widely studied for natural languages, it is rarely considered for programming languages. Accordingly, a PAN@FIRE task, named Authorship Identification of SOurce COde (AI-SOCO), is proposed with the focus on the identification of source code authors. The dataset consists of crawled source codes submitted by the top 1,000 human users with 100 correct C++ submissions or more from the CodeForces online judge platform. The participating systems are asked to predict the author of a given source code from the predefined list of code authors. In total, 60 teams registered on the task's CodaLab page. Out of them, 14 teams submitted 94 runs. The results are surprisingly high with many teams and baselines breaking the 90% accuracy barrier. These systems used a wide range of models and techniques from pretrained word embeddings (especially, those that are tweaked to handle source code) to stylometric features.

Keywords

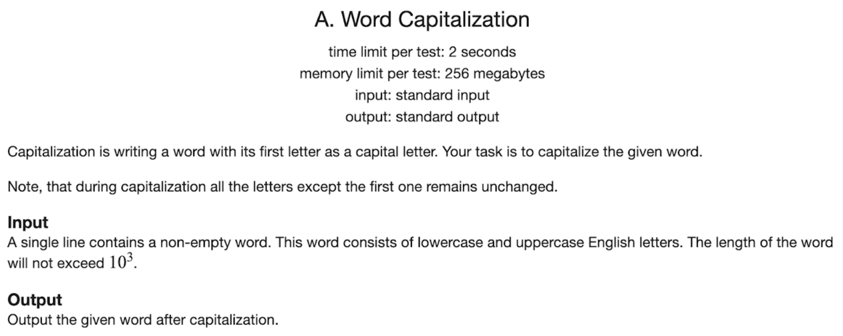
authorship-identification, source-code, datasets

1. Introduction

After the success of the previous tasks on source code such as SOurce COde re-use (SOCO) [1], Cross-Language SOurce COde re-use (CL-SOCO) [2], and Personality Recognition in SOurce COde (PR-SOCO) [3] , a new task is proposed in this paper. Specifically, we describe the **A**uthorship **I**dentification of **SO**urce **CO**de (AI-SOCO)[1] task, one of the tracks of the 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020).[2] In the following sections, we define the task, discuss the dataset we introduced to the research community, present the available baselines and results, and highlight the achieved accomplishments.

FIRE'20: Forum for Information Retrieval Evaluation, December 16–20, 2020, Hyderabad, India
aliosm1997@gmail.com (A. Fadel); husam.sa3@gmail.com (H. Musleh); bro.t.1996@gmail.com (I. Tuffaha); malayyoub@gmail.com (M. Al-Ayyoub); yijararweh@just.edu.jo (Y. Jararweh); E.Benkhelifa@staffs.ac.uk (E. Benkhelifa); prosso@dsic.upv.es (P. Rosso)
https://github.com/AliOsm/ (A. Fadel)
0000-0002-3447-8331 (A. Fadel); 0000-0001-9372-9076 (M. Al-Ayyoub)

[1]https://sites.google.com/view/ai-soco-2020
[2]http://fire.irsi.res.in/fire/2020


A. Word Capitalization

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Capitalization is writing a word with its first letter as a capital letter. Your task is to capitalize the given word.

Note, that during capitalization all the letters except the first one remains unchanged.

Input

A single line contains a non-empty word. This word consists of lowercase and uppercase English letters. The length of the word will not exceed 10^3.

Output

Output the given word after capitalization.

Figure 1: CodeForces Problem Sample.

2. Task Definition

AI-SOCO focuses on identifying the author who wrote a given piece of code out of a predefined set of authors. This facilitates solving issues related to cheating in academic, work and open source environments. The detection of cheating in academic communities is significant to properly assess the contribution of students or researchers. Moreover, in work environments, credit sometimes goes to people who did not deserve it. Such issues of plagiarism could arise in open source projects that are available on public platforms. Similarly, systems developed for the AI-SOCO task can be used in public or private online coding contests, whether done in coding interviews or in official coding training contests, to detect the cheating of applicants or contestants. Such systems can also play a significant role in detecting the source of anonymous malicious software.

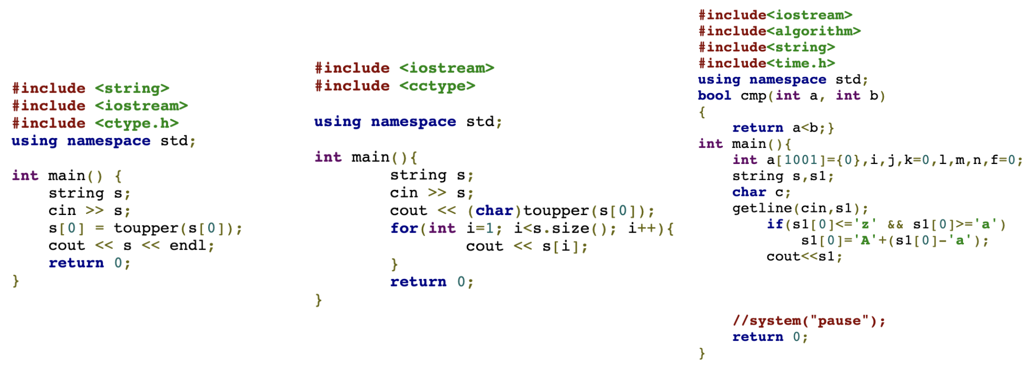
3. Dataset and Evaluation

The dataset is composed of source codes collected from the open submissions in the CodeForces[3] platform. CodeForces is an online judge for hosting competitive programming contests, where each contest consists of multiple problems. The problems' difficulty (aka index) ranges from A to Z, where A is the easiest and Z is the hardest. The majority of contests have 5 to 7 problems (A-G). So, the occurrence of problems with indexes bigger than G is very rare. Figure 1 shows an example of a CodeForces problem, while Figure 2 shows a sample of the possible solutions by three different CodeForces users.

A user can solve a problem by writing a solution for it using any of the available programming languages on the website, and then submitting the solution through the website. The solution is tested automatically against a set of test cases and its result can be correct (accepted) or incorrect (wrong answer, time limit exceeded, etc.).

For our dataset, we selected the top rated 1,000 users (according to the website's rating system) and collected 100 source codes per user. This gives a total number of 100,000 source codes. All collected source codes are correct and written using the C++ programming language. For each user, all collected source codes are from unique problems and were submitted within the

[3]https://codeforces.com



```cpp
#include <string>
#include <iostream>
#include <ctype.h>
using namespace std;

int main() {
    string s;
    cin >> s;
    s[0] = toupper(s[0]);
    cout << s << endl;
    return 0;
}
```

```cpp
#include <iostream>
#include <cctype>

using namespace std;

int main(){
        string s;
        cin >> s;
        cout << (char)toupper(s[0]);
        for(int i=1; i<s.size(); i++){
                cout << s[i];
        }
        return 0;
}
```

```cpp
#include<iostream>
#include<algorithm>
#include<string>
#include<time.h>
using namespace std;
bool cmp(int a, int b)
{
    return a<b;}
int main(){
    int a[1001]={0},i,j,k=0,l,m,n,f=0;
    string s,s1;
    char c;
    getline(cin,s1);
        if(s1[0]<='z' && s1[0]>='a')
            s1[0]='A'+(s1[0]-'a');
        cout<<s1;



    //system("pause");
    return 0;
}
```

Figure 2: CodeForces Problem Solutions Sample.

Table 1
AI-SOCO Dataset Statistics

Users Count	1,000	Unique Problems	6,553
Solutions Count	100,000	Maximum Solutions/Problem	61
Tokens Count	22,795,141	Minimum Solutions/Problem	1
Whitespaces Count	46,944,461	Avg. Solutions/Problem	15.26
Unique Tokens	1,171,991	Avg. Solutions/Codeforces Index	2,439.02
Avg. Solutions/User	100	Median Solutions/Problem	12
Avg. Tokens/Solution	227.95	Unique User Countries	78
Avg. Whitespaces/Solution	469.45	Avg. Solutions/User Country	1,282.05
Maximum Tokens in a Solution	10,189	Minimum Tokens in a Solution	3

smallest possible time period to avoid the possibility of users changing their coding styles over time. The dataset was split randomly with 50% used for training, 25% used for development and the remaining 25% used for testing. The training, development and unlabeled testing datasets are available through our GitHub repository.[4] The full version of the dataset (with testing dataset labels) is available through Zenodo.[5] Table 1 shows some statistics about the dataset, while Figure 3 shows the solutions distribution over CodeForces indexes and Figure 4 shows the solutions distribution over C++ compiler versions.

For a multi-class classification problem like ours, many evaluation metrics are suggested in the literature. However, since our dataset is completely balanced, we are evaluating the performance of the systems with accuracy since it is a simple and easily interpreted metric.

4. Baseline Approaches

In this section, we present the baseline approaches we developed for this task along with their results. All implementations mentioned in this section are available on our GitHub repository (See Footnote 4). Table 2 shows all baselines results. The pretrained and fine-tuned models are

[4]https://github.com/AliOsm/AI-SOCO
[5]https://zenodo.org/record/4059840

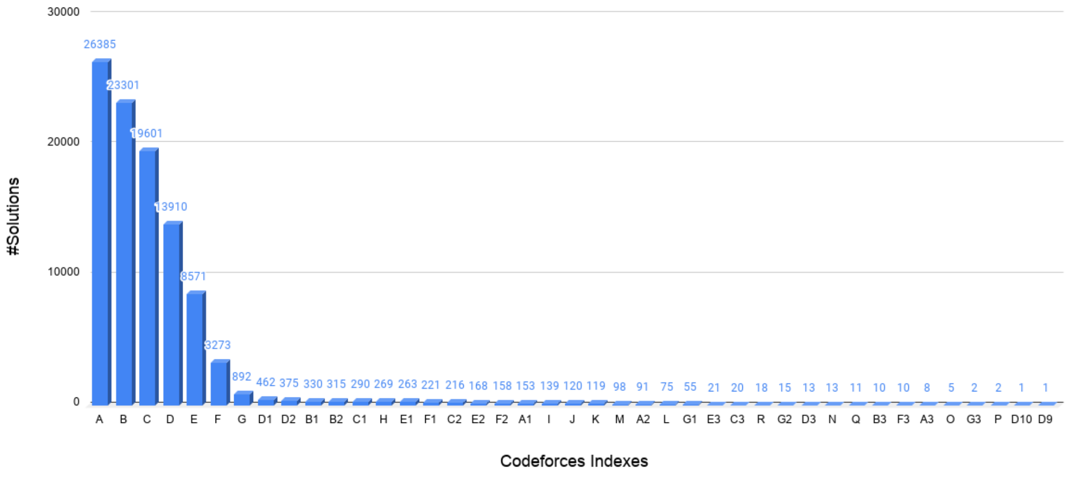


Figure 3: Number of Solutions per CodeForces Index.

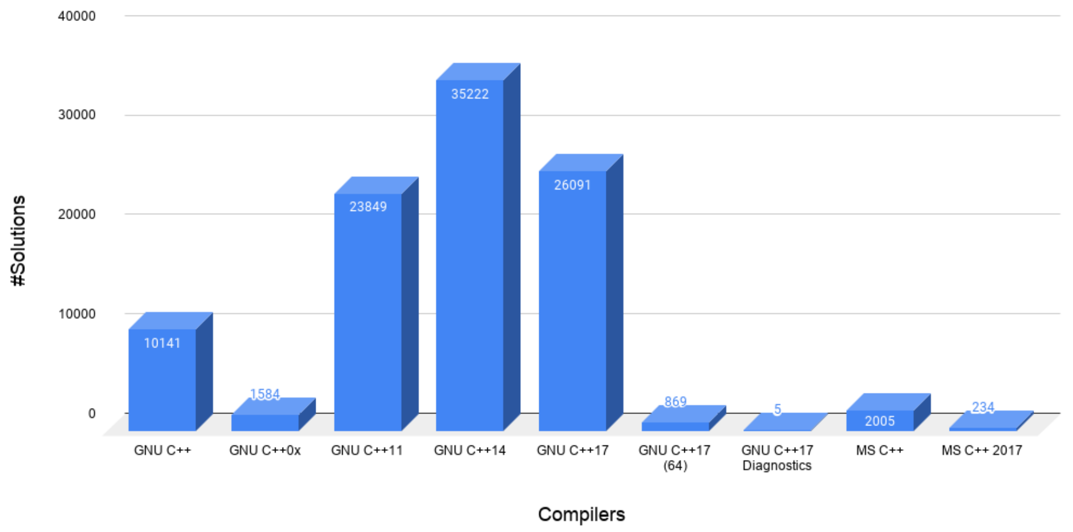


Figure 4: Number of Solutions per C++ Compiler.

available through HuggingFace models hub.[6]

4.1. Random Baseline

This baseline predicts a random author for each piece of code from the list of 1,000 authors. Its expected accuracy is 0.1%. We used this baseline to put a threshold for systems, whether they

[6]https://huggingface.co/models?search=ai-soco

Table 2
AI-SOCO Baselines Results on Development and Testing Datasets

Baseline Name	**Development**	**Testing**
Random	~0.1%	~0.1%
Characters Logistic	29.25%	29.92%
TF-IDF KNN	62.13%	62.78%
C++ RoBERTa Tiny	87.66%	87.46%
C++ RoBERTa Tiny-96	91.12%	91.02%
C++ RoBERTa Small	**93.19%**	**92.88%**

are better or worse than the randomness.

4.2. Characters Logistic Baseline

To build a code representation, this baseline converts each source code into a vector that represents the count of the 100 printable characters. Then, it builds a logistic regression [4] model on the vectorized representations. It achieved an accuracy of 29.25% and 29.92% on the development and testing datasets, respectively.

4.3. TF-IDF KNN Baseline

In this baseline, we tried to improve the code representation by vectorizing the source codes using term frequency-inverse document frequency (TF-IDF) [5] with 10K features. These features are fed into a K-Nearest Neighbors (KNN) [6] classifier with $K = 25$. Its accuracy is 62.13% and 62.78% on the development and testing datasets, respectively. This is a significant improvement over the previous baseline. One downside of this approach is efficiency. This lazy learner baseline is very slow in producing its predictions. It took about four hours to predict all examples in the development or testing datasets.

4.4. C++ RoBERTa Tiny

We pretrained a RoBERTa [7] model with a single Transformer [8] layer and 12 attention heads using the concatenation of the training dataset source codes. The model trained on Google Colab [9] platform[7] with 8 TPU cores for 200 epochs, 32×8 batch size, 512 max sequence length and Masked Language Model (MLM) objective. Other parameters were set to their default values as mentioned in the run_mlm.py script provided by the HuggingFace Transformers [10] package.[8] To tokenize the source codes, we trained a byte level BPE tokenizer [11] from the HuggingFace Tokenizers package[9] with vocabulary size equals to 30K. Before tokenization, each four continuous spaces were converted into a single tab character (\t). After that, the model was fine-tuned on a classification task to predict the source code author out of the predefined list of the 1,000 authors. The fine-tuning was done on Google Colab platform using V100 GPU for 10

[7]https://colab.research.google.com
[8]https://github.com/huggingface/transformers
[9]https://github.com/huggingface/tokenizers

epochs, 32 batch size and 512 max sequence length. Source codes longer than 512 were truncated while fine-tuning. Finally, the model achieved 87.66% and 87.46% accuracy on development and testing datasets, respectively.

4.5. C++ RoBERTa Tiny-96

To do more experimentation, we tried to increase the number of attention heads in the model discussed in Section 4.4 from 12 to 96 attention heads. The model was trained and fine-tuned using the same pretraining and fine-tuning procedures as the previously mentioned one. The only change was in the batch size, where we set it to 16×8 instead of 32×8 for training and 16 instead of 32 for fine-tuning. The model achieved 91.12% and 91.02% accuracy on development and testing datasets, respectively.

4.6. C++ RoBERTa Small

To compare the effect of increasing the number of attention heads vs. increasing the number of Transformer layers, we trained another model with 6 Transformer layers and 12 attention heads. This model follows C++ RoBERTa Tiny pretraining and fine-tuning procedures as well. It achieved 93.19% and 92.88% accuracy on development and testing datasets, respectively.

5. Competition Results

The competition was hosted on the CodaLab[10] platform. Surprisingly, 60 teams registered for the competition, 14 of which made 94 runs on the Evaluation and Post-Evaluation phases. The participating teams' accuracy results were higher than expected, ranging from 74.52% up to 95.11% on the testing dataset.

5.1. Overview of Participating Teams

In this section, we discuss the details of the teams' submissions that were disclosed by the time this paper is written. Table 3 shows the results of all submissions.

We start by discussing the submissions made by the deadline.

1. **Team UoB [12]:** The best approach in the competition was proposed by the Team UoB. The authors proposed several approaches and the best one was built using byte-level n-grams. Each source code is encoded into a vector representing the 20,000 most commonly occurring byte-level n-grams in the training dataset, where $n = 6$. Each n-gram in the vector was represented by a binary count, with 1 indicating the n-gram was present and 0 indicating the n-gram was absent in the given source code. The training dataset n-gram vector representations were then used to train a densely connected neural network classifier, which could then be used to predict authorship of the development and testing datasets. Their approach achieved 95.41% and 95.11% accuracy on the development and testing datasets, respectively.

[10]https://competitions.codalab.org/competitions/25148

2. **Team Yang1094 [13]:** According to the official results, the second best approach is Team Yang1094. Similar to some of the models of Team UoB [12], Team Yang1094 used an approach that relies on word-level and character-level n-gram features that are used to train logistic regression classifier. The character-level n-gram features (where $n = 2, \ldots, 7$) filtered using term frequency-inverse document frequency (TF-IDF) gave the best results for this team with an accuracy of 94.28%.
3. **Team Alexa [14]:** Team Alexa proposed an ensemble of Naïve Bayes (NB) models and pretrained models. Specifically, their best model is a weighted combination of Multinomial NB (MultinomialNB), Bernoulli NB (BernoulliNB), and two versions of CodeBERTa. For the NB models, their features are the top 30,000 character-level n-grams (where $n = 1, \ldots, 5$) with TF-IDF, whereas the CodeBERTa versions differ only in the learning rate. The resulting accuracy is 93.36%.
4. **Team LAST [15]:** This team's submissions are all based on the KNN procedure with $K = 1$. As for the preprocessing and feature extraction steps, they first used an indentation-aware tokenization followed by extracting token-level n-grams (where $n = 1, \ldots, 4$) and skip-grams (1 for each 3-gram and 3 for each 4-gram). Then, they used a binary coding of the presence of the features and relevance frequency as a supervised term weighting scheme along with L2 normalization. The best accuracy reported for this team is 92.19%.
5. **Team FSU_HLJIT [16]:** Inspired by earlier work on plagiarism detection and microblog filtering, this team approached the task at hand as a ranking problem. The best performing system from this team ranked the source codes according to the number of occurrences of the character level n-grams (where $n = 15$). Despite its simplicity, this system got an accuracy of 91.57%.
6. **Team UMUTeam [17]:** This team's top system consisted of a combination of char-level n-grams and other author's traits features. The former were combinations between 1 and 8 length applying TF-IDF with Sublinear TF scaling. On the other hand, the author's traits included (1) the average length of code blocks and comments, (2) regular expressions to detect languages other than English, such as Arabic, Indian, or Russian; as well as (3) characters employed in the creation of ASCII-art. In addition, since some source codes included contact data such as nicknames, URLs, or Twitter accounts, the UMUTeam also compiled a Bag of Words (BOW) composed of the tokens that followed the words *name* and *author* in the block comments. Once all the features were compiled, they were filtered by applying a feature selection consisting of discarding low-variance features and the models were trained using a Random Forest (RF) classifier. The result is an accuracy of 91.16%.
7. **Team Abdalrhman:** This team used word-level and character-level n-grams with some preprocessing and cleaning steps for the dataset. For the preprocessing step; they normalized numbers into a special token then cleaned all lines with length greater then 50 or less than 4 characters. As for extracting the features, they used word-level and character-level (n-grams from text inside word boundaries) using TF-IDF features, (where $n = 2, \ldots, 6$) for both word-level and character-level. Then the features are stacked from both vectorizers and fed into Linear Support Vector Classification (LinearSVC) which achieved an accuracy of 90.88%.

8. **Team bits_nlp_2020 [18]:** This team's top system used Word2Vec embeddings along with a Convolutional Neural Network (CNN). The word2vec embeddings were trained from scratch. After concatenating vectors of both Continuous BOW (CBOW) and skip-gram for each word, the authors used the resulting embeddings for their CNN. The submitted system consisted of an ensemble of five such models, with different hyperparameters. They assigned a result for each test case based on whichever model had the highest confidence. The resulting accuracy was 90.64%.
9. **Team gaojiaming:** This team did not disclose the details of its submitted systems, which achieved an accuracy of 86.16%.
10. **Team Chanchal [19]:** This team used an abstract syntax tree (AST) for the tokenization of the code. Then, they applied TF-IDF to word-level and character-level n-grams to generate the code representation. Finally, different classifiers were used. Their best model used word bigrams on a stacked model. The stacked model is an ensemble of extra tree classifier, RF classifier, and XG-Boost classifier. The achieved accuracy is 82.95%.
11. **Team Kode_Stylers [20]:** This team used what they call "naturalness" of code as the key to their solution. They used different methods to obtain features such as tokenization, n-gram TF-IDF, warning messages, and coding styles. They used these features to train different classifiers, such as RF and Transformer. Then, they applied an ensemble approach to get better results. The best result obtained by this team is 82.08%.
12. **Team meghana:** This team did not disclose the details of its submitted systems, which achieved an accuracy of 81.2%.

After the competition's deadline, two teams made interesting submissions. Their details are as follows.

- **Team Twist Bytes:** This team did not disclose the details of its submitted systems, which achieved an accuracy of 94.4%. This result is the second best result reported for this task.

- **Team SSNCSE_NLP [21]:** This team used character count vectorization and TF-IDF to extract features that are then fed into an RF classifier. The resulting accuracy is 85.73%.

5.2. Discussion and Error Analysis

There are some common characteristics among all participating teams. For example, they have all explored the use of different versions of n-grams. In most cases, n-gram features proved to be very powerful achieving the best (or near-best) results. Moreover, byte-level and character-level n-grams seem more promising than word-level n-grams. As for classification, different models showed strong performance individually and within ensembles. Actually, many teams have reported that ensembling only improves the results by a small margin. Finally, deep learning models and pretrained models showed good performance, but they do not emerge as clear winners like the case for many authorship analysis tasks [22, 23]. This applies to general models, such as RoBERTa, as well as models customized for handling source code, such as CodeBerta.

Looking at the predictions made by the participating systems, one can see certain patterns and trends. Appendix A shows five "easy" samples that each participating system managed to

Table 3
Summary of the results of the participating teams. The asterisk sign (*) is used to indicate baseline approaches while the pound sign (#) is used to indicate teams which made their submissions after the deadline.

Rank	Team	Accuracy
1	UoB	95.11%
2	Yang1094	94.28%
3	Alexa	93.36%
*	C++ RoBERTa Small Baseline	92.88%
4	LAST	92.19%
5	FSU_HLJIT	91.57%
6	UMUTeam	91.16%
*	C++ RoBERTa Tiny-96 Baseline	91.02%
7	Abdalrhman	90.88%
8	bits_nlp_2020	90.64%
*	C++ RoBERTa Tiny Baseline	87.46%
9	gaojiaming	86.16%
10	Chanchal	82.95%
11	Kode_Stylers	82.08%
12	meghana	81.20%
*	TF-IDF KNN Baseline	62.78%
*	Characters Logistic Baseline	29.92%
*	Random Baseline	0.08%
#	Twist Bytes	94.40%
#	SSNCSE_NLP	85.73%

predict correctly, whereas Appendix B shows five "hard" samples that none of the participating system managed to predict correctly. Easy samples tend to exhibit easily detectable unique patterns, such as the use of `typedef` and `#define`, whereas hard examples tend to be short or use patterns that are common among many authors. We performed a simple intersection between the participating systems' predictions and discovered that 17,501 test samples out the 25,000 test samples (i.e., 70% of the test set) were correctly predicted by all systems. On the other hand, only 589 test samples (i.e., 2.36% of the test set) were never predicted correctly by any system. This indicates that the problem might not be as difficult as we initially expected.

To support this conjecture, we studied the participating systems' predictions at the user level. We found out that our test set, has 133 "very easy" users out of 1,000 users. These users have such an easily identifiable coding styles that all participating systems managed to predict all of their testing cases correctly. One might think that having 13.3% very easy users in the test set is not that bad. However, if we consider slightly less easier users, i.e., ones with prediction rates of 90% or more across all participating systems, we end up with 603 users. Going to lower prediction rates (e.g., 80% rate of 70% rate) covers a vast majority of the users (83.1% and 93.2%, respectively).

On the other hand, very few users are difficult to identify. For example, the most difficult user, User 579, had a prediction rate of only 35.14% across all participating systems. Other difficult users with prediction rates lower than 50% include User 198 (prediction rate 36%), User 998

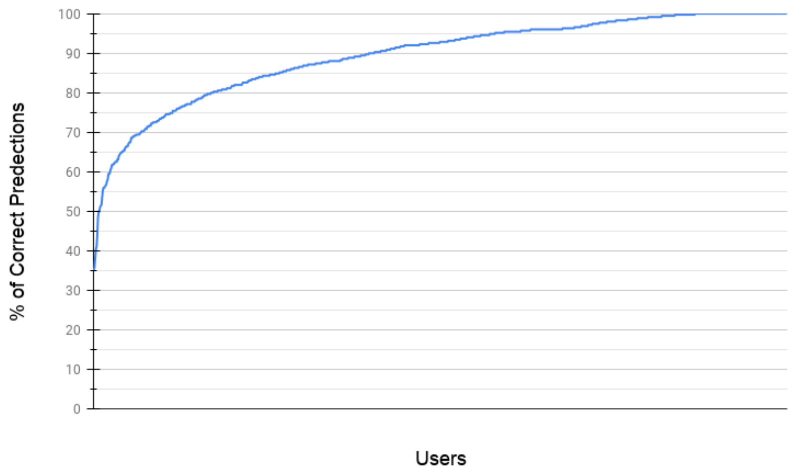


Figure 5: Percentage of Correctly Predicted Problems per User.

(prediction rate 38.29%), User 945 (prediction rate 40.57%), User 563 (prediction rate 41.14%), User 780 (prediction rate 44%), and User 205 (prediction rate 48.57%). Nonetheless, the number of hard users is rather low, which justifies the relatively high accuracy levels achieved by all participating systems. Figure 5 shows the percentage of correctly predicted samples per user.

Finally, instead of looking at this issue from the sample-level or the user-level, we look at it from the problem-level. It turns out that, for most problems, all participating systems managed to correctly identify the code authors. For very few problems, none of the participating systems managed to correctly identify any of the code authors for these problems. Figure 5 shows the percentage of systems that predicted samples of problems correctly.

Acknowledgments

The work of the last author was partially funded by the Spanish MICINN under the research project MISMISFAKEnHATE on MISinformation and MIScommunication in social media: FAKE news and HATE speech (PGC2018-096212-B-C31); as well as by the European Cooperation in Science and Technology under the COST Action 17124 DigForAsp.

References

[1] E. Flores, P. Rosso, L. Moreno, E. Villatoro-Tello, PAN@FIRE: Overview of SOCO track on the detection of SOurce COde re-use, in: Notebook Papers of FIRE 2014, 2014.

[2] E. Flores, P. Rosso, E. Villatoro-Tello, L. Moreno, R. Alcover, V. Chirivella, PAN@FIRE: Overview of CL-SOCO track on the detection of Cross-Language SOurce COde re-use., in: Post Proceedings of the Workshops at the 7th Forum for Information Retrieval Evaluation,

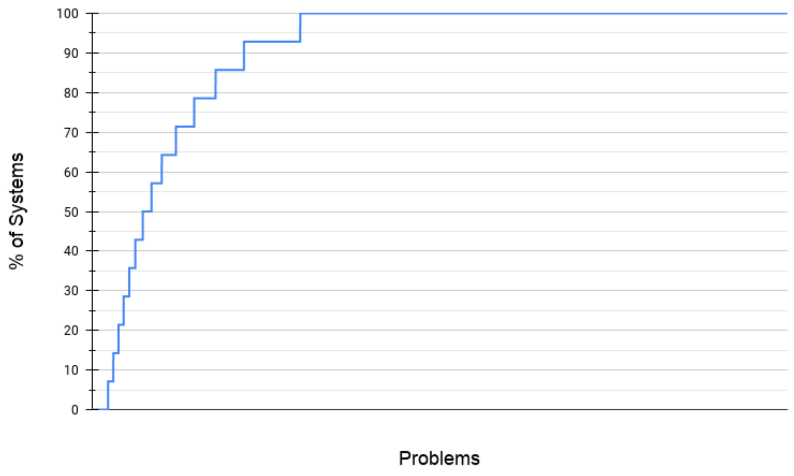


Figure 6: Percentage of Systems that Predicted Problems Correctly.

Gandhinagar, India, December 4-6, 2015, volume 1587 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015, pp. 1–5.

[3] F. Rangel, F. González, F. Restrepo, M. Montes, P. Rosso, PAN@FIRE: Overview of the PR-SOCO track on Personality Recognition in SOurce COde, in: Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation, Kolkata, India, December 7-10, 2016, volume 1737 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016, pp. 1–15.

[4] C.-Y. J. Peng, K. L. Lee, G. M. Ingersoll, An introduction to logistic regression analysis and reporting, The journal of educational research 96 (2002) 3–14.

[5] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information processing & management 24 (1988) 513–523.

[6] G. Guo, H. Wang, D. Bell, Y. Bi, K. Greer, Knn model-based approach in classification, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2003, pp. 986–996.

[7] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[9] T. Carneiro, R. V. M. Da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, P. P. Reboucas Filho, Performance analysis of google colaboratory as a tool for accelerating deep learning applications, IEEE Access 6 (2018) 61677–61685.

[10] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al., Huggingface's transformers: State-of-the-art natural language processing, ArXiv (2019) arXiv–1910.

[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are

unsupervised multitask learners, OpenAI blog 1 (2019) 9.

[12] A. Crosby, H. T. Madabushi, UoB at AI-SOCO 2020: Approaches to Source Code Classification and the Surprising Power of n-grams, in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[13] Y. Yang, L. Kong, Z. Han, Y. Han, H. Qi, N-gram-based Authorship Identification of Source Code, in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[14] M. Bni Younes, N. Al-Khdour, Team Alexa at Authorship Identification of SOurce COde (AI-SOCO), in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[15] Y. Bestgen, Boosting a KNN Classifier by Improving Feature Extraction for Authorship Identification of Source Code, in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[16] Z. Han, T. Li, X. Wang, Y. Xu, M. Wu, Z. Li, Z. Wu, Y. Han, Ranking-based and Classification-based Approaches for Code Author Identification, in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[17] J. A. García-Díaz, R. Valencia-García, UMUTeam at AI-SOCO'2020: Source Code Authorship Identification based on Character N-Grams and Author's Traits, in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[18] A. V. Mandalam, Abhishek, Embedding-based Authorship Identification of Source Code, in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[19] C. Suman, A. Raj, S. Saha, P. Bhattacharyya, Source Code Authorship Attribution Using Stacked Classifier, in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[20] P. Sriiesaranusorn, S. Wattanakriengkrai, T. Son, T. Tanaka, C. Wiraatmaja, T. Ishio, R. G. Kula, Kode_Stylers: Author Identification through Naturalness of Code: An Ensemble Approach, in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[21] N. N. A. Balaji, B. Bharathi, SSNCSE_NLP@Authorship Identification of SOurce COde (AI-SOCO) 2020, in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[22] W. Daelemans, M. Kestemont, E. Manjavacas, M. Potthast, F. Rangel, P. Rosso, G. Specht, E. Stamatatos, B. Stein, M. Tschuggnall, et al., Overview of pan 2019: Bots and gender profiling, celebrity profiling, cross-domain authorship attribution and style change detection, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2019, pp. 402–416.

[23] M. N. AlRashdan, M. Abdullah, M. Al-Ayyoub, Y. Jararweh, Authorship analysis of english and spanish tweets, Proceedings of the Association for Information Science and Technology 57 (2020) e261.

A. Easy Samples

Listing 1: Easy Sample 1 | Problem ID: 99992 | User ID: 814

```cpp
#include <bits/stdc++.h>
typedef long long ll;
typedef long double ld;
#define pb push_back
using namespace std;


int n, s;
int last[1010];
int main(){
  ios::sync_with_stdio(0);
  cin.tie(0);

  cin >> n >> s;
  for(int i = 0; i < n; i++){
    int f, t;
    cin >> f >> t;
    last[f] = max(last[f], t);
  }
  int t = 0;
  for(int i = s; i >= 1; i--){
    if(t < last[i]) t = last[i];
    t++;
  }

  cout << t << endl;

  return 0;
}
```

Listing 2: Easy Sample 2 | Problem ID: 99452 | User ID: 742

```cpp
#include <bits/stdc++.h>

using namespace std;

//using ll = int64_t;
using ll = long long;
using ull = uint64_t;
using i32 = int32_t;
using u32 = uint32_t;
using i64 = int64_t;
using u64 = uint64_t;
using pii = pair<int, int>;
using pll = pair<ll, ll>;
using ld = double;

#define X first
#define Y second
```

```

#ifndef ONLINE_JUDGE
#define FWRITE
#endif

namespace io
{

#ifndef FWRITE
#include <unistd.h>
#endif
  const int BUFSIZE = 1 << 20;

  int isize, osize;
  char ibuf[BUFSIZE + 10], obuf[BUFSIZE + 10];
  char* is, * it, * os = obuf, * ot = obuf + BUFSIZE;

  char getchar()
  {
    if (is == it)
    {
      is = ibuf;
#ifdef FWRITE
      it = ibuf + fread(ibuf, 1, BUFSIZE, stdin);
#else
      it = ibuf + read(STDIN_FILENO, ibuf, BUFSIZE);
#endif
      if (is == it) return EOF;
    }
    return *is++;
  }

  char getalpha()
  {
    char c = getchar();
    while (!isalpha(c)) c = getchar();
    return c;
  }

  void putchar(char c)
  {
    *os++ = c;
    if (os == ot)
    {
#ifdef FWRITE
      fwrite(obuf, 1, BUFSIZE, stdout);
#else
      write(STDOUT_FILENO, obuf, BUFSIZE);
#endif
      os = obuf;
    }
  }

  int inp() {
```

```
  int x = 0, f = 0; char ch;
  for (ch = getchar(); !isdigit(ch); ch = getchar())
  {
    if (ch == EOF) return -1;
    if (ch == '-') f = 1;
  }
  for (; isdigit(ch); x = x * 10 + ch - '0', ch = getchar());
  return f ? -x : x;
}

ll inp_ll() {
  ll x = 0; int f = 0; char ch;
  for (ch = getchar(); !isdigit(ch); ch = getchar())
    if (ch == '-') f = 1;
  for (; isdigit(ch); x = x * 10 + ch - '0', ch = getchar());
  return f ? -x : x;
}

template<class T>
bool read(T& x)
{
  x = 0;
  char ch = getchar();
  if (ch == EOF) return 0;
  for (; !isdigit(ch); )
  {
    ch = getchar();
    if (ch == EOF) return 0;
  }
  for (; isdigit(ch); x = x * 10 + ch - '0', ch = getchar());
  return 1;
}

template<class T>
void write(T x)
{
  static char s[22];
  static char* it = s + 20;
  static char* end = s + 20;

  if (x < 0)
  {
    putchar('-');
    x = -x;
  }

  do
  {
    *--it = x % 10 + '0';
    x /= 10;
  } while (x);
  /*
  if (!x)
    *-- it = '0';
    while (x)
    {
      *-- it = x%10+'0';
      x /= 10;
    }
    */
    for (; it < end; ++it)
      putchar(*it);
  }

  template<>
  void write(const char* s)
  {
    for (; *s; ++s) putchar(*s);
  }

  template<>
  void write(char c)
  {
    putchar(c);
  }

  template<class T, class V>
  void write(T x, V y)
  {
    write(x);
    write(y);
  }

  template<class T>
  void writeln(T x)
  {
    write(x);
    putchar('\n');
  }

  struct ender
  {
    ~ender()
    {
      if (os != obuf)
#ifdef FWRITE
        fwrite(obuf, 1, os - obuf, stdout);
#else
        write(STDOUT_FILENO, obuf, os - obuf);
#endif
    }
  }__ender;

}

template<class T>
void print(const T& a)
{
```

```cpp
  for (auto x : a) printf("%d ", x); puts("");
}

int64_t power(int64_t a, int64_t b, int64_t p)
{
  if (!b) return 1;
  int64_t t = power(a, b >> 1, p);
  t = t * t % p;
  if (b & 1) t = t * a % p;
  return t;
}

//mt19937_64 rd(chrono::steady_clock::now().time_since_epoch().count());
mt19937 rd(chrono::steady_clock::now().time_since_epoch().count());

using namespace io;

template<class T>
inline void freshmin(T& a, const T& b)
{
  if (a > b) a = b;
}

template<class T>
inline void freshmax(T& a, const T& b)
{
  if (a < b) a = b;
}

const ll B = 31;
//const ll MOD = 998244353;
const int INF = 1000000010;
//const ll INFll = 1000000000000000000LL;
const int MAXN = 400010;

int dx[] = { -1, 1, 0, 0, -1, -1, 1, 1 };
int dy[] = { 0, 0, -1, 1, -1, 1, -1, 1 };

ld det(ld x1, ld y1, ld x2, ld y2, ld x3, ld y3)
{
  return x1 * y2 - x2 * y1 + x2 * y3 - x3 * y2 + x3 * y1 - x1 * y3;
}

int n;
vector<int> v[MAXN];
ll a[MAXN];

void solve()
{
  n = inp();
  for (int i = 1; i <= n; ++i) a[i] = 1LL << i;
  ll sum = a[n];
  for (int i = 1; i < n / 2; ++i) sum += a[i];
  for (int i = n / 2; i < n; ++i) sum -= a[i];
  sum = abs(sum);
  writeln(sum);
}

int main()
{

  for (int T = inp(); T--; )
  solve();

  return 0;
}
```

Listing 3: Easy Sample 3 | Problem ID: 98902 | User ID: 721

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

#pragma GCC optimize(-O3)
#pragma GCC optimize(Ofast)
#pragma GCC optimize("unroll-loops")

#define fi first
#define se second
#define sqr(x) (x) * (x)
#define p_b push_back
#define m_p make_pair
#define pll pair<ll, ll>
#define all(v) v.begin(), v.end()
#define pw(x) (1ll << x)

using namespace std;
using namespace __gnu_pbds;
typedef long long ll;
typedef long double ld;
const ll MAXN = 1123456;
const ll N = 1e6;
const ll MOD = 1e9 + 7;

template <typename T> using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

template <typename T> void vout(T s){cout << s << endl; exit(0);}



int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);

    vector<pll> v;

    ll n, m;
    cin >> n >> m;

    ll l = 1, r = n;

    ll le = 1, ri = m;

    ll st = 0;

    while(l <= r){
        if(!st){
            v.p_b({l, le});
            le++;
        }else{
            v.p_b({r, ri});
            ri--;
            if(ri == 0){
                l++;
                le = 1;
                r--;
                ri = m;
            }
        }
        st = 1 - st;
        if(l == r && le > ri)break;
    }

    for(auto i : v)cout << i.fi << " " << i.se << "\n";

    return 0;
}
```

Listing 4: Easy Sample 4 | Problem ID: 98374 | User ID: 930

```
#include<bits/stdc++.h>
#define pb push_back
#define fast ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define int long long
#define pii pair<int,int>
#define all(x) x.begin(),x.end()
#define lb lower_bound
#define ld long double
using namespace std;
const int N = 3e5+1;
vector<pii> adj[N];
int dp[N];
int a[N];
int ans = 0;
void dfs (int src,int par){
  dp[src] = a[src];
  multiset<int,greater<int>> st;
  for(auto it:adj[src]){
    if(it.first!=par){
      dfs(it.first,src);
      dp[src] = max(dp[src],a[src] - it.second + dp[it.first]);
      st.insert(dp[it.first] - it.second);
    }
  }
  int nn = st.size();
  if(nn>=2){
    int fmx = *st.begin();
    st.erase(st.begin());
    int smx = *st.begin();
    ans = max(ans,fmx + smx + a[src]);
  }
  ans = max(ans,dp[src]);
}
signed main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  int n;
  cin>>n;
  for(int i = 1;i<=n;i++)
    cin>>a[i];
  for(int i = 1;i<=n-1;i++){
    int u,v,w;
    cin>>u>>v>>w;
    adj[u].pb({v,w});
    adj[v].pb({u,w});
  }
  dfs(1,-1);
  cout<<ans;
}
```

Listing 5: Easy Sample 5 | Problem ID: 97797 | User ID: 665

```
#include<bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

template<typename T>
using ordered_set = tree<T, null_type, less<T>, rb_tree_tag,
    tree_order_statistics_node_update>;
#define ll long long int
#define vi vector< int >
#define vl vector< ll >
#define pii pair< int, int >
#define pll pair< ll , ll >
#define pdd pair< double, double >
#define vii vector< pii >
#define vll vector< pll >
#define vd vector< double >
#define vb vector< bool >
#define el "\n"
#define ull unsigned long long int
#define ld long double
```

```cpp
#define fi first
#define se second
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define sqr(x) ((x) * (x))
#define FOR(i, a, b) for(ll i=a; i<=b; i++)
#define RFOR(i, b, a) for(ll i=b; i>=a; i--)
#define all(vec) (vec).begin(),(vec).end()
template <typename T> void maxi(T &x, T y) {if (y > x) x = y; }
template <typename T> void mini(T &x, T y) {if (y < x) x = y; }
#define error(args...) { string _s = #args; replace(_s.begin(), _s.end(), ',', ' '); stringstream _ss(_s); istream_iterator<string> _it(_ss); err(_it, args); }
void err(istream_iterator<string> it) {}
template<typename T, typename... Args>
void err(istream_iterator<string> it, T a, Args... args) {
  cerr << *it << " =: " << a << "\n";
    // cerr << "\n";
  err(++it, args...);
}
template<class T1, class T2, class T3>
struct triple {
    T1 a; T2 b; T3 c;
    triple() : a(T1()), b(T2()), c(T3()) {};
    triple(T1 _a, T2 _b, T3 _c) :a(_a), b(_b), c(_c) {}
};
template<class T1, class T2, class T3>
bool operator<(const triple<T1,T2,T3>&t1,const triple<T1,T2,T3>&t2) {
    if(t1.a!=t2.a)return t1.a<t2.a;
    else if(t1.b!=t2.b)return t1.b<t2.b;
    else return t1.c<t2.c;
}
template <typename T1, typename T2>
inline std::ostream& operator << (std::ostream& os, const std::pair<T1, T2>& p) {
    return os << "(" << p.first << ": " << p.second << ")";
}
template<typename T>
inline std::ostream &operator << (std::ostream & os,const std::vector<T>& v) {
    bool first = true;
    os << "[";
    for(unsigned int i = 0; i < v.size(); i++) {
        if(!first) os << ", ";
        os << v[i];
        first = false;
    }
    return os << "]";
}
template<typename T>
inline std::ostream &operator << (std::ostream & os,const std::set<T>& v) {
    bool first = true;
    os << "{";
    for (typename std::set<T>::const_iterator ii = v.begin(); ii != v.end();
        ++ii) {
        if(!first) os << ", ";
        os << *ii;
        first = false;
    }
    return os << "}";
}
template<typename T1, typename T2>
inline std::ostream &operator << (std::ostream & os,const std::map<T1, T2>& v
    ) {
    bool first = true;
    os << "{";
    for (typename std::map<T1, T2>::const_iterator ii = v.begin(); ii != v.
        end(); ++ii) {
        if(!first) os << ", ";
        os << *ii ;
        first = false;
    }
    return os << "}";
}
template<typename T>
inline std::ostream &operator << (std::ostream & os,const std::unordered_set<
    T>& v) {
    return os << std::set<T>(v.begin(), v.end());
}
template<typename T1, typename T2>
inline std::ostream &operator << (std::ostream & os,const std::unordered_map<
    T1, T2>& v) {
    return os << std::map<T1, T2>(v.begin(), v.end());
}

const ll MOD = 1e9+7;
const ll INF = 1e18;
const double EPS = 1e-6;
/**********************************************************************/

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; ++i) {
        ll x;
        cin >> x;
        a[i] = (int) __builtin_popcountll(x);
    }
    ll ans = 0, sum = 0;
    vector<int> cnt(2, 0);
    cnt[0]++;
    for (int e : a) sum += e, cnt[sum & 1]++;
    for (int e : cnt) ans += (ll) e * (e - 1) / 2LL;
    for (int i = 0; i < n; ++i) {
        sum = 0;
        int mx = -1;
        for (int j = i; j < i + 65 && j < n; ++j) {
            sum += a[j];
            maxi(mx, a[j]);
            if (sum % 2 == 0 && mx > sum - mx) ans--;
        }
    }
    cout << ans << el;

    return 0;
}
```

B. Hard Samples

Listing 6: Hard Sample 1 | Problem ID: 43 | User ID: 428

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    int a, b, c;
    cin >> a >> b >> c;
    cout << min(a + 2, min(b + 1, c)) * 3 - 3;
    return 0;
}
```

Listing 7: Hard Sample 2 | Problem ID: 15564 | User ID: 289

```cpp
#include <iostream>

#include<vector>
#define pb push_back


using namespace std;
int n,st = 0;

vector<vector<int>> v;

int main()
{
    cin >> n;
    v.resize(n+1);
    v[st].pb(0);
    st++;
    for (int  i = 0; i < n; i++)
    {
        int x;
        cin >> x;
        int l = 0;
        int r = st;
        while (l < r)
        {
            int c = (l+r) / 2;
            if (v[c].back() >= x) l = c+1; else r = c;
        }
        if (l ==st || v[l].back() > x)
        {
            v[st].pb(x);
            st++;
            continue;
        }
        v[l].pb(x);
    }
    for (int i = 0; i < st; i++)
    {
        for (int j = 0; j < v[i].size(); j++) if (v[i][j] != 0) cout << v[i][
            j] << ' ';
        cout << "\n";
    }
    return 0;
}
```

Listing 8: Hard Sample 3 | Problem ID: 31849 | User ID: 148

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <map>
#include <set>
#include <sstream>

using namespace std;

map<pair<int,int>,int> lol;

int gcd(int a,int b)
{
    if (!b)return a;
    return gcd(b,a%b);
}

pair<int,int> kek[200500];

int main()
{
    int n;
    cin >> n;
    for (int i=0;i<n;i++)
    {
        string s;
        cin >> s;
        for (auto &c:s)
        {
            if (c<'0'||c>'9')
            {
                c=' ';
            }
        }
        stringstream z(s);
        int a,b,c;
        z >> a >> b >> c;
        a+=b;
        b=gcd(a,c);
        a/=b;
        c/=b;
        ++lol[{a,c}];
        kek[i]={a,c};
    }
    for (int i=0;i<n;i++)
    {
        cout << lol[kek[i]] << ' ';
    }
    return 0;
}
```

Listing 9: Hard Sample 4 | Problem ID: 49881 | User ID: 71

```
#include<bits/stdc++.h>
#define pb push_back
using namespace std;
const int N=1e5+5;


int main()
{
  ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);

  cout<<2000<<endl;
  for(int i=1000;i>=1;i--)
    cout<<i<<" "<<1<<" "<<i<<" "<<2<<endl;

  for(int i=1;i<=1000;i++)
    cout<<i<<" "<<1<<" "<<i<<" "<<2<<endl;
  return 0;
}
```

Listing 10: Hard Sample 5 | Problem ID: 65320 | User ID: 21

```
#include <bits/stdc++.h>
using namespace std;

int input() {
  char c; cin >> c;
  return c == '1';
```

```cpp
}

voProblem id print_ans(vector<int> r, vector<int> c) { | User ID:
  int ans = 0;
  for (int i: r) ans += i;
  for (int i: c) ans += i;

  cout << ans << '\n';
  for (int i = 0; i < r.size(); i++) {
    if (r[i]) cout << "row " << i << '\n';
    if (c[i]) cout << "col " << i << '\n';
  }
}

int main() {
  ios::sync_with_stdio(false); cin.tie(0);

  int n; cin >> n;
  int a[n][n], c[n];
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      a[i][j] = input();
    }
  }
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      a[i][j] ^= input();
    }
  }
  for (int i = 0; i < n; i++) {
    c[i] = input();
  }
  vector<int> tog_row(n), tog_col(n);
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      if (c[i] + c[j] == 0) {
        if (a[i][j]) {
          cout << -1; exit(0);
        }
      } else if (c[i] + c[j] == 1) {
        if (a[i][j]) {
          if (c[j] == 1) {
            if (!tog_row[i]) {
              tog_row[i] = 1;
            }
          } else {
            if (!tog_col[j]) {
              tog_col[j] = 1;
            }
          }
        }
      }
    }
  }

  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      if (tog_row[i] || tog_col[j]) {
        if (tog_row[i] && c[j]) a[i][j] ^= 1;
        if (tog_col[j] && c[i]) a[i][j] ^= 1;

        if (a[i][j]) {
          cout << -1; exit(0);
        }
      }
    }
  }

  string sample_space;
  for (int i = 0; i < n; i++) {
    if (!tog_row[i] && c[i]) {
      for (int j = 0; j < n; j++) {
        if (!tog_col[j] && c[j]) {
          sample_space.push_back(a[i][j] + '0');
        } else {
          sample_space.push_back('X');
        }
      }
      break;
    }
  }
  if (sample_space.empty()) {
    print_ans(tog_row, tog_col);
  }

  for (int i = 0; i < n; i++) {
    if (!tog_row[i] && c[i]) {
      string space1, space2;
      for (int j = 0; j < n; j++) {
        if (tog_col[j] == 0 && c[j]) {
          space1.push_back(a[i][j] + '0');
          space2.push_back(1 - a[i][j] + '0');
        } else {
          space1.push_back('X');
          space2.push_back('X');
        }
      }

      if (space1 == sample_space) continue;
      if (space2 == sample_space) {
        tog_row[i] = 1;
        for (int j = 0; j < n; j++) {
          if (c[j]) a[i][j] ^= 1;
        }
      } else {
        cout << -1;
        exit(0);
      }
    }
  }
  for (int i = 0; i < n; i++) {
    if (sample_space[i] == '1') {
      tog_col[i] = 1;
    }
  }

  print_ans(tog_row, tog_col);

  return 0;
}
```