LOTFIZADEH OLYMPIAD

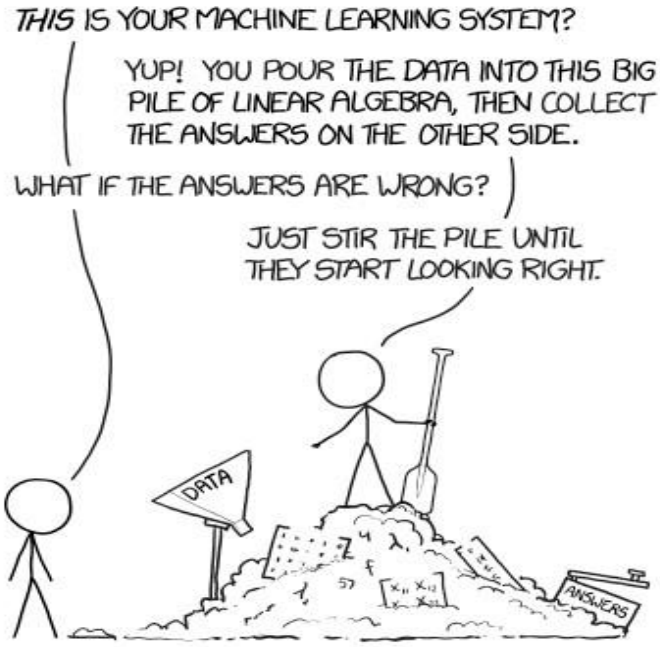
Main Round
23 Dec. 2021

Fuzzy Pairs

Time limit: 1 s

Memory limit: 512 MB

After learning about fuzzy logic, Barish decided to use it to train his brand-new machine learning model for predicting the results of IOI 2022. Although his previous model was achieving 99.1921% accuracy in the validation set from IOI 2019, Barish wants the new model to work perfectly. However, he also realized that applying fuzzy logic will not suffice: he also needs to design a better evaluation algorithm to improve the model. This is where he came up with the "fuzzy pairs" validation algorithm that allows Barish to train his model in a semi-supervised learning.



In short, he creates N disjoint pairs using all numbers from 1 to $2N$ (inclusive). Then, he gives the model an access to a function `distinct(S)` that takes an argument `S` which is a subset of the set $\{1, 2, 3, \ldots, 2N\}$ and reports number of *distinct pairs that have at least one element in that set*. For example, if $N = 2$ and the pairs are $(1, 4)$ and $(2, 3)$, the following results will be true: `distinct({1})=1`, `distinct({1,2})=2`, `distinct({1,2,3})=2`, `distinct({1,4})=1`. The final goal is to find out all pairs.

Obviously, fuzzy gradient descent is a perfect algorithm to train this model, but Barish ran out of free GPU usage in Google Colaboratory, so he needs your urgent help to manually simulate his model so he knows if the model achieves 100% accuracy.

Implementation details

This is an interactive task. You will be given the file "`train.cpp`" in which you should implement the function `train(N)`:

`vector<pair<int, int>> train(int N);` This function will be called once. It can call the function `distinct(S)` several times. At the end, it should return the correct pairs in any order. Please note that each pair should be a `std::pair<int, int>` object and they should be returned in a `std::vector` container.

`int distinct(vector<int> S);` This function will take the subset described in the problem statement and will return the number of distinct pairs that have at least one element inside the subset. If `S` violates the rule of being the subset of $\{1, 2, 3, \dots, 2N\}$, the function will return -1. If you ever get -1, you should immediately terminate the function `train(N)`, otherwise you may get the verdict "`Execution timed out`".

Note: *In case you don't use the provided file "`train.cpp`", do not forget to write function prototype "`int distinct(vector<int> S);`" in your code, otherwise you will get the verdict* "`Compilation failed`".

Local testing

You will be given the file "`grader.cpp`", which you can compile together with your program to test it. It will read N pairs in any order (first two numbers represent the first pair, the next two numbers represent the second pair, etc.) and then call the function `train(N)` and check your answer.

Constraints

- $N = 128$
- $q \leq 2^{15}$, here q is the number of calls you make to the function `distinct(S)`. Note that the score you will get from this problem will depend on this constraint. See the scoring section below.

Sample Interaction

`train(2)` function starts:

`distinct({1,2})` returns 2,

`distinct({1,4})` returns 1,

The function returns a vector container with pairs $\{1, 4\}$ and $\{2, 3\}$. Indeed, this is the correct answer. This example is from the problem statement.

Scoring

If you ask any wrong query or the number of queries exceeds the limit 2^{15} or just the set of pairs you found is wrong, you will get 0 points and the verdict "`Wrong Answer`". In other cases, your score will be determined as follows:

- If $2^{14} < q \leq 2^{15}$, you will get 5 points.
- If $2^{13} < q \leq 2^{14}$, you will get 13 points.
- If $2^{12} < q \leq 2^{13}$, you will get 23 points.
- If $2^{11} < q \leq 2^{12}$, you will get 37 points.
- If $2^{10} < q \leq 2^{11}$, you will get 63 points.
- If $960 < q \leq 2^{10}$, you will get 96 points.
- If $q \leq 960$, you will get 100 points.