Metody automatycznego tworzenia programów komputerowych

Adam Żychowski

DeepMind ✓
@DeepMind

Introducing #AlphaCode: a system that can compete at average human level in competitive coding competitions like @codeforces. An exciting leap in AI problem-solving capabilities, combining many advances in machine learning!

Read more: dpmd.ai/Alpha-Code 1/

5:11 PM · Feb 2, 2022 · Twitter Web App

Programisto! Szukaj nowego zajęcia

8 lutego 2022 0 Comments aktualności

Nowa sztuczna inteligencja DeepMind może pisać kod lepszy niż twój

4 lutego, 2022 Omkar Kale News, Tech Guides, Windows 0

ŚWIAT

Komputery piszą programy jak ludzie. Dzień sądu już niedługo?

Programiści zagrożeni bezrobociem? AI od Google potrafi programować

Czy nowe osiągnięcie w dziedzinie sztucznej inteligencji jest zagrożeniem dla wartości zawodu programisty? Automatyczne tworzenie programów stało się faktem dzięki AlphaCode – dziełu firmy należącej do Google.

COMPUTERWORLD KONFERENCJE TOP 200 OKIEM PREZESA TEMATY

Strona główna » Wiadomości » DeepMind: sztuczna inteligencja pisze kod lepiej...

DeepMind: sztuczna inteligencja pisze kod lepiej niż programiści

DeepMind's AlphaCode: Is AI ready to replace programmers?

DeepMind has claimed a breakthrough with AlphaCode, an AI which can beat humans in a programming contest. Should software developers fear for their jobs?

By Matthew Gooding

Program synthesis

Automatyczne generowanie kodu na podstawie:

- par: dane wejściowe + oczekiwany wynik
- opisu w języku naturalnym

Powiązane zagadnienia:

- *code summarization*
- naprawianie błędów w kodzie
- dodawanie komentarzy
- speech to code
- optymalizacja kodu
- zrównoleglanie programów

Test set

Input			Output	
inb	usep	dsep	expected	actual
1	0	100	0	0
1	11	110	1	0
0	100	50	1	1
1	-20	60	1	0
0	0	10	0	0

```
1  int buggy(int inb, int usep, int dsep) {
2    int bias;
3    if (inhb)
4      bias = dsep; //fix: bias = usep+100
5    else
6      bias = usep;
7    if (bias > dsep)
8      return 1;
9    else
10     return 0;
11 }
```

Figure 2.11: An example code repair synthesized by the SemFix [99] on a Tcas benchmark using the set of passing and failing test cases.

```
1  def computeDeriv(pol):
2    deriv = []
3    zero = 0
4    if (len(pol)==1):
5      return deriv
6    for e in range(0, len(pol)):
7      if (pol[e]==0):
8        zero += 1
9      else:
10       deriv.append(pol[e]*e)
11   return deriv
```

Generated Feedback (Repair)

The program requires **3** changes:

- In the return statement **return deriv** in **line 5**, replace **deriv** by **[0]**.
- In the comparison expression **(pol[e] == 0)** in **line 7**, change **(pol[e] == 0)** to **False**.
- In the expression **range(0, len(pol))** in **line 6**, change **0** to **1**.

Figure 2.12: An example repair generated by the AutoProf [130] system on a student submission to an introductory programming course on the edX platform.

Zastosowania

- autouzupełnianie, podpowiadanie
- nauka programowania
- upowszechnienie dostępu do narzędzi programistycznych, łatwiejsza automatyzacja
- obróbka danych (usuwanie obserwacji odstających, podział tekstów, normalizacja tabeli)
- grafika komputerowa (tworzenie powtarzalnych obiektów)
- i wiele więcej…

Toward Automatic Program Synthesis

Zohar Manna
Stanford University,* Stanford, California
and
Richard J. Waldinger
Stanford Research Institute,†
Menlo Park, California

* Computer Science Department. † Artificial Intelligence Group.

Communications of the ACM

March 1971
Volume 14
Number 3

Example 1. Construction of an iterative division program. We wish to construct an iterative program to compute the integer quotient and the remainder of two natural numbers x_1 and x_2, where $x_2 \neq 0$. The program should set the output variable z_1 to be the quotient of x_1 divided by x_2, and the output variable z_2 to be the corresponding remainder.

Thus $x = x_1, x_2$, and $z = z_1, z_2$. Since we are not interested in the program's performance for $x_2 = 0$, our input predicate is

$\varphi(x) : x_2 \neq 0.$

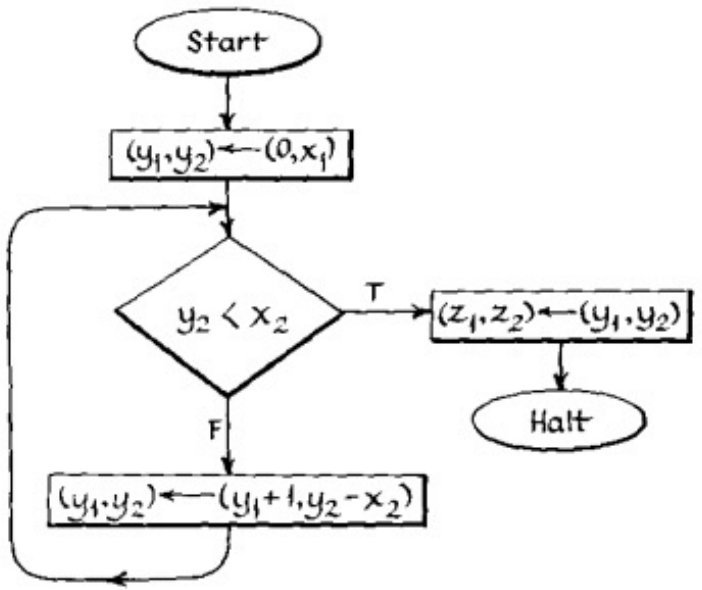
The output predicate is

$\psi(x, z) : (x_1 = z_1 \cdot x_2 + z_2) \wedge (z_2 < x_2).$

The theorem induced is then

$(\forall x_1)(\forall x_2)$
$\{x_2 \neq 0 \supset (\exists z_1)(\exists z_2)[(x_1 = z_1 \cdot x_2 + z_2) \wedge (z_2 < x_2)]\}.$

Fig. 1. A division program



A Representation for the Adaptive Generation of Simple Sequential Programs

Nichael Lynn Cramer
Texas Instruments Inc.
PO Box 226015,MS 238
Dallas, TX 75266

PROCEEDINGS OF
AN INTERNATIONAL CONFERENCE ON
GENETIC ALGORITHMS
AND THEIR APPLICATIONS

July 24-26, 1985

The basic language to be used is a variation of the algorithmic language PL having the following operators:

```
(:INC VAR) ;;add 1 to the variable VAR
(:ZERO VAR) ;;set the variable VAR to 0
(:LOOP VAR STAT) ;;perform the statement STAT VAR times
(:GOTO LAB) ;;jump to the statement with label LAB
```

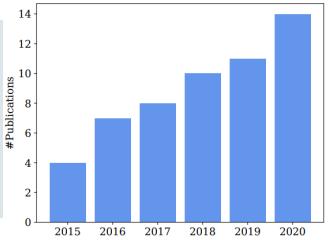
Główne kierunki rozwoju

systemy regułowe

- najstarsze i najbardziej pymitywne
- tylko najprostsze problemy, ściśle zdefiniowane

algorytmy ewolucyjne

- ciągle popularne
- główne różnice w kodowaniu problemu



sieci neuronowe

- najnowsze
- najbardziej zaawansowane
- metody przetwarzania tekstu: sieci rekurencyjne, transformery

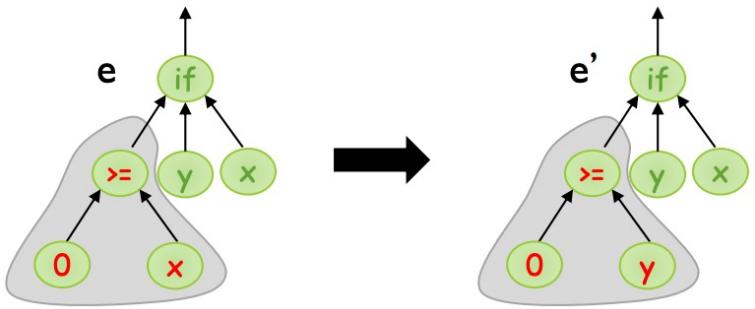
różne języki programowania, różne zbiory benchmarkowe, różny poziom skomplikowania zadań

click search button → click(findButton("search"))
left margin 2 inches → ActiveDocument.PageSetup.LeftMargin = InchesToPoints(2)

Little G. et al., *Translating keyword commands into executable code*, 2006.

```
1  harness void tripleSketch(int x) {
2      int h = ??; // hole for unknown constant
3      assert h * x == x + x + x;
4  }
```

Solar-Lezama A., *Program synthesis by sketching,* 2008.



Candidate Expressions	Counter-example from Verifier
x	$(x = 0, y = 1, f(x, y) = 1)$
y	$(x = 1, y = 1, f(x, y) = 1)$
1	$(x = 0, y = 0, f(x, y) = 0)$
$x + y$	$(x = 1, y = 1, f(x, y) = 1)$
$\texttt{if}(x \leq y, y, x)$	ϕ

Rajeev A. et al., *Syntax-guided synthesis*, 2013.

A Python function that calculates the average of a list of numbers.

```
def average(numbers): total = 0 for number in numbers: total += number return total/len(numbers)
```

A Python function that calculates the sum of a list of numbers.

```
def sum(numbers): total = 0 for number in numbers: total += number
```

Shin E. et al., *Program synthesis and semantic parsing with learned code idioms*, 2019.

Konkursy programistyczne (competetive programming)

- CodeForces, TopCoder, Code Jam
- Potyczki Algorytmiczne, Olimpiada Informatyczna

Stałe elementy:

- opis słowny
- definicja wejścia
- definicja wyjścia
- przykład
- wyjaśnienie przykładu

Backspace

You are given two strings s and t, both consisting of lowercase English letters. You are going to type the string s character by character, from the first character to the last one.

When typing a character, instead of pressing the button corresponding to it, you can press the "Backspace" button. It deletes the last character you have typed among those that aren't deleted yet (or does nothing if there are no characters in the current string). For example, if s is "abcbd" and you press Backspace instead of typing the first and the fourth characters, you will get the string "bd" (the first press of Backspace deletes no character, and the second press deletes the character 'c'). Another example, if s is "abcaa" and you press Backspace instead of the last two letters, then the resulting text is "a".

Your task is to determine whether you can obtain the string t, if you type the string s and press "Backspace" instead of typing several (maybe zero) characters of s.

Input

The first line contains a single integer q ($1 \le q \le 10^5$) the number of test cases.
The first line of each test case contains the string s ($1 \le |s| \le 10^5$). Each character of s is a lowercase English letter.
The second line of each test case contains the string t ($1 \le |t| \le 10^5$). Each character of t is a lowercase English letter.
It is guaranteed that the total number of characters in the strings over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "YES" if you can obtain the string t by typing the string s and replacing some characters with presses of "Backspace" button, or "NO" if you cannot.
You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Example Input

```
4
ababa
ba
ababa
bb
aaa
aaaa
aababa
ababa
```

Example Output

```
YES
NO
NO
YES
```

Explanation

In order to obtain "ba" from "ababa", you may press Backspace instead of typing the first and the fourth characters.

There's no way to obtain "bb" while typing "ababa".

There's no way to obtain "aaaa" while typing "aaa".

In order to obtain "ababa" while typing "aababa", you have to press Backspace instead of typing the first character, then type all the remaining characters.

Zadanie: PAN
Pandemia [B]

POTYCZKI ALGORYTMICZNE

Potyczki Algorytmiczne 2021, runda druga. Limity: 512 MB, 2 s. *07.12.2021*

W Bajtocji panuje pandemia! Rozprzestrzeniający się wirus SPLAY-CRT-2 (zwany potocznie Kruskalowirusem) zagraża zdrowiu obywateli. Na szczęście najtęższe bajtockie głowy szybko wynalazły skuteczną szczepionkę. Jednak nie jest to koniec problemów – teraz należy zaplanować szczepienia.

Bajtocja składa się z n miast o numerach od 1 do n. Miasta są rozlokowane wzdłuż jednej głównej drogi; tak więc dla każdego i ($1 \leq i \leq n-1$), miasta o numerach i oraz $i+1$ sąsiadują ze sobą.

Każde miasto początkowo albo jest całkowicie wolne od wirusa, albo wszyscy jego mieszkańcy są nim zarażeni. Każdego dnia bajtocka służba zdrowia może zaszczepić wszystkich mieszkańców jednego dowolnie wybranego miasta – pod warunkiem, że nie jest ono jeszcze opanowane przez zarazę. Niestety, każdej nocy, każde niewyszczepione miasto, które było do tej pory wolne od choroby, ale sąsiaduje z miastem opanowanym przez wirusa, staje się opanowane przez chorobę. Proces wyszczepiania Bajtocji rozpoczyna się nad ranem: najpierw służba zdrowia może wyszczepić jedno miasto, a dopiero potem wirus zaczyna się rozprzestrzeniać.

Twoim zadaniem jest tak rozplanować szczepienia, aby w momencie, gdy sytuacja w Bajtocji się ustabilizuje (tzn. każde miasto będzie albo zarażone wirusem, albo zaszczepione), zarażonych miast było możliwie jak najmniej.

Pomóż mieszkańcom Bajtocji i policz, ile miast będzie zarażonych przy optymalnej strategii szczepienia.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba całkowita t ($1 \leq t \leq 10^5$), oznaczająca liczbę przypadków testowych.

W kolejnych $2t$ wierszach znajdują się opisy kolejnych przypadków testowych. Każdy z nich składa się z dwóch wierszy. Pierwszy z nich zawiera jedną liczbę całkowitą n ($1 \leq n \leq 10^5$), oznaczającą liczbę miast w Bajtocji. Drugi z nich zawiera ciąg n znaków `0` i `1`. Jeśli i-ty z tych znaków to `1`, to i-te miasto jest początkowo zarażone wirusem, w przeciwnym wypadku jest ono wolne od choroby.

Suma wartości n we wszystkich przypadkach testowych nie przekroczy 10^6.

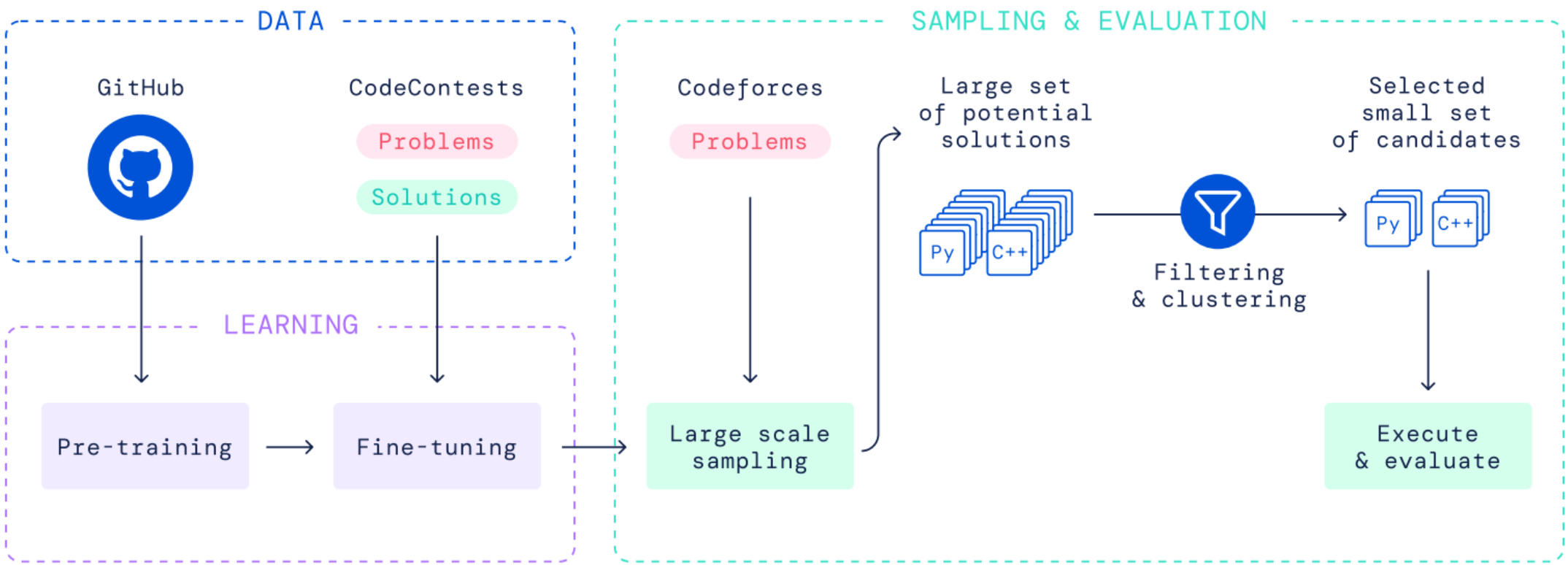
Wyjście

Na wyjściu powinno znaleźć się t wierszy. W i-tym z nich powinna znaleźć się jedna liczba całkowita, oznaczająca minimalną możliwą liczbę zarażonych miast w i-tym przypadku testowym.

Przykład

Dla danych wejściowych:	poprawnym wynikiem jest:
3 8 00110100 10 1001000010 4 0000	5 7 0

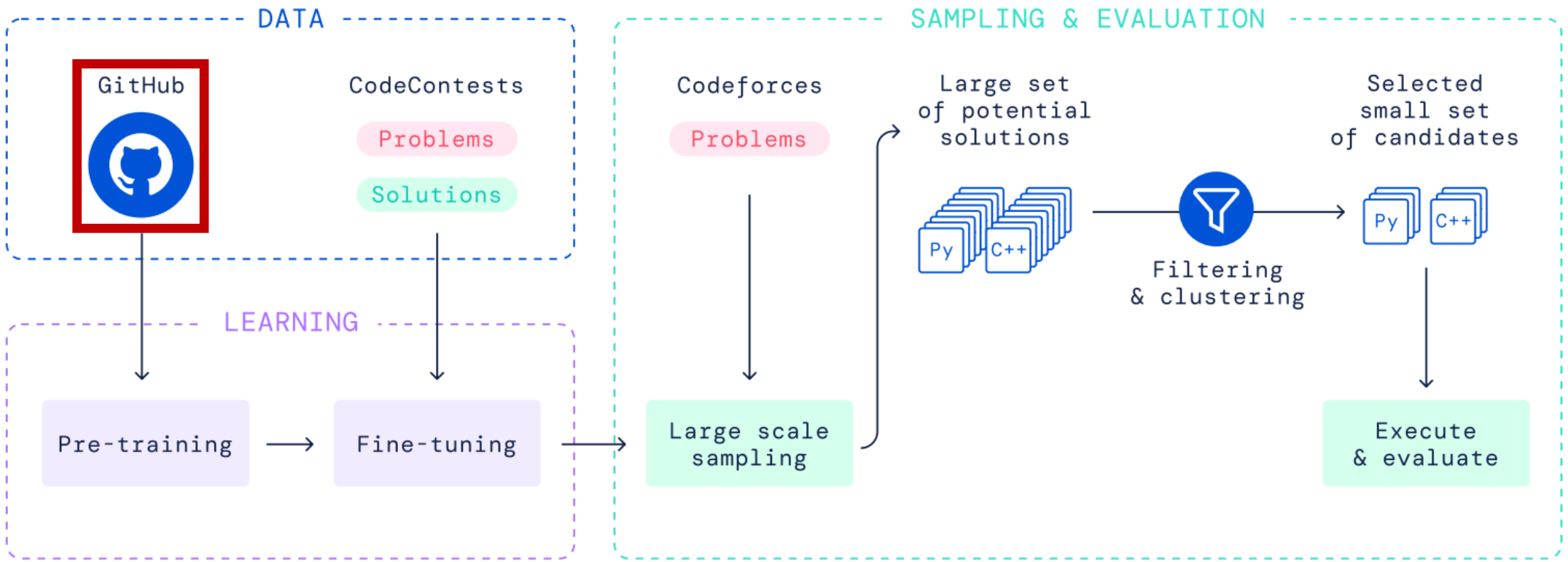
AlphaCode



Competition-Level Code Generation with AlphaCode, Yujia Li et al., 2022

https://arxiv.org/pdf/2203.07814.pdf

AlphaCode

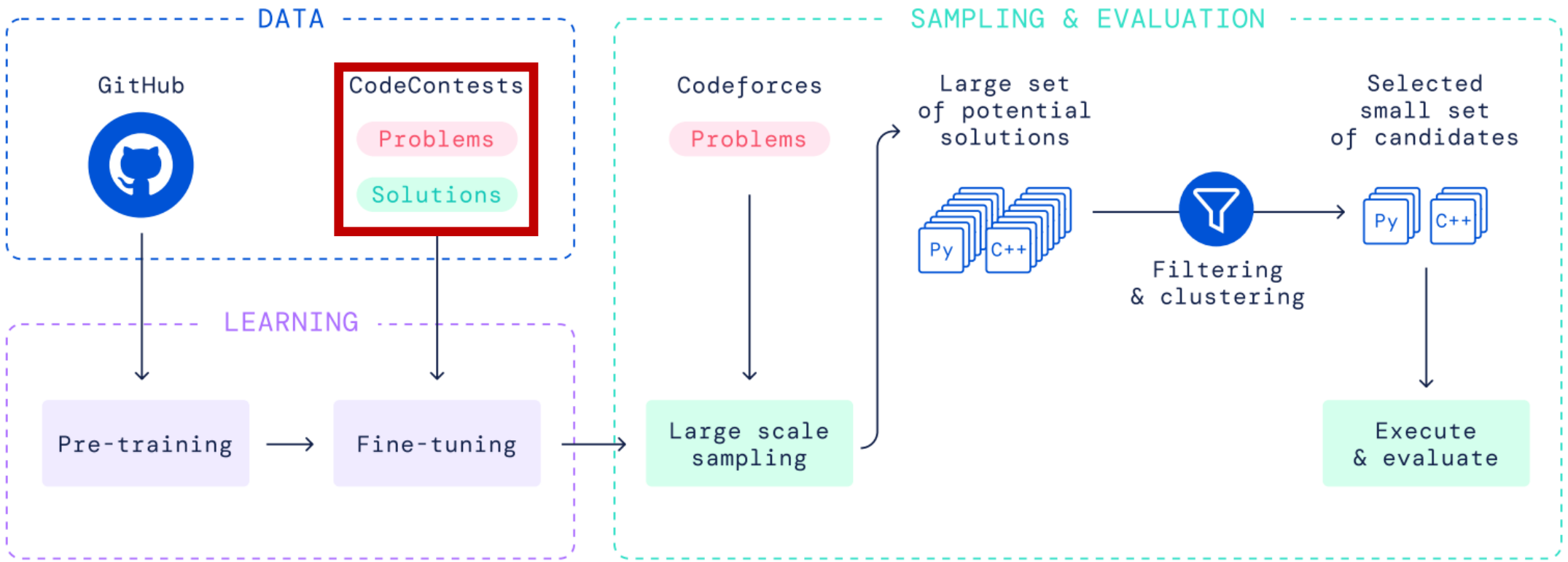

DATA
GitHub
CodeContests
Problems
Solutions
LEARNING
Pre-training
Fine-tuning
SAMPLING & EVALUATION
Codeforces
Problems
Large scale sampling
Large set of potential solutions
Py
C++
Filtering & clustering
Selected small set of candidates
Py
C++
Execute & evaluate

Dane treningowe

- 86 milionów plików z kodem (715GB danych) z publicznie dostępnych repozytoriów na GitHubie
- preprocessing:
  - usunięcie duplikatów
  - usunięcie białych znaków
  - rozwinięcie skrótów (np. *#define rep(i,n) for(int i=0;i<n;i++)*)
  - usunięcie plików większych niż 1MB lub z ponad 1000 liniami kodu

Language	Files (Millions)	Bytes (GB)	Bytes percentage
C++	21.50	290.5	40.62%
C#	6.73	38.4	5.37%
Go	2.19	19.8	2.77%
Java	19.35	113.8	15.91%
JavaScript	10.55	88.0	12.31%
Lua	0.57	2.9	0.41%
PHP	11.03	64.0	8.95%
Python 2	1.00	10.7	1.50%
Python 3	6.09	43.6	6.10%
Ruby	4.45	11.6	1.62%
Rust	0.32	2.8	0.39%
Scala	0.83	4.1	0.57%
TypeScript	1.69	24.9	3.48%
Total	86.31	715.1	100.00%

AlphaCode


DATA
GitHub
CodeContests
Problems
Solutions
LEARNING
Pre-training
Fine-tuning
SAMPLING & EVALUATION
Codeforces
Problems
Large scale sampling
Large set of potential solutions
Py
C++
Filtering & clustering
Selected small set of candidates
Py
C++
Execute & evaluate

Fine-tuning

- dane z konkursów programistycznych: opis problemu, metadane (trudność, rodzaj problemu), testy, rozwiązania użytkowników

Split	Problems	Tests per problem			Solutions per problem (% correct)		
		Example	Hidden	Generated	C++	Python	Java
Train	13328	2.0	14.8	79.1	493.4 (27%)	281.1 (47%)	147.9 (46%)
Valid	117	1.5	12.9	190.0	231.6 (47%)	137.2 (55%)	131.1 (54%)
Test	165	1.7	9.4	192.7	196.0 (45%)	97.3 (54%)	105.2 (51%)

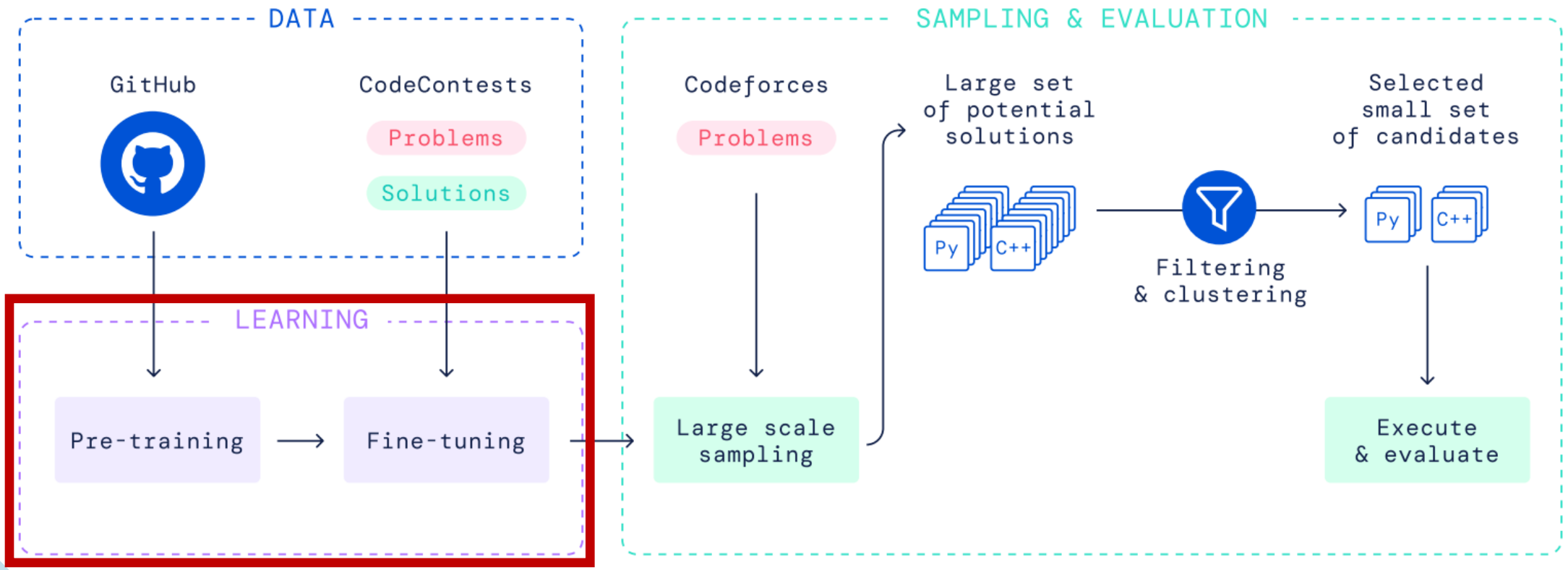
Encoder Input X:

```
// RATING: 1200
// TAGS: math
// LANGUAGE IS cpp
// CORRECT SOLUTION
// n towns are arranged in a circle sequentially. The towns are numbered from 1
// to n in clockwise order. In the i-th town, there lives a singer with a
// repertoire of a_i minutes for each i ∈ [1, n].
//
// Each singer visited all n towns in clockwise order, starting with the town he
// lives in, and gave exactly one concert in each town. In addition, in each
// town, the i-th singer got inspired and came up with a song that lasts a_i
// minutes. The song was added to his repertoire so that he could perform it in
// the rest of the cities.
//
// Hence, for the i-th singer, the concert in the i-th town will last a_i
// minutes, in the (i + 1)-th town the concert will last 2 · a_i minutes, ...,
// in the ((i + k) mod n + 1)-th town the duration of the concert will be (k +
// 2) · a_i, ..., in the town ((i + n - 2) mod n + 1) - n · a_i minutes.
//
// You are given an array of b integer numbers, where b_i is the total duration
// of concerts in the i-th town. Reconstruct any correct sequence of positive
// integers a or say that it is impossible.
//
// Input
//
// The first line contains one integer t (1 ≤ t ≤ 10^3) - the number of test
// cases. Then the test cases follow.
//
// Each test case consists of two lines. The first line contains a single
// integer n (1 ≤ n ≤ 4 · 10^4) - the number of cities. The second line contains
// n integers b_1, b_2, ..., b_n (1 ≤ b_i ≤ 10^{9}) - the total duration of
// concerts in i-th city.
//
// The sum of n over all test cases does not exceed 2 · 10^5.
//
// Output
//
// For each test case, print the answer as follows:
//
// If there is no suitable sequence a, print NO. Otherwise, on the first line
// print YES, on the next line print the sequence a_1, a_2, ..., a_n of n
// integers, where a_i (1 ≤ a_i ≤ 10^{9}) is the initial duration of repertoire
// of the i-th singer. If there are multiple answers, print any of them.
//
// Example
//
// Input
//
//
// 4
// 3
// 12 16 14
// 1
// 1
// 3
// 1 2 3
// 6
// 81 75 75 93 93 87
//
//
// Output
//
//
// YES
// 3 1 3
// YES
```

Decoder Output Y:

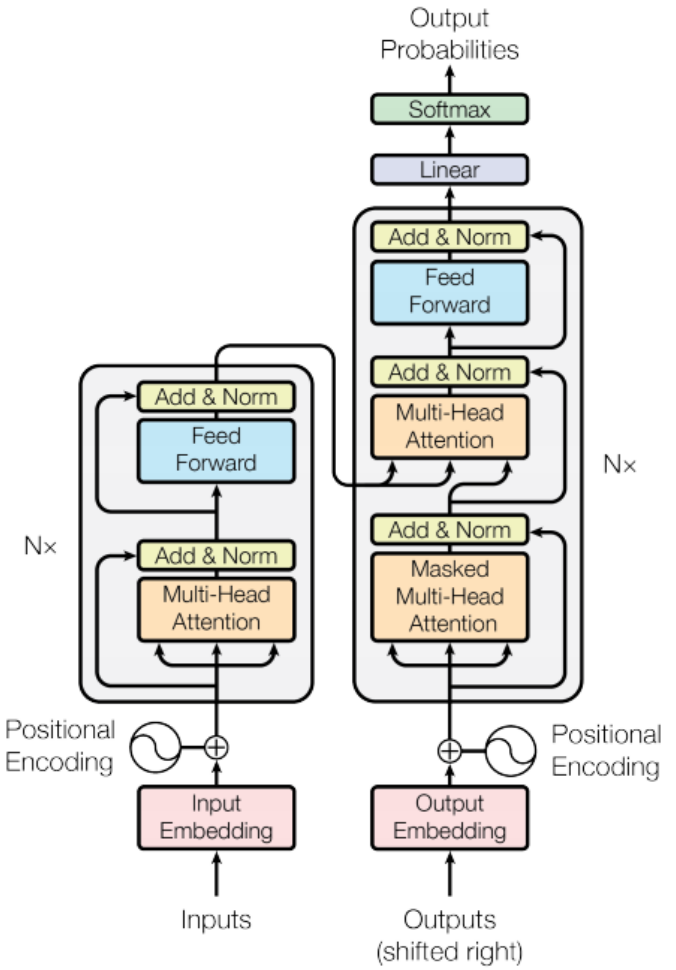
```
#include <bits/stdc++.h>
using namespace std;
void solve() {
  long long int n, i, j;
  cin >> n;
  vector<long long int> a(n);
  for (i = 0; i < n; i++) cin >> a[i];
  long long int sum = 0;
  for (i = 0; i < n; i++) sum += a[i];
  if (sum %
    cout << "NO" << '\n';
    return;
  }
  long long int k = sum / (n * (n + 1)
      / 2);
  vector<long long int> ans(n);
  for (i = 0; i < n; i++) {
    long long int t = 0;
    if (i == 0)
      t = a[n - 1];
    else
      t = a[i - 1];
    long long int p = k - (a[i] - t);
    if (p %
      cout << "NO" << '\n';
      return;
    }
    ans[i] = p / n;
    if (ans[i] <= 0) {
      cout << "NO" << '\n';
      return;
    }
  }
  cout << "YES" << '\n';
  for (i = 0; i < n; i++) cout << ans[i
      ] << " ";
  cout << '\n';
}
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  long long int t;
  cin >> t;
  while (t--) {
    solve();
  }
}
```

AlphaCode

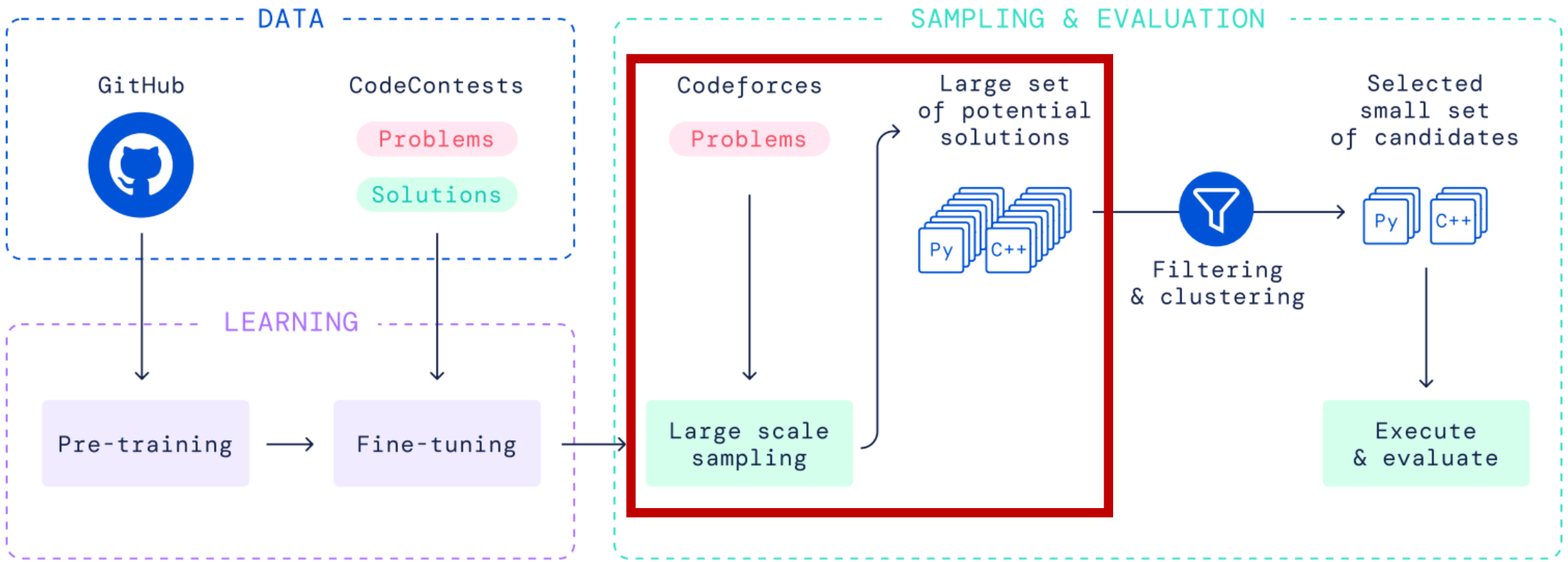

DATA
GitHub
CodeContests
Problems
Solutions
LEARNING
Pre-training
Fine-tuning
SAMPLING & EVALUATION
Codeforces
Problems
Large scale sampling
Large set of potential solutions
Py
C++
Filtering & clustering
Selected small set of candidates
Py
C++
Execute & evaluate

Trening

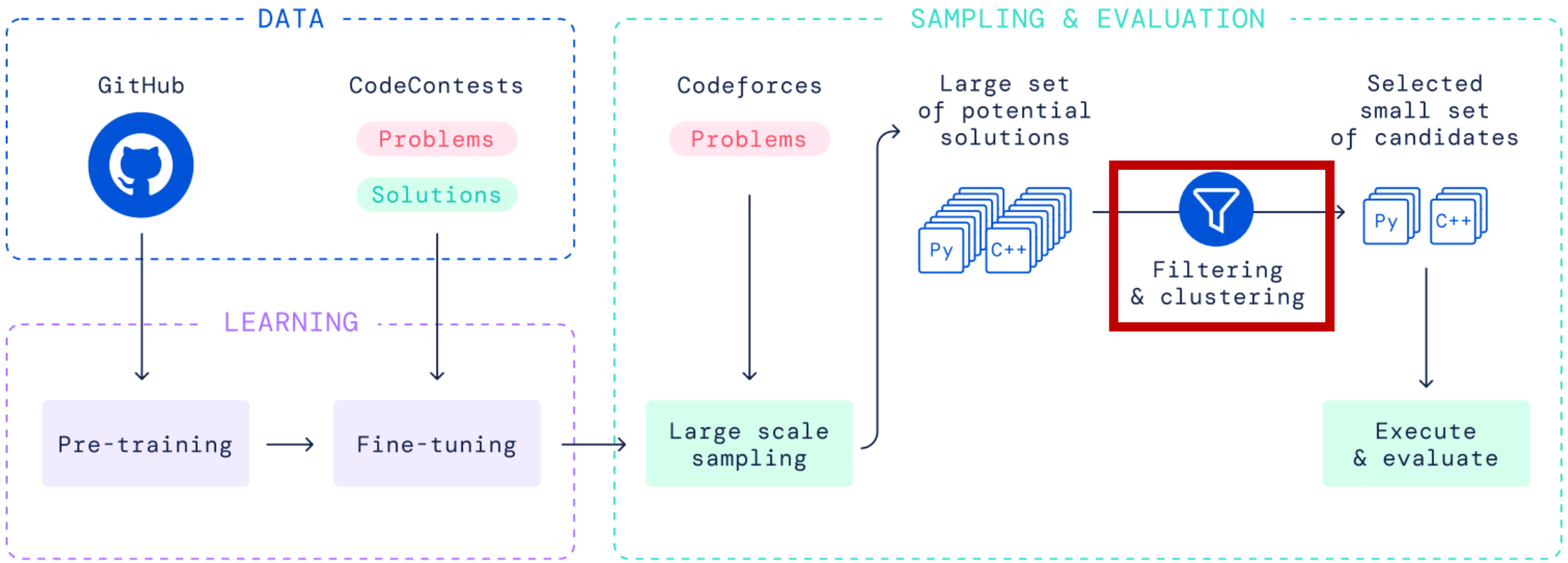
- potraktowanie problemu jako tłumaczenie języka naturalnego (angielskiego) na język programowania (kod) – *sequence-to-sequence*
- *encoder-decoder transformer* (Vaswani et al., *Attention is all you need*, 2017)
- 9 lub 41 miliardów parametrów treningowych
- 1536 tokenów tekstu w j. angielskim, 768 tokenów kodu
- SentencePiece do tokenizacji (https://github.com/google/sentencepiece)



AlphaCode


DATA
GitHub
CodeContests
Problems
Solutions
LEARNING
Pre-training
Fine-tuning
SAMPLING & EVALUATION
Codeforces
Problems
Large scale sampling
Large set of potential solutions
Py
C++
Filtering & clustering
Selected small set of candidates
Py
C++
Execute & evaluate

AlphaCode



- założenie: co najwyżej 10 zgłoszeń do każdego problemu

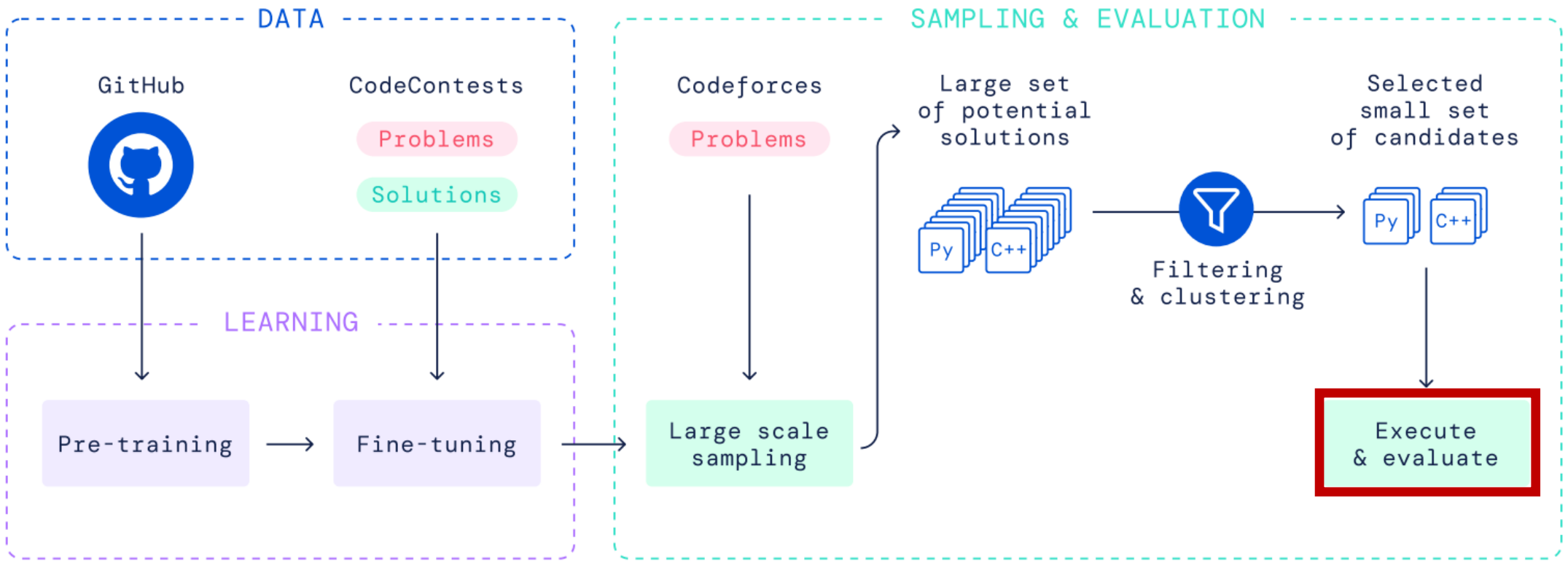
Filtrowanie

- odrzucenie kodów, które nie kompilują się, skutkują błędem lub nie zwracają poprawnego wyniku dla testu przykładowego → usunięcie około 99% wygenerowanych kodów
- dla około 10% problemów żaden z wygenerowanych kodów nie przechodził testu przykładowego
- dla większości programów nadal pozostawało setki lub tysiące kodów

Klaseryzacja

- nowy model nauczony generowania przypadków testowych (wejść) na podstawie tekstu zadania
- kody zwracające te same wyniki są grupowane w klastry
- nawet niepoprawnie wygenerowane dane są użyteczne
- jeden losowy reprezentant wybierany z klastrów (począwszy od klastra najbardziej licznego do najmniej licznego)

AlphaCode


DATA
GitHub
CodeContests
Problems
Solutions
LEARNING
Pre-training
Fine-tuning
SAMPLING & EVALUATION
Codeforces
Problems
Large scale sampling
Large set of potential solutions
Py
C++
Filtering & clustering
Selected small set of candidates
Py
C++
Execute & evaluate

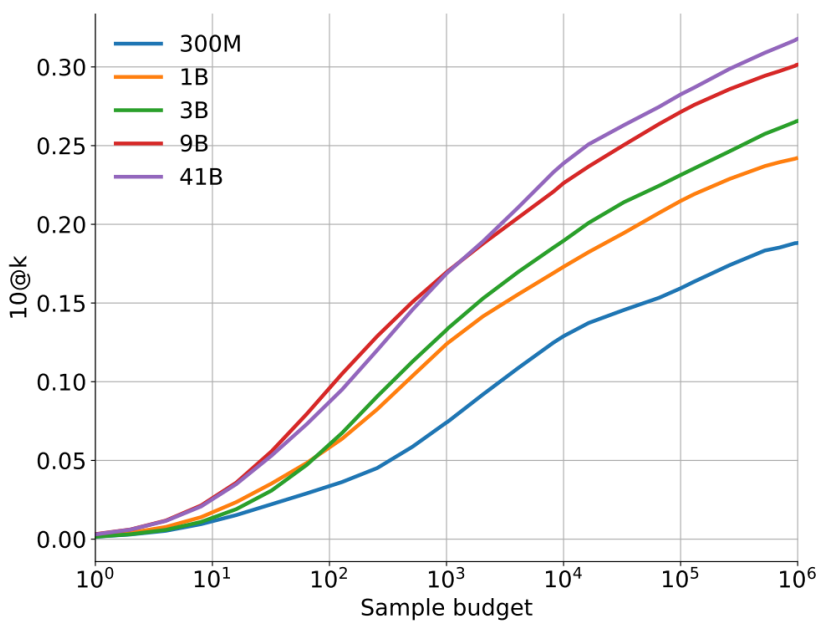
Wyniki

- 10 prawdziwych konkursów z platformy Codeforces ze średnią około 5000 uczestników
- średnio 2.4 zgłoszeń na poprawnie rozwiązane zadanie
- ranking: 1238 co stanowi 28% najlepszych użytkowników platformy konkursowej

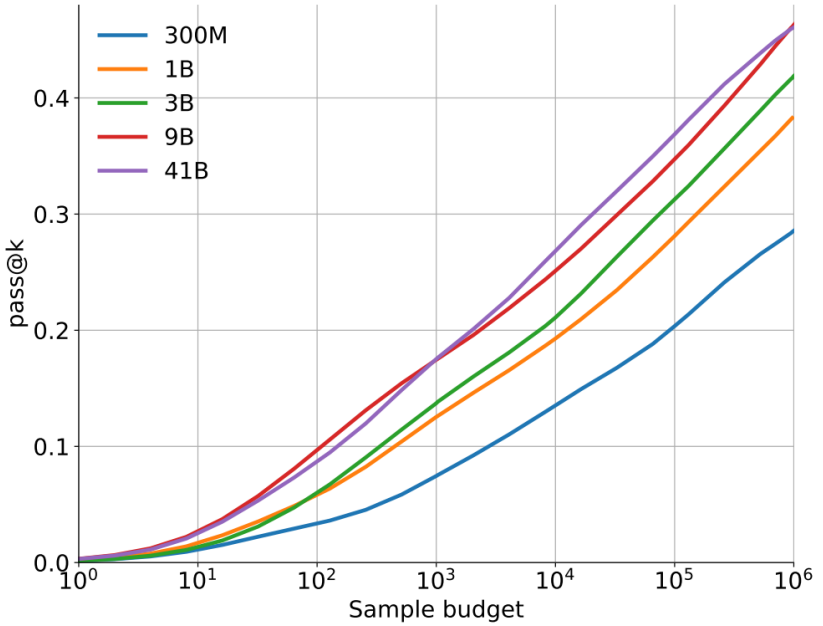
Contest ID	1591	1608	1613	1615	1617	1618	1619	1620	1622	1623	Average
Best	43.5%	43.6%	59.8%	60.5%	65.1%	32.2%	47.1%	54.0%	57.5%	20.6%	**48.4%**
Estimated	44.3%	46.3%	66.1%	62.4%	73.9%	52.2%	47.3%	63.3%	66.2%	20.9%	**54.3%**
Worst	74.5%	95.7%	75.0%	90.4%	82.3%	53.5%	88.1%	75.1%	81.6%	55.3%	**77.2%**

- bez limitu zgłoszeń: średnio 28.8 zgłoszeń na poprawne zadanie i 48.8% lepszych uczestników

Wyniki



(a) 10 attempts per problem



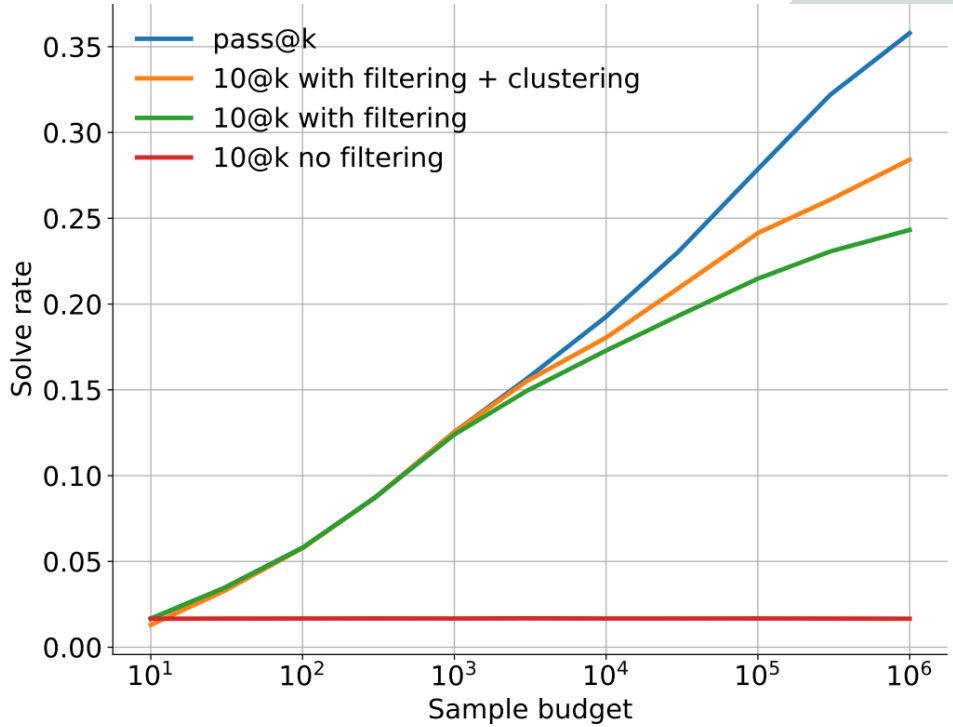
(b) Unlimited attempts per problem

Model	Greedy	Math	DP	Constructive Algorithms	Brute Force	Data Structures	Implem-entation	Graphs	Bitmasks	Sortings
300M	13.1%	19.3%	4.5%	7.5%	9.8%	8.8%	5.0%	0.2%	22.2%	16.9%
1B	19.7%	22.7%	4.5%	9.1%	12.0%	10.5%	14.1%	5.9%	26.8%	21.5%
3B	19.9%	22.7%	4.9%	11.2%	13.2%	11.9%	13.4%	8.8%	25.4%	23.8%
9B	23.7%	29.4%	7.1%	13.8%	19.5%	16.9%	16.4%	16.6%	27.4%	27.8%
41B	25.0%	28.2%	8.8%	14.9%	20.4%	15.7%	16.5%	13.6%	33.8%	25.5%

Wyniki

Pre-training dataset	Solve rate 10@1K	10@10K	10@100K
No pre-training	4.5%	7.0%	10.5%
GitHub (Python only)	5.8%	11.1%	15.5%
MassiveText	9.7%	16.1%	21.2%
GitHub (all languages)	12.4%	17.3%	21.5%

Wpływ zbioru treningowego

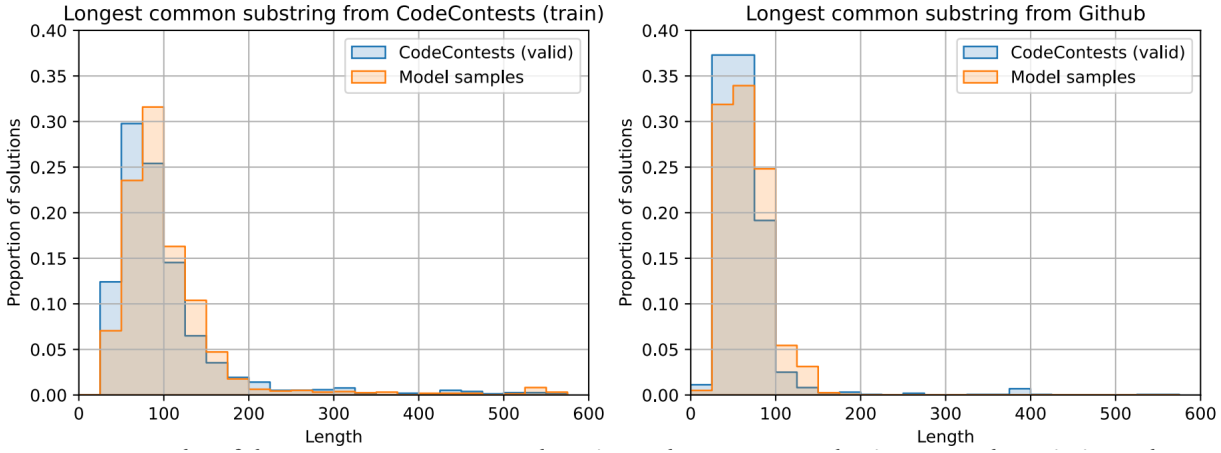


Wpływ filtrowania i klasteryzacji

Wyniki – porównanie z innymi metodami

	Filtered From (k)	Attempts (n)	Introductory n@k	Interview n@k	Competition n@k
GPT-Neo 2.7B	N/A	1	3.90%	0.57%	0.00%
GPT-Neo 2.7B	N/A	5	5.50%	0.80%	0.00%
Codex 12B	N/A	1	4.14%	0.14%	0.02%
Codex 12B	N/A	5	9.65%	0.51%	0.09%
Codex 12B	N/A	1000	25.02%	3.70%	3.23%
Codex 12B	1000	1	22.78%	2.64%	3.04%
Codex 12B	1000	5	24.52%	3.23%	3.08%
AlphaCode 1B	N/A	1000	17.67%	5.24%	7.06%
AlphaCode 1B	1000	5	14.36%	5.63%	4.58%
AlphaCode 1B	10000	5	18.18%	8.21%	6.65%
AlphaCode 1B	50000	5	20.36%	9.66%	7.75%

Wyniki czy AlphaCode kopiuje rozwiązania?



porównanie długości najdłuższych wspólnych podciągów rozwiązań ludzi i AlphaCode

Model-generated solution

```
n,m=map(int,input().split())
d={}
for i in range(m):
    u,v=map(int,input().split())
    if u not in d:
        d[u]=[v]
    else:
        d[u].append(v)
    if v not in d:
        d[v]=[u]
    else:
        d[v].append(u)
q=int(input())
ans=[]
for i in range(q):
    l=list(map(int,input().split()))
    if l[0]==1:
        if l[1] not in d:
            d[l[1]]=[l[2]]
        else:
            d[l[1]].append(l[2])
        if l[2] not in d:
            d[l[2]]=[l[1]]
        else:
            d[l[2]].append(l[1])
    elif l[0]==2:
        d[l[1]].remove(l[2])
        d[l[2]].remove(l[1])

    else:
        alive=[]
        for i in range(1,n+1):
            alive.append(i)
            if i in d:
                for j in d[i]:
                    if j in alive:
                        alive.remove(j)
        ans.append(len(alive))
for i in ans:
    print(i)
```

Source document of LCS

```
n, m, k = map(int, input().split())
d = {}
for _ in range(m):
    u, v = map(int, input().split())
    if u not in d:
        d[u] = [v]
    else:
        d[u].append(v)
    if v not in d:
        d[v] = [u]
    else:
        d[v].append(u)


# print(d)
stack = []
not_visited = set(range(1, n+1))

while not_visited:
    if not stack:
        stack.append((not_visited.pop(), 1, 0))
        path = []
        visited = {}
        c = 0
    p, l, parent = stack.pop()

    path.append(p)
    visited[p] = c
    c += 1
    if p in not_visited:
        not_visited.remove(p)
    f = False
    for x in d[p]:
        if x not in visited:
            stack.append((x, l+1, p))
        else:

            if c - visited[x] > k:
                path = path[visited[x]:]
                f = True
                break
    if f:
        break
print(len(path))
print(*path)
```

Human solution

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 100010;
long long int a[N], b[N], ans = 0;
int main() {
  int t;
  cin >> t;
  while (t--) {
    long long n;
    cin >> n;
    if (n <= 6) {
      cout << 15 << endl;
      continue;
    }
    if (n & 1) ++n;
    long long res = n % 6;
    switch (res) {
      case 0:
        cout << n / 6 * 15 << endl;
        break;
      case 2:
        cout << (n - 6) / 6 * 15 + 20 << endl;
        break;
      case 4:
        cout << (n - 6) / 6 * 15 + 25 << endl;
        break;
      default:
        break;
    }
  }
  return 0;
}
```

Model solution

```cpp
#include <bits/stdc++.h>
using namespace std;
const long long INF = 1e18;
void solve() {
  long long n;
  cin >> n;
  long long ans = INF;
  for (long long i = 0; i < 43; i++) {
    for (long long j = 0; j < 43; j++) {
      long long tmp = n;
      tmp -= (i * 6 + j * (long long)5);
      if (tmp <= 0) {
        ans = min(ans, i * (long long)15 + j * (long long)20);
        continue;
      }
      long long x = tmp / (long long)10;
      tmp -= x * 10;
      if (tmp > 0) x++;
      ans = min(ans, i * (long long)15 + j * (long long)20 + x * (long long)25);
    }
  }
  cout << ans << "\n";
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  long long t = 1;
  cin >> t;
  for (long long i = 1; i <= t; i++) {
    solve();
  }
  return 0;
}
```

Wyniki – up

Nim – Original

Nim is a game in which 2 players take turns removing objects from heaps of different sizes. On each turn, a player must remove at least one object, and may remove any number of objects provided they all come from the same heap. The player to remove the last object is the winner.

Formally there are n heaps, with integer values a_1, ..., a_n. A turn consists of reducing the value of some a_i to a value between zero and a_i - 1.

Given the list of heap sizes you need to figure out which player wins if both play optimally.

Input

The first line contains a single integer t (1 <= t <= 10 000) - the number of test cases.

The first line of each test case contains a single integer n (2 <= n <= 10^5).

The second line of each test case contains n integers a_1, a_2, ..., a_n (1 <= a_i <= 10^6).

Output

If the first player wins print "1", otherwise print "2"

Nim – Simplified

Given an array a, of length n, with values a_1, ..., a_n, compute the xor of all of the a_i.

If the xor is zero, output "2", else output "1".

Input

The first line contains a single integer t (1 $\leq$ t <= 10 000) - the number of test cases.

The first line of each test case contains a single integer n (2 <= n <= 10^5).

The second line of each test case contains n integers a_1, a_2, ..., a_n (1 <= a_i <= 10^6).

Output

If the xor of all the a_i is zero, print "2", otherwise print "1"

Example

Input

```
3
2
10 10
3
1 2 3
2
1 2
```

Output

```
2
2
1
```

Problem	Original	Simplified
1554A Cherry	2.98%	15.74%
1559A Mocha and Math	12.25%	55.53%
1569A Balanced Substring	10.61%	31.97%
No consecutive zeros	0.17%	1.25%
Nim	0.95%	85.38%

Backspace

You are given two strings s and t, both consisting of lowercase English letters. You are going to type the string s character by character, from the first character to the last one.

When typing a character, instead of pressing the button corresponding to it, you can press the "Backspace" button. It deletes the last character you have typed among those that aren't deleted yet (or does nothing if there are no characters in the current string). For example, if s is "abcbd" and you press Backspace instead of typing the first and the fourth characters, you will get the string "bd" (the first press of Backspace deletes no character, and the second press deletes the character 'c'). Another example, if s is "abcaa" and you press Backspace instead of the last two letters, then the resulting text is "a".

Your task is to determine whether you can obtain the string t, if you type the string s and press "Backspace" instead of typing several (maybe zero) characters of s.

Input

The first line contains a single integer q ($1 \le q \le 10^5$) the number of test cases.
The first line of each test case contains the string s ($1 \le |s| \le 10^5$). Each character of s is a lowercase English letter.
The second line of each test case contains the string t ($1 \le |t| \le 10^5$). Each character of t is a lowercase English letter.
It is guaranteed that the total number of characters in the strings over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "YES" if you can obtain the string t by typing the string s and replacing some characters with presses of "Backspace" button, or "NO" if you cannot.
You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Example Input

```
4
ababa
ba
ababa
bb
aaa
aaaa
aababa
ababa
```

Example Output

```
YES
NO
NO
YES
```

Explanation

In order to obtain "ba" from "ababa", you may press Backspace instead of typing the first and the fourth characters.

There's no way to obtain "bb" while typing "ababa".

There's no way to obtain "aaaa" while typing "aaa".

In order to obtain "ababa" while typing "aababa", you have to press Backspace instead of typing the first character, then type all the remaining characters.

```python
t=int(input())
for i in range(t):
    s=input()
    t=input()
    a=[]
    b=[]
    for j in s:
        a.append(j)
    for j in t:
        b.append(j)
    a.reverse()
    b.reverse()
    c=[]
    while len(b)!=0 and len(a)!=0:
        if a[0]==b[0]:
            c.append(b.pop(0))
            a.pop(0)
        elif a[0]!=b[0] and len(a)!=1:
            a.pop(0)
            a.pop(0)
        elif a[0]!=b[0] and len(a)==1:
            a.pop(0)
    if len(b)==0:
        print("YES")
    else:
        print("NO")
```

E. Gregor and the Two Painters

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Two painters, Amin and Benj, are repainting Gregor's living room ceiling! The ceiling can be modeled as an $n \times m$ grid.

For each i between 1 and n, inclusive, painter Amin applies a_i layers of paint to the entire i-th row. For each j between 1 and m, inclusive, painter Benj applies b_j layers of paint to the entire j-th column. Therefore, the cell (i, j) ends up with $a_i + b_j$ layers of paint.

Gregor considers the cell (i, j) to be *badly painted* if $a_i + b_j \leq x$. Define a *badly painted region* to be a **maximal** connected component of badly painted cells, i. e. a connected component of badly painted cells such that all adjacent to the component cells are not badly painted. Two cells are considered adjacent if they share a side.

Gregor is appalled by the state of the finished ceiling, and wants to know the number of badly painted regions.

Input

The first line contains three integers n, m and x ($1 \leq n, m \leq 2 \cdot 10^5$, $1 \leq x \leq 2 \cdot 10^5$) — the dimensions of Gregor's ceiling, and the maximum number of paint layers in a badly painted cell.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 2 \cdot 10^5$), the number of paint layers Amin applies to each row.

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \leq b_j \leq 2 \cdot 10^5$), the number of paint layers Benj applies to each column.

Output

Print a single integer, the number of badly painted regions.

Examples

input

```
3 4 11
9 8 5
10 6 7 2
```

output

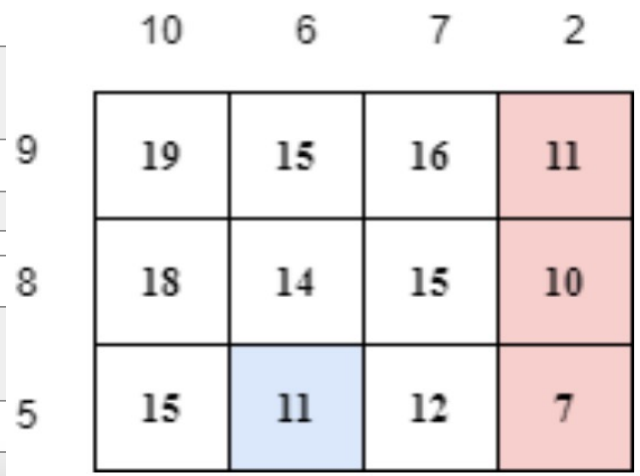
```
2
```

input

```
3 4 12
9 8 5
10 6 7 2
```

output

```
1
```



Solution

```
#include <bits/stdc++.h>
using namespace std;
int n, m, x;
int a[200005], b[200005];
int vis[200005];
int ans = 0;
void dfs(int i, int j) {
  if (i >= n || j >= m || i < 0 || j < 0 || vis[i * m + j] == 1 ||
      a[i] + b[j] > x)
    return;
  vis[i * m + j] = 1;
  dfs(i + 1, j);
  dfs(i, j + 1);
  dfs(i - 1, j);
  dfs(i, j - 1);
}
int main() {
  ios::sync_with_stdio(0);
  cin >> n >> m >> x;
  for (int i = 0; i < n; i++) cin >> a[i];
  for (int i = 0; i < m; i++) cin >> b[i];
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
      if (vis[i * m + j] == 0 && a[i] + b[j] <= x) {
        dfs(i, j);
        ans++;
      }
    }
  }
  cout << ans << endl;
  ;
  return 0;
}
```

https://alphacode.deepmind.com/#layer=30,problem=8

```
import sys,os,io
input = sys.stdin.readline # for strings
# input = io.BytesIO(os.read(0, os.fstat(0).st_size)).readline # for non-strings

PI = 3.141592653589793238460
INF =  float('inf')
MOD  = 1000000007
# MOD = 998244353


def bin32(num):
    return '{0:032b}'.format(num)

def add(x,y):
    return (x+y)%MOD

def sub(x,y):
    return (x-y+MOD)%MOD

def mul(x,y):
    return (x*y)%MOD

def gcd(x,y):
    if y == 0:
        return x
    return gcd(y,x%y)

def lcm(x,y):
    return (x*y)//gcd(x,y)

def power(x,y):
    res = 1
    x%=MOD
    while y!=0:
        if y&1 :
            res = mul(res,x)
        y>>=1
        x = mul(x,x)

    return res

def mod_inv(n):
    return power(n,MOD-2)
```

```
def prob(p,q):
    return mul(p,power(q,MOD-2))

def ii():
    return int(input())

def li():
    return [int(i) for i in input().split()]

def ls():
    return [i for i in input().split()]

for t in range(ii()):
    t+=1
    l,r = li()
    if r%2 == 1:
        if l > r//2:
            print(r%l)
        else:
            print(r%((r+1)//2))
    else:
        x = r//2
        x+=1
        if x>=l:
            print(r%x)
        else:
            print(r%l)
```

Solution

```
import sys

def input(): return sys.stdin.readline().strip()
def list2d(a, b, c): return [[c] * b for i in range(a)]
def list3d(a, b, c, d): return [[[d] * c for j in range(b)] for i in range(a)]
def list4d(a, b, c, d, e): return [[[[e] * d for j in range(c)] for j in range(b)]
for i in range(a)]
def ceil(x, y=1): return int(-(-x // y))
def INT(): return int(input())
def MAP(): return map(int, input().split())
def LIST(N=None): return list(MAP()) if N is None else [INT() for i in range(N)]
def Yes(): print('Yes')
def No(): print('No')
def YES(): print('YES')
def NO(): print('NO')
INF = 10 ** 19
MOD = 10 ** 9 + 7

for _ in range(INT()):
    N = INT()
    S = input()

    ans = [''] * N
    for i in range(N):
        if S[i] == '?':
            if i - 1 >= 0 and ans[i-1] == 'R':
                ans[i] = 'B'
            elif i - 1 >= 0 and ans[i-1] == 'B':
                ans[i] = 'R'
            else:
                ans[i] = 'B'
        else:
            ans[i] = S[i]
    print(''.join(ans))
```

AlphaCode – osąd bohatera

Minusy

- rozwiązanie à la brute force (*infinite monkey theorem*)
- bardzo duże zasoby obliczeniowe (petaflopy, TPU) i ludzkie (miesiące pracy zespołu 14 ludzi)
- kilka „sztuczek”

Plusy

- imponujący efekt
- kilka ciekawych pomysłów
- pokazanie, że „da się”
- połączenie pracy inżynieryjnej z naukową

Czy programiści staną się bezużyteczni?

Nie tak szybko, bo:

- rozwiązywany jest pewien szczególny rodzaj zadań (konkursy algorytmiczne)
- algorytmy to tylko część programowania – bazy danych, przepływ informacji, web serwisy, testy, itp.
- w praktyce prawie nigdy zadanie nie są określone tak ściśle
- programowanie to tylko część pracy programisty – zbieranie wymagań, tworzenie raportów, obróbka danych, itp.
- podczas tworzenia programów komputerowych nie wystarczy rozwiązać problemu – często ważne są dodatkowe kryteria: optymalizacja pamięci/czasu działania, bezpieczeństwo itp.

Dziękuję za uwagę