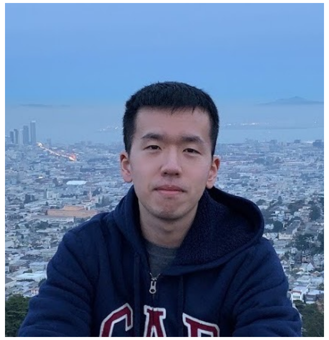
Planning with Large Language Models for Code Generation






Shun Zhang[1] Zhenfang Chen[1] Yikang Shen[1] Mingyu Ding[3] Joshua B. Tenenbaum[2] Chuang Gan[1,4]

[1]MIT-IBM Watson AI Lab, [2]MIT, [3]The University of Hong Kong, [4]UMass Amherst

Code Generation from Natural Language Descriptions

Problem Statement

Given is a string S. Replace every character in S with x and print the result.

Constraints

(1). S is a string consisting of lowercase English letters.

(2). The length of S is between 1 and 100 (inclusive).

Input

Input is given from Standard Input in the following format: S

Output

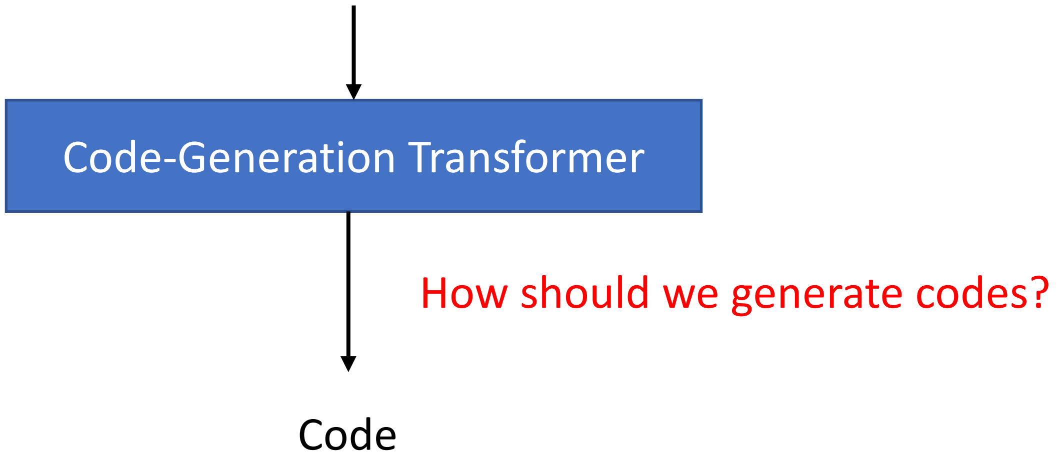
Replace every character in S with x and print the result.

Sample Test Input

$sardine$

Sample Test Output

$xxxxxxx$



Problem Statement

Given is a string S. Replace every character in S with x and print the result.

Constraints

(1). S is a string consisting of lowercase English letters.

(2). The length of S is between 1 and 100 (inclusive).

Input

Input is given from Standard Input in the following format: S

Output

Replace every character in S with x and print the result.

Sample Test Input

$sardine$

Sample Test Output

$xxxxxxx$

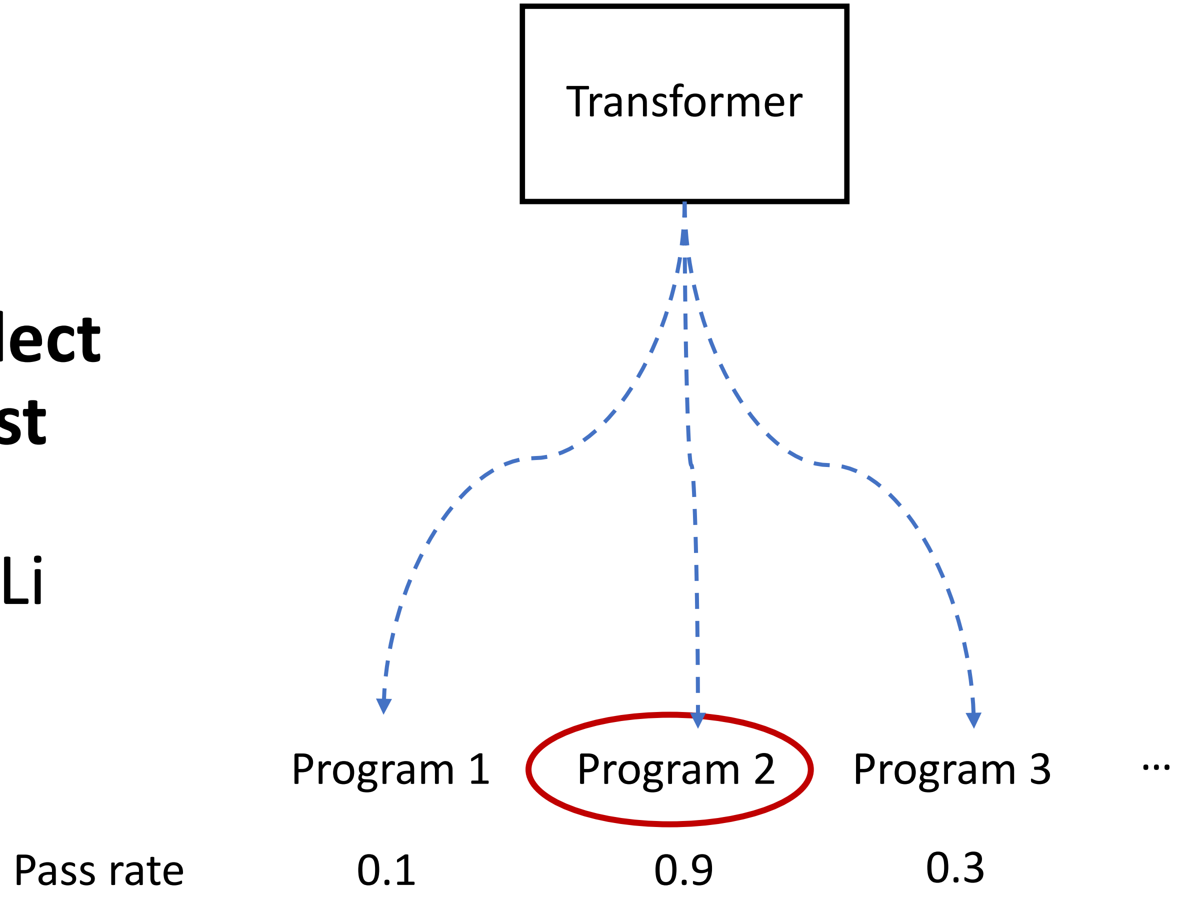
```
s=input()
s=list(s)
for i in range(len(s)):
    for j in range(len(s)):
        if s[i]=="x":
            s[i]=j
    print("".join(s))

```

Beam Search (Pass Rate: 0.00).

Hendrycks, Dan, et al. "Measuring coding challenge competence with APPS." *arXiv preprint arXiv:2105.09938* (2021).

A Well-Accepted Approach: Sampling + Filtering

- We sample a large number of programs using Transformer, run them on some test cases, and **select the program that passes the most number of test cases**.
- Used in DeepMind's AlphaCode (Li et al. 2022).



Li, Yujia, et al. "Competition-level code generation with alphacode." Science 378.6624 (2022): 1092-1097.

Problem Statement

Given is a string S. Replace every character in S with x and print the result.

Constraints

(1). S is a string consisting of lowercase English letters.

(2). The length of S is between 1 and 100 (inclusive).

Input

Input is given from Standard Input in the following format: S

Output

Replace every character in S with x and print the result.

Sample Test Input

$sardine$

Sample Test Output

$xxxxxxx$

```python
s=input()
s=list(s)
for i in range(len(s)):
    for j in range(len(s)):
        if s[i]=="x":
            s[i]=j
    print("".join(s))

```

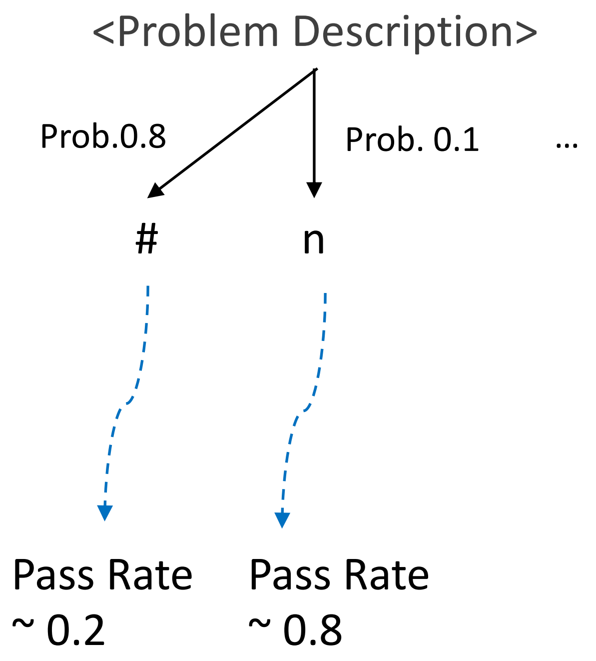
Beam Search (Pass Rate: 0.00).

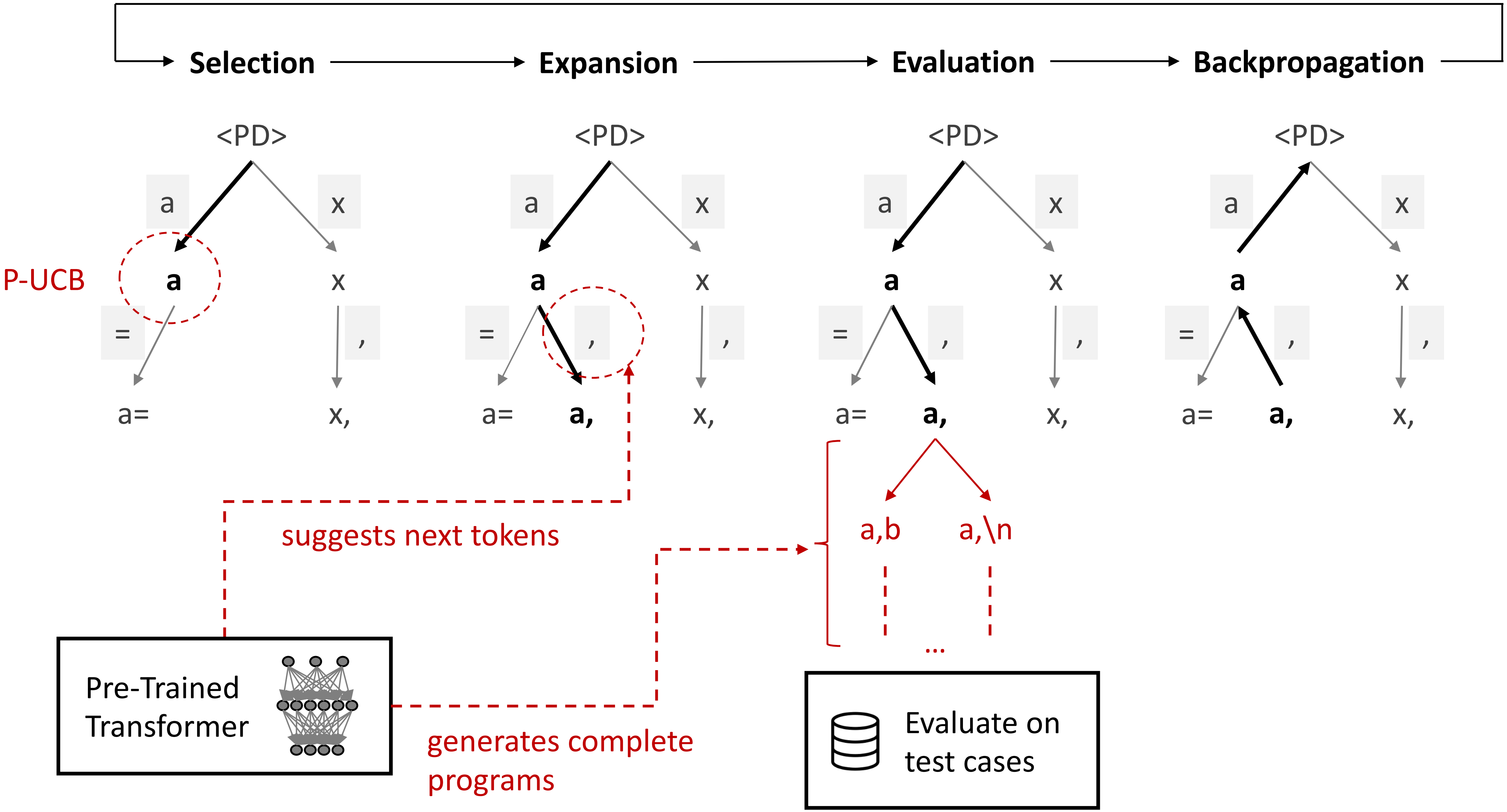
```python
s=input()
s=list(s)
for i in range(len(s)):
    if s[i]=="x":
        s[i]="x"
    else:
        continue
print("".join(s))
```

Sampling + Filtering (Pass Rate: 0.22).

What’s wrong with Sampling + Filtering?

- Sampling + Filtering is not efficient for code generation.
- Our contribution: using **a planning algorithm (a tree search algorithm)** that actively searches for programs with higher pass rates.



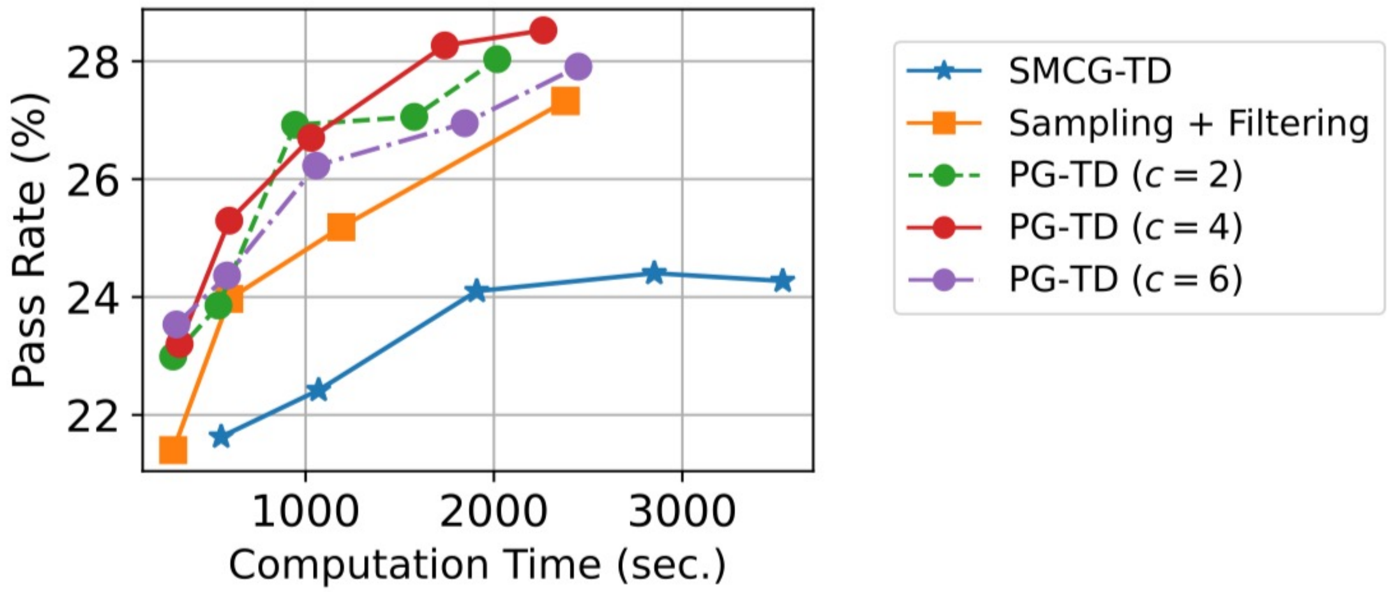

Selection
Expansion
Evaluation
Backpropagation
<PD>
a
x
P-UCB
=
,
a=
x,
a,
a,b
a,\n
...
suggests next tokens
Pre-Trained Transformer
generates complete programs
Evaluate on test cases

Empirical Results

		Pass Rate (%)				Strict Accuracy (%)			
		APPS Intro.	APPS Inter.	APPS comp.	CodeContests	APPS Intro.	APPS Inter.	APPS comp.	CodeContests
APPS GPT-2	Beam Search	11.95	9.55	5.04	5.10	5.50	2.10	1.00	0.00
	Sampling+Filtering	25.19	24.13	11.92	20.40	**13.80**	5.70	2.30	3.64
	SMCG-TD	24.10	21.98	10.37	17.47	11.70	5.50	2.10	4.24
	PG-TD ($c = 4$)	**26.70**	**24.92**	**12.89**	**24.05**	13.10	**6.10**	**3.10**	**4.85**
APPS GPT-Neo	Beam Search	14.32	9.80	6.39	5.73	6.70	2.00	2.10	0.00
	Sampling+Filtering	27.71	24.85	12.55	25.26	**15.50**	5.80	3.00	4.24
	SMCG-TD	25.09	20.34	9.16	15.44	13.80	5.10	1.80	3.03
	PG-TD ($c = 4$)	**29.27**	**25.69**	**13.55**	**26.07**	**15.50**	**6.43**	**3.50**	**4.85**

Empirical Results

Evaluated on the APPS introductory-level problems.



Summary

- **Effectiveness:** We contributed an algorithm that combines the advantages of **Transformer** and a **planning algorithm** and empirically showed that it generates problems with better quality.
- **Efficiency:** We empirically show that our algorithm is **more efficient** than sampling-based methods.
- **Generalization:** Our framework can be used to optimize different objectives *without fine-tuning* the Transformer model.
  - Generating shorter codes, codes with more comments.

Website: https://codeaimcts.github.io/