

プロンプトエンジニアリングによる 大規模言語モデルが生成する ソースコードの高精度化

増田研究室

小池温大

目次

- ▶ 研究背景
- ▶ 研究方法
- ▶ 研究結果
- ▶ 考察
- ▶ 終わりに

研究背景

▶ 大規模言語モデル

▶ Large Language Models(LLM)

- ▶ 膨大なテキストデータと高度なディープラーニング技術を用いて、構築された言語モデル
- ▶ システム開発の現場ではLLMを用いて、自動プログラミングを行うことで、開発効率の向上や時間の短縮を図っている。

研究背景

- ▶ 自動プログラミングにおけるLLMの課題

- ▶ LLMで作られたコードが常に有用とは限らない。
- ▶ 場合によっては、不正確なコードが生成される可能性がある。
- ▶ 原因のひとつには、プロンプトの入力内容



- ▶ プロンプトの入力内容によって、LLMはどのくらい正確なコードを生成できるか

研究方法

1.プログラミングの問題(AtCoderの問題)をChatGPT4.0でソースコードを生成する。

1 回目でプロンプトを作成する時は、問題文を元にプロンプトを再作成する。

2.ChatGPT4.0で作成されたソースコードが正確かどうかをオンラインジャッジシステムを用いて、判定する。

3.ソースコードが正確である場合は終了する。

研究方法

4.コードが不正確である場合は、正解コードと比較し、原因を調べる。

5.原因を元にプロンプトを再生成する。

6.2に戻り、ソースコードが正確か判定する。

このようにして、プロンプトの再作成とソースコードの生成を繰り返す

研究結果

- ▶ 今回は、44個のプログラミング問題に対するソースコードを生成した。
- ▶ 1回目で、44個のソースコードのうち、正確に生成することができたコードは8個である。
- ▶ 36個のソースコードのうち、何回か生成させたことにより、正確に生成することができたコードは5個であった。

プロンプトと間違って生成されたコード

問題

2のN乗を(2のM乗)-(2のK乗)で割った余りの1の位を求めてください。
T個のテストケースが与えられるので、それぞれについて答えを求めてください。

制約

$1 \leq T \leq 2 \times (10^5)$
 $1 \leq N \leq (10^{18})$
 $1 \leq K \leq M \leq (10^{18})$
N,M,Kは整数

入力

入力は以下の形式で標準入力から与えられる。ここで、i+1行目には、i番目のテストケースが入る。

T

N M K <-1番目のテストケース

N M K <-2番目のテストケース

⋮

N M K <-T番目のテストケース

出力

答えを出力せよ。i行目には、i番目のテストケースに対する答えを出力せよ

以下の問題をC言語で作成してください。

```
c コードをコピーする

#include <stdio.h>
#include <stdint.h>

// 指数計算を行いながらmodを取る関数
uint64_t powmod(uint64_t base, uint64_t exp, uint64_t mod) {
    uint64_t result = 1;
    while (exp > 0) {
        if (exp % 2 == 1) {
            result = (result * base) % mod;
        }
        base = (base * base) % mod;
        exp /= 2;
    }
    return result;
}

int main() {
    int T;
    scanf("%d", &T);

    while (T-- > 0) {
        uint64_t N, M, K;
        scanf("%llu %llu %llu", &N, &M, &K);

        // 2^M と 2^K の値を計算
        uint64_t mod = (1ULL << M) - (1ULL << K);

        // 2^N % mod を計算
        uint64_t remainder = powmod(2, N, mod);

        // 結果の一の位を出力
        printf("%llu\n", remainder % 10);
    }

    return 0;
}
```

符号なし整数でN,M,Kを定義しているため、大きな数を入力した時実行エラーになる。

2のM乗と2のK乗をそのまま計算している
→オーバーフローが発生してしまう。

その後のプロンプトと正解コード

問題

2のN乗を(2のM乗)-(2のK乗)で割った余りの1の位を求めてください。
T個のテストケースが与えられるので、それぞれについて答えを求めてください。

制約

$1 \leq T \leq 2 \times (10^5)$ 乗

$1 \leq N \leq (10^9)$ 乗

$1 \leq K \leq M \leq (10^9)$ 乗

N,M,Kは整数

入力

入力は以下の形式で標準入力から与えられる。ここで、i+1行目には、i番目のテストケースが入る。

T

N M K <-1番目のテストケース

N M K <-2番目のテストケース

⋮

⋮

⋮

N M K <-T番目のテストケース

(1番目のテストケースの答え)を出力

(2番目のテストケースの答え)を出力

⋮

⋮

⋮

(T番目のテストケースの答え)を出力

ヒント

2の累乗の1の位の規則性をうまく使えば、簡単に解くことができます。
ある数字をある数字で割った時に余りがどうなるかを考えて実装できると簡単に解くことができます。

このような感じである数字をある数字で割った時の結果を考えて実装すると、2のK乗や2のM乗を計算せずに、答えを求めることができます。

計算した値の1の位が何を表すために、2の累乗の1の位の周期を表す(6,2,4,8)を設定します。

まずnがkより小さい場合、答えは、n&3を計算した時の添え字の要素が答えとなります。

次にnがkより大きく、m-1 > kの時、

次にnがkより大きく、m-1 <= kの場合、((n-k)%(m-k)+k)&3を計算した時の添え字の要素が答えになります。

n=9,m=6,k=2の時、2の9乗%(2の6乗-2の2乗)=512%(64-4)=512%60=32となり、2となります。

(n-k)%(m-k)+kに当てはめて考えると、(9-2)%(6-2)+2&3=(7%4+2)&3=(3+2)&3=5&3=2となります。

nがkより大きく、m-1 <= kの場合またはmがkより小さい場合は、必ず0になります。

例として、n=5,m=6,k=5の時は、2の5乗%(2の6乗-2の5乗)=32%(64-32)=32%32=0となります。

n=7,m=6,k=5の時は、2の7乗%(2の6乗-2の5乗)=128%(64-32)=128%32=0となります。

以下の問題をC言語で作成してください。

C

コードをコピーする

```
#include <stdio.h>

int main() {
    int T;
    scanf("%d", &T);

    int last_digits[] = {6, 2, 4, 8}; // 2のべき乗の1の位の周期

    while (T--) {
        long long N, M, K;
        scanf("%lld %lld %lld", &N, &M, &K);

        if (N < K) {
            // NがK未満の場合、2^Nの1の位
            printf("%d\n", last_digits[N & 3]); // N & 3 は N % 4 と同等
        } else if (M - 1 > K) {
            // NがKより大きく、M-1がKより大きい場合
            printf("%d\n", last_digits[((N - K) % (M - K) + K) & 3]);
        } else {
            // M-1 <= Kの場合と、その他の条件に合わない場合、余りは必ず0
            printf("0\n");
        }
    }

    return 0;
}
```

考察

- ▶ 今回は、プログラミング問題における正解率が低くなったが、原因として以下が挙げられる。
 - ▶ 工夫せずに問題の通りにそのまま解いてしまっている場合
 - ▶ 問題の要件を満たすアルゴリズムがわからない場合
→適切なアルゴリズムを使えるように工夫が必要
 - ▶ 問題の意図にそぐわないコードが出力される場合
→AtCoderのサイトでは、分かりにくく記述している場合があるので、プロンプトの方で分かりやすく書く工夫が必要

終わりに

- ▶ プロンプトによって、問題の意図にそぐわないコードが生成されることもあったので、適切にプロンプトを入れる必要がある。
- ▶ アルゴリズムが複雑な問題について
アルゴリズムについて詳細に書くことで、正確なコードを生成することができた。
- ▶ しかし、問題に適切なアルゴリズムがわからない人にとっては不便
- ▶ アルゴリズムがわからなくても正確なコードを生成する方法を検討している。