

プログラミング課題とLLM

東京大学情報基盤センター 田中哲朗

背景(自己紹介)

- 田中哲朗 東京大学情報基盤センター情報メディア教育研究部門
- プログラミング教育
 - 1年生を主な対象にした「アルゴリズム入門」(関連授業も含めて1997年から),
 - 教養学部学際科学科3年生の「情報数理科学2」という名前の「データ構造」の授業(2013年から).
 - ACM-ICPC審判委員(2003 - 2016, うち2010, 2014は審判長)
- 研究
 - プログラミング言語処理系 (UtiLisp/C)
 - 漢字フォント(和田研フォント由来のフォントが今でも amazonで発行する明細書のPDFなどで使われている)
 - ゲーム情報学(どうぶつしょうぎ, GPS将棋など)
 - openai は 強化学習のフレームワーク openai gym で馴染み

LLMでプログラミング課題が解けるのか？

- LLMはプログラミングと相性が良い？
 - 自然言語よりも予測しやすい？
 - LLMを作る人たちも自分が使えると便利な対象として選んでいる？
 - プログラミングコンテストサイトの問題を解かせる課題もある程度対応
- LLMで多様性のある解答を作成可能
 - 同一性の判定のみでは判定できない (自分が手元でLLMで作成した解答といつも同じ解答が出るわけではない).
 - 「プログラムの説明を書かせる」ことで、プログラムを理解しているどうかをチェックさせるのもことも可能
- 田中が担当した実際の講義のプログラミング課題で例をみていく.
 - アルゴリズム入門
 - 情報数理学2

「アルゴリズム入門」の授業概要

- プログラミング経験がない受講者も対象にした情報科学入門講義
- 受講者は100名(全教員で1000名程度?)程度, 1年生が主, 一部2年生, クラス指定で基本は同じ内容
- 教科書は 森畑 明昌著「Pythonによるプログラミング入門東京大学教養学部テキスト」
- 2018年からPython言語を用いる
 - プログラミング環境は2018-2019は教育用計算機システムのiMac環境でのJupyter Notebookがメイン, 2020からはGoogle Colab
- 対面試験を実施
- 評価の一部に課題を用いる.
 - 絵やアニメーションを描くプログラム(2次元画像の配列あるいは, 画像フレームの配列を作成する)を除くと, 決まりきった課題.
 - 教科書の練習問題からの出題が多い.
 - 2018年から課題チェックプログラムを使って提出前にチェック可能(提出自体はITC-LMSの掲示板機能で)
 - 2020年から東京大学MIセンターの佐藤先生作成のPLAGS UTを使って提出

2022年度アルゴリズム入門で田中が用いた課題

- 11回のプログラミング課題を出題。
 - うち1回は、画像あるいは動画を作成するプログラムを ITC-LMSで提出させるもの。
 - 残り10回はPLAGS UTの練習問題を提出するもの(難易度が低いものを中心に全 12題)。
- 13題を ChatGPT (GPT-4使用)で解かせてみる。
 - GPT-4 は課金が必要(研究目的で登録)
 - GPT-4 はコーディング能力が低下したということだが、試したのは 2023年7月
 - プロンプトは「以下の問題にPython言語で解答しなさい。」
 - 問題文の数式は、 $\frac{a(r^n-1)}{r-1}$ のように、LaTeXの数式で与える。
 - 英語で入力の方が解答の精度が良いこともあるとのことだが、日本人学生が使用することを想定して、プロンプトも含めて日本語で問題文を与える。
 - 1回の試行のみ(統計的に意味のある結果を得ているわけでない)、毎回リセットせず続けて問題を入力(他の問題の入力が解答に影響することも)

結果

問題	結果	追加指示後結果
ex03_1	○	
ex03_2	○	
ex03_4	○	
ex04_2	△	○
ex04_3	○	
ex04_7	○	
ex04_8	△	○
		△
ex05_2	×	○
ex06_2	○	
ex07_2	○	
ex07_4	△	○
ex08_4	○	
画像あるいは動画	△	

出力例

正解例

入力

以下の問題にPython言語で解答しなさい。

Ex08_4(難易度☆☆☆) リストxと整数n ($1 \leq n \leq \text{len}(x)$ の長さ)が与えられたとき、x中の小さい方からn個 ($n \geq 1$)の要素を抽出する再帰関数ex08_4を定義せよ。この関数は、併合整列法を参考にして、リストを再帰的に分割し、それぞれの部分の小さい方からn個の要素を抽出し、それを併合してゆくことで最終的な結果を得るものとせよ。なおx中に同じ大きさの値が複数あったとしても、それらは別のものとしてよい。例えば{2,1,2,1,3}から3個抽出する場合、結果は{1,1,2}となる。なお、結果として出力される要素は値の小さい順に整列すること。(教科書練習問題5)

```
import ita
def ex08_4(x, n):
    ...
```

この問題のための関数は併合整列法を使用するので、まず、リストを2つのリストに分割します。それから、再帰的に分割したリストのそれぞれに対して同じ操作を行います。それぞれの再帰呼び出しから返された結果をマージしますが、各マージ操作では最小のn個の要素だけを維持します。以下に、そのためのPythonコードを示します。

```
def ex08_4(x, n):
    if len(x) <= 1:
        return x
    else:
        mid = len(x) // 2
        left = ex08_4(x[:mid], n)
        right = ex08_4(x[mid:], n)
        return merge_min_n(left, right, n)

def merge_min_n(left, right, n):
    result = []
    i = j = 0
    while i < len(left) and j < len(right) and len(result) < n:
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
    while i < len(left) and len(result) < n:
        result.append(left[i])
        i += 1
    while j < len(right) and len(result) < n:
        result.append(right[j])
        j += 1
    return result
```

このコードでは、ex08_4関数がリストxを分割してそれぞれの部分に対して再帰的に同じ関数を適用します。結果として得られる2つのリストはmerge_min_n関数に渡され、そこで最小のn個の要素を持つリストが作られます。マージ操作はリストの要素が尽きるか、n個の要素がマージされるまで続きます。

追加指示で正しい説明を誘導した問題(1)

入力

以下の問題にPython言語で解答しなさい。

Ex07_2 (難易度☆☆)

a^n は以下の漸化式を満たす。

$a^0 = 1$

$a^{2n} = (a^n)^2$

$a^{2n+1} = a \cdot (a^n)^2$

この漸化式に基づき、数 a と正整数 n をこの順に入力にとり、 a^n を求める再帰関数 `ex07_2` を定義せよ。(教科書練習問題7.4)

プログラムを作成するだけでなく、「計算量を見積もる過程と、その結果をO記法」で表して解答セルのコメント#で始まる行)で表してください。整数同士の乗算は結果のビット数がある程度よりも大きくなると時間がかかるので、計算量の見積もりの際には a を小数と仮定して解答してください。

出力例

以下に求めるプログラムを提供します。また、このアルゴリズムの計算量についても説明します。

```
def ex07_2(a, n):
    if n == 0:
        return 1
    elif n % 2 == 0:
        return ex07_2(a, n//2) ** 2
    else:
        return a * (ex07_2(a, n//2) ** 2)
```

上記の関数は再帰関数であり、入力値 n が0でない限り、自分自身を呼び出しています。その際に、 n を2で割った商を新たな n の値として再帰呼び出しを行っています。このため、この関数の計算量は、入力の大さき n を2で何回割ることができるか、つまり、 n の2進表現の桁数に比例します。そのため、この関数の時間計算量は $O(\log n)$ となります。

乗算については、一般に二つの整数の乗算はそれらのビット数に比例する時間がかかります。そのため、この関数では、 a のビット数と n のビット数の積に比例する時間がかかる乗算を行っています。したがって、この関数の時間計算量は $O((\log a) \cdot (\log n))$ となると言えます。

ただし、この見積もりは最悪ケースの時間計算量を見積もるものであり、平均ケースや最良ケースの時間計算量を見積もるものではありません。また、この見積もりは漸近的な計算量を見積もるものであり、具体的な入力値のサイズに対する正確な実行時間を予測するものではありません。

プログラムは正しいが、 a が実数ではなく整数だと思っている。また、 a が整数だとしても、正しくない。

追加指示で正しい説明を誘導した問題(2)

入力

この問題では、入力の a を小数(実数)と仮定しているのですが、結果も実数であり、 a のビット数が時間計算量に影響するのはおかしいのではないのでしょうか？

出力例

おっしゃる通り、入力の a が実数である場合には、ビット数に基づく考察が適切ではありません。再考します。

再帰的に計算を行うときに、 n が偶数か奇数かによって二つの計算パスがあります。

n が偶数の場合、 n を半分にして同じ計算を繰り返します。このたびの計算は a のべき乗の計算(一回の乗算)です。

n が奇数の場合、 n を半分にした後に同じ計算を行い、その後に a を掛けます。この計算パスでは乗算が二回行われます。

このアルゴリズムは、 n を半分にして計算を進めていくため、計算回数(再帰の深さ)はおおよそ $\log(n)$ となります。各ステップで行われる乗算の回数は一定(最大で2回)であるため、全体としてこのアルゴリズムの時間計算量は $O(\log(n))$ と評価されます。

正しい説明になっている。

不正解だが追加指示で正解した問題(1)

入力

以下の問題にPython言語で解答しなさい。

正の整数 n が与えられたとき n 行 n 列の市松模様を生成する関数 `ex05_2` を定義せよ。左上隅が白(つまり1)、それに隣接するマスが黒(つまり0)となるようにせよ。画像を表示するのではなく、画像に対応する2次元配列を作る関数とせよ。(教科書練習問題5.3)

```
import ita
def ex05_2(n):
    ...
```

出力例

この問題では、与えられた正の整数 n の値に基づいて市松模様を生成する次元配列を作成する必要があります。ここではPythonの内包表記を使用して、配列を生成します。

各セルは、その行番号と列番号が偶数または奇数であるかによってその値が決まります。具体的には、行番号と列番号の和が偶数の場合は値が、奇数の場合は値が0となります。

このロジックを用いて、問題の関数を次のように定義します。

```
def ex05_2(n):
    return [[(i + j) % 2 for j in range(n)] for i in range(n)]
```

この関数では、内側のリスト内包表記 `(i + j) % 2 for j in range(n)` が各行を生成し、外側のリスト内包表記 `[(i + j) % 2 for j in range(n)] for i in range(n)` が全体の配列を生成します。

ここで、 $(i + j) \% 2$ は行番号 i と列番号 j の和を2で割った余りを計算します。この値は、 i と j の両方が偶数または奇数である場合(つまり、 i と j の和が偶数である場合)は0、一方が偶数で一方が奇数である場合(つまり、 i と j の和が奇数である場合)は1になります。これにより、市松模様が生成されます。

説明の前半では行番号と列番号の和が偶数の場合は値が 1 になることを理解しているが、後半では 0 になると説明していて、プログラムもそのようになっている。

不正解だが追加指示で正解した問題(2)

追加指示

行番号と列番号の和が偶数の時に0でなく1になるようにプログラムを修正してください。

出力例

了解しました。行番号と列番号の和が偶数の時に1になるように、条件を逆にします。具体的には、 $(i + j) \% 2$ の結果を1から引くことで、0と1が反転します。修正した関数は以下のようになります。

```
def ex05_2(n):  
    return [[1 - (i + j) % 2 for j in range(n)] for i in range(n)]
```

この関数では、 $(i + j) \% 2$ が0なら $1 - 0$ となり1に、 $(i + j) \% 2$ が1なら $1 - 1$ となり0になります。したがって、行番号と列番号の和が偶数の場合は1、奇数の場合は0となります。

指示に従って、正しいプログラムを出力している。

なお、これに更に「2次元配列を作成するのに、リストの内包表記を用いずに関数 `ita.array.make2d`を用いたものに修正してください。」という指示を加えると、リストの内包表記を用いない想定解を出力する。

自由に画像を出力させる問題

出力例

入力

第1-6章まとめ課題 自分の好きな図あるいはアニメーションを描くプログラムを書いて、NotebookファイルをITC-LMSで第1-6章まとめ課題に投稿して下さい。第5章の練習問題にある図でも良いし、自分で工夫して考えた図でも良いです。なるべく `ita.plot.image_show` または `ita.plot.animation_show` で画像、動画を表示するプログラムを作成してください

(ヒントとして、簡単な静止画像を作成するプログラム、簡単なアニメーションを作成するプログラムを提示するが省略)

以下に、画像を生成するシンプルな関数を作成しましたこの関数は、ピクセルのRGB値を座標に基づいて計算します。これにより、縦方向にはグリーンからブルーへ、横方向にはレッドからマゼンタへ変化するグラデーション画像が作成されます。さらに、経過時間によってグラデーションの色調が変化するアニメーションを作ります。

まず、静止画像を生成する関数を作ります。

```
def show_color_picture(t):
    a = ita.array.make3d(50, 50, 3)
    for y in range(50):
        for x in range(50):
            a[y][x] = [x / 49, 1 - y / 49 + t / 10, t / 10]
    return a
ita.plot.image_show(show_color_picture(0)) # t = 0での画像を表示
```

この関数は、画像の各ピクセルについてGB値を計算します。R値はx座標に基づいているため、画像は左から右へとレッドからマゼンタに変化します。値はy座標に基づいており、画像は上から下へとグリーンからブルーに変化します。値は時間に基づいており、t値によって色調が変わります。

次に、この関数を使ってアニメーションを作ります。

```
def show_animation():
    images = ita.array.make1d(10)
    for t in range(10):
        images[t] = show_color_picture(t)
    return images
ita.plot.animation_show(show_animation())
```

この関数は、`show_color_picture(t)`を呼び出して時間ごとに画像を生成し、それらの画像を配列に格納します。その結果の配列を`plot.animation_show()`に渡すことでアニメーションを表示します。アニメーションでは、時間の経過とともに色調が変化します。

ご指定のitaライブラリの関数は私が模擬的に説明しており、実際には実行できないことをご承知ください。ですがpythonのPILやmatplotlibなどのライブラリを用いると、同様の結果を得ることが可能です。

サンプルの数値を変更しただけの、面白くないプログラム。

「アルゴリズム入門」の課題まとめ

- 13題のプログラミング課題のうち12題に最低限のプロンプトで正しいプログラムを出力する.
- 残りの1題も誤りの指摘に対応して正解.
- プログラムが正しいだけでなく、プログラムの説明も概ね適切だが、計算量のオーダーを解答させる問題は、誘導が必要な場合がある.
- 想定解と異なる出力が出たときも、追加の指示を出せば想定解に誘導可能.
- 自由に画像を出力させる問題では、ヒントとして出したサンプルとほぼ同じ内容だったので、評価は低い(授業では、「相互評価」として受講者同士の評価を点数の一部に用いた).
 - 「サンプル」が「制約」と解釈された?
- もともと、課題の評価で大きな差はつけないが、受講者の平均レベルは達成.
- プログラムだけを見て、LLMの出力をそのまま出したものかどうかを判定することは、困難

「情報数理学2」の授業の概要

- 受講者は教養学部学際科学科総合情報コース3年生が主，一部4年生や他コース，他学科，他学部
- 教科書は Pat Morin 著の「[Open Data Structures](#)」
 - Creative Commonsで公開されており，Java版，C++版，擬似コード(ほぼPython)版
 - 内容は一般的な「アルゴリズムとデータ構造」
 - ただし，Skiplistを扱う，binary treeでparentを持つ，red black treeを扱う(特にleft leaned red black tree)，singly linked listでFIFO queueを実現することなどが特徴的
 - Java版をもとにC++版が作成されたため，通常のC++プログラミング的には不自然な点もあり，授業ではC++版をベースに独自に修正したもの(本家でPRが採用されなかったもの)を使用
 - 英語を読んでもらうという意図もあったが，過去の受講者が [日本語訳](#)を作成，出版.

「情報数理学2」の課題概要

- 6回の課題を出題
 - 教科書の章末の練習問題の数題のうちから選択して解答
 - 2023年度は計18題から
 - プログラミング課題と説明課題 (図を用いて説明させるものを含む)
 - プログラミング課題はプログラミング言語を問わず (教科書の派生版が公開されているものに限る)
 - 2022年度は Rust で解答した受講者もいた。2023年度は半数近くが Python で解答
- チェック用プログラムを配布
 - tar.gz (C++版)および, Google Colabで実行可能なNotebook (C++版, Python版)を配布.
- 提出はLMSを用いてファイルで提出
 - Notebook, pdf, テキストファイル, zipファイルなど

ChatGPT (GPT-4使用)を使って解かせてみる

- プロンプト

以下のExercise ??にC++言語で解答しなさい。なお、ファイル入力をおこなうときには、入力は標準入力から与えられると仮定せよ。データ構造を定義する際は、要素の型をテンプレート引数で与えること。

なお、出題者からは以下のヒントが与えられている。
(問題ごとに受講者に与えるヒント、サンプルプログラムなど)

また、(データ構造名) は以下のように実装されている。

(教科書や配布プログラム中のデータ構造の C++クラス定義)

Exercise ??

(教科書の練習問題の英文。数式は LaTeX形式で)

- 図を参照しているときも、与えることができないので、「Figure ??」などをそのまま残す。

結果

Exercise	結果	追加指示後結果	備考
1.1	○		
2.2	○		
2.3	△	○	
2.7	○		
3.4	○		
3.7	○		
3.14	○		
4.5	×		図の入力が必要
4.11	○		
4.12	×	○	
5.5	×		説明問題
5.7	△		説明問題
6.4	×	×	
6.5	×	×	
6.14	○		
7.1	×		図を入力, 図を出力
7.3	×		図を入力, 図を出力
7.4	○		

正解例の入力

以下のExercise 3.4にC++言語で解答しなさい。なお、ファイル入力をおこなうときには、入力は標準入力から与えられると仮定せよ。データ構造を定義する際は、要素の型をテンプレート引数で与えること。

なお、出題者からは以下のヒントが与えられている。

Exercise 3.4ヒント

Exercise 3.4 は "should not use any secondary data structures, and should not create any new nodes." という条件を満たす必要がありますが,report3.tgzの中に含まれるSList.hの中からincludeされているex3_4.hは,

```
void reverse(){  
...  
} //Ex3_4
```

のように、この条件を満たしていませんがtempというsecondary data structureを使っている、new Nodeのところで、create new nodesを実行している、動くことは動きます。

また、SList.h は以下のように実装されている。

....

Exercise 3..4 Design and implement an SList method, reverse() that reverses the order of elements in an SList . This method should run in $O(\text{math}{n})$ time, should not use recursion, should not use any secondary data structures, and should not create any new nodes.

「...」部分は省略

正解例(出力)

以下に、SLListのメソッドであるreverse()を実装します。この実装は $O(n)$ 時間で動作し、再帰を使わず、補助的なデータ構造を使わず、新しいノードを作成せずにリストの要素の順序を逆にします。

```
template<class T>
void SLList<T>::reverse() {
    if (n == 0) return;
    Node *prev = NULL;
    Node *current = head;
    Node *next;
    while (current != NULL) {
        next = current->next; // temporarily store the next node
        current->next = prev; // reverse the link
        prev = current;      // move pointers one position ahead
        current = next;
    }
    // Swap head and tail
    Node* temp = head;
    head = tail;
    tail = temp;
}
```

このreverse()メソッドは、現在のノード(current)とその前のノード(prev)、そしてその次のノード(next)の三つのポインタを使用します。各反復では、現在のノードのnextポインタを前のノードを指すように更新します。そして、すべてのポインタを一つ前に進めます。この処理をリストの終端(NULL)まで続けると、リンクの向きが逆になります。

最後に、headとtailのポインタを交換します。これでリストの順序が反転します。

Singly Linked Listのreverseというのはよく知られた問題だが、FIFO queueとするために、tailも持つというこの教科書独自の仕様に従って、正しく書き換えている。

プログラミング問題の不正解例(入力)

以下の質問に回答してください。

また、教科書の以下のプログラムを参考にすること。

```
int size2() {
    Node *u = r, *prev = nil, *next;
    int n = 0;
    while (u != nil) {
        if (prev == u->parent) {
            n++;
            if (u->left != nil) next = u->left;
            else if (u->right != nil) next = u->right;
            else next = u->parent;
        } else if (prev == u->left) {
            if (u->right != nil) next = u->right;
            else next = u->parent;
        } else {
            next = u->parent;
        }
        prev = u;
        u = next;
    }
    return n;
}
```

Exercise 6..4 Implement a non-recursive method, $\text{size2}(u)$, that computes the size of the subtree rooted at node u .

入力として、rootからのnon recursive traversalのサンプルのみを与えた。

プログラミング問題の不正解例(出力)

上記の size2()メソッドは、二分木のサイズ(ノードの数)を非再帰的に計算します。これはルートノードから開始し、子ノード(左または右)が存在する場合はその方向に進み、子ノードがない場合は親ノードに戻ります。そして各ノードを一度だけ訪問し、訪問する度にカウンタ nを増加させます。

この関数を用いて特定のノード uから始まる部分木のサイズを計算する size2(u)メソッドを実装します。この方法では、開始ノードを uに変更し、終了条件を u != nilからu != u->parentに変更します。これにより、関数は uの部分木を訪問し、そのサイズを計算します。

以下に該当のコードを示します。

```
int size2(Node* u) {
    Node *prev = nil, *next;
    int n = 0;
    while (u != nil && (prev == nil || u != prev->parent)) {
        if (prev == u->parent) {
            n++;
            if (u->left != nil) next = u->left;
            else if (u->right != nil) next = u->right;
            else next = u->parent;
        } else if (prev == u->left) {
            if (u->right != nil) next = u->right;
            else next = u->parent;
        } else {
            next = u->parent;
        }
        prev = u;
        u = next;
    }
    return n;
}
```

ここで、nilは通常 NULLまたは nullptrを指し、Nodeは適切な型と仮定します。また、このコードは uが有効なノードポインタであると仮定します。そのため、無効な値を渡さないように注意が必要です。

受講者にもよくあることだが、whileの終了条件を間違えている。このあと、どう指示を与えても正解できなかった。ただし、最初rootからのnon recursive traversalのサンプルを与えないと、△の解答を出すことがある。

図を入出力とする不正解例(入力)

Exercise 7.1 Illustrate the addition of 4.5 (with priority 7) and then 7.5 (with priority 20) on the Treap in Figure 7.5.

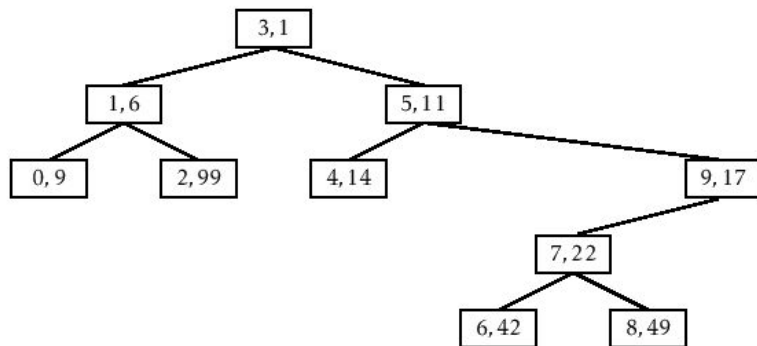


Figure 7.5: An example of a Treap containing the integers $0, \dots, 9$.

Each node, u , is illustrated as a box containing $u.x, u.p$.

図は与えない

図を入出力とする不正解例(出力)

Figure 7.5に対応するTreap(優先度付き二分探索木)を描画できないので、私はテキストを用いて操作の進行を説明します。

最初に4.5(優先度7)を追加します。これは現在のTreapにおいて、値が4と5の間に位置します。4.5を追加した後、このノードは親(値が4、優先度5)よりも高い優先度を持つため、右回転を行います。これにより、4.5(優先度7)が親ノードとなり、元の親であった4(優先度5)がその左の子ノードとなります。

次に7.5(優先度20)を追加します。この値はTreapの最大値よりも大きいため、最右部に位置するノードになります。しかし、7.5(優先度20)は全てのノードよりも高い優先度を持つため、ルートノードになるまで左回転を続けます。

以上の操作により、4.5と7.5がTreapに追加され、優先度の順序が保たれます。ノードの追加と回転により、Treapは二分探索木の性質(左の子は親より小さく、右の子は親より大きい)とヒープの性質(親は子より優先度が高い)を維持します。

なお、具体的なTreapの状態は問題の内容やFigure 7.5の状態に依存しますので、正確な状態を理解するにはそれらの情報が必要となります。

図 7.5を適当に仮定して、解答を生成している。

説明問題の不正解例(入力)

以下の質問に日本語で回答してください。なお、出題者からは以下のヒントが与えられている。

Exercise 5.5 ヒント

「giving an example of a sequence of $O(n)$ `add(x)`, `remove(x)`, and `find(x)` operations」にあるような `add`, `remove`, `find` の列のパターンを与えた上で、「take on the order of n^2 time to execute.」であることを説明する。その際に、教科書中のプログラム `add` では、そのような問題が起きないことも確認すること。

また、教科書のプログラム `add`, `find` は以下のコードである。

...

Exercise 5..5 Consider the following simplified version of the code for adding an element x to a `LinearHashTable`, which simply stores x in the first `null` array entry it finds. Explain why this could be very slow by giving an example of a sequence of $O(n)$ `add(x)`, `remove(x)`, and `find(x)` operations that would take on the order of n^2 time to execute.

```
bool addSlow(T x) {
    if (2*(q+1) > t.length) resize(); // max 50% occupancy
    int i = hash(x);
    while (t[i] != null) {
        if (t[i] != del && x.equals(t[i])) return false;
        i = (i == t.length-1) ? 0 : i + 1; // increment i
    }
    t[i] = x;
    n++; q++;
    return true;
}
```


説明問題の不正解例(出力)

ここでの問題は、addSlow(T x)メソッドがハッシュテーブルのある範囲が満杯になる状況を引き起こし得ることにあります。addSlow(T x)メソッドは、与えられた要素をハッシュテーブルの最初のnullエントリに格納します。しかし、これはハッシュ値が衝突した場合、ハッシュテーブル内の連続するセグメントがすぐに満杯になる可能性があることを意味します。具体的な操作のシーケンスとしては以下のようなものが考えられます：
n回のaddSlow(x)操作を行い、特定のハッシュ値の周辺にn個のアイテムが集まります。このステップでは、ハッシュ値が衝突するため、各addSlow(x)操作はハッシュテーブルの一部が既に満杯であるためO(n)の時間がかかります。
次に、上記で追加したn個のアイテムをすべて削除(remove(x))します。このとき、ハッシュテーブルでは実際にはnullに置き換えられずにdelマークに置き換えられます。
最後に、上記で削除したアイテムを探すfind(x)操作をn回行います。各検索操作は、ハッシュテーブルの該当するセグメントがdelマークで満たされているため、O(n)の時間がかかります。
以上から、全体の操作はO(n^2)の時間がかかることがわかります。n回のaddSlow(x)、remove(x)、そしてfind(x)の各操作がそれぞれO(n)の時間を必要とします。これはaddSlow(T x)が最初のnullエントリを使用するために、特定のハッシュ値周辺が満杯になる可能性があるためです。
一方、教科書のadd(x)メソッドではこの問題は生じません。なぜなら、このメソッドでは新たな要素が追加される際に、最初に見つかったnullエントリではなく、ハッシュ関数によって計算された位置か、その位置が既に占有されている場合はその次のnullエントリを使用します。これにより、特定のハッシュ値の周辺が急速に満杯になるという事態は避けられます。したがってadd(x)メソッドではO(1)の時間で操作を完了することができます。
また、find(x)関数もadd(x)関数と同様に、ハッシュ関数によって計算された位置から検索を開始しnullエントリに達するまで順次探索を進めます。このため、特定のハッシュ値周辺が満杯になるという事態が発生したとしてもO(n)の時間複雑度を保証できます。

受講者のレポートにもよくある誤り。別々のオブジェクトがハッシュ値の衝突を起こすことが前提となるが、この場合はaddも同様にO(n^2)かかる。

受講者へのLLMの使用について

- 出題時の指示
 - LLMの利用は禁止しない
 - LLMを利用した時には使用したダイアログを提出すると評価する
- 解答者のLLMへの言及
 - 「アルゴリズムの仕組みとコードはChatGPTを参考にした。」というのが1件. ダイアログは提供されず.
 - プログラムは少なめのプロンプトでChatGPTが解答する内容と一致. コメントは独自.

「情報数理学2」の課題まとめ

- 18題の課題のうち9題に追加の指示なしに一応正解する.
- 図の入力, 図の出力は現在は非対応(3題)
- 教科書のデータ構造については, 与えなくても解答できることもあるが, 与えた方が解答しやすい.
 - 一般的でないデータ構造については, 追加で指示を出しても正解に至らないこともある.
- 説明問題もあまり一般的でない設問では正解できないことも.
- 学習に教科書の「[Open Data Structures](#)」が使われている可能性もある.
 - CC by CAなのでLLM学習の入力に使えるはず.
 - ChatGPTに著者が誰かを英語で聞いてみると答えてくれる.
 - 不正解のプログラムの断片に, [Open Data Structures](#)の別のデータ構造で使われたコードの一部を参考にしていると思われるものが出ることが.

プログラミング課題とLLM

- 好ましくないLLMの利用
 - 内容を理解しなくても課題の説明を LLMに入れて、出力をそのまま提出
- 好ましい(あるいは今後は避けられない)LLMの利用
 - 自分の作成したプログラムの修正を LLMと対話してデバッグ
 - 関連資料をどこで探すか LLMにアドバイスを求める。
- プログラミングで何が重要かが変わってきたのと同様に、今後も変わることは避けられないのか。
 - 機械語 -> アセンブリ言語
 - アセンブリ言語 -> 高級言語
 - 型チェック, IDE等々
 - github copilotを入れると `def fib(n):` まで入れるとフィボナッチ数を求める関数の補完を提案する。

LLMに対応したプログラミング課題

- 図の入出力
 - 2023年7月現在の多くの対話式LLMは図の入出力をサポートしていないので、図の入出力をおこなう課題には対応できない。
 - あくまで現在の話、そのうち、「人間が見て分かる図はだいたい理解できる」ようになって不思議はない。
- 自分で例を作らせて説明する。
 - 設問自体がありふれている場合は、学習データの中に同じように例を作って説明させる文が含まれている場合もあり、その場合は無効
- 一般的な課題とずらした内容
 - 「フィボナッチ数列」以外の漸化式、通常とは変わった関数名、引数等、対応可能な場合も、どれを入力に含めるのかなどが難しい
- LLMに正解を出させるための入力を作成させる課題
 - 問題を解かせるために、どのような前提を与えなければいけないかを考えさせることも勉強になる

「好ましくないLLMの利用」を受講者がしようと思わないような授業をすることが必要