1 Техническое задание

1.1 Описание

Необходимо разработать короткий Python скрипт, для генерации математической задачи. Размер скрипта около **150-200** строк кода, при генерации без картинки. **300-400** при генерации с картинкой.

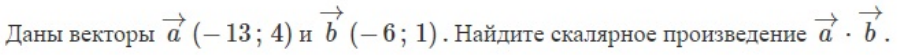

Рис. 1: Исходная задача

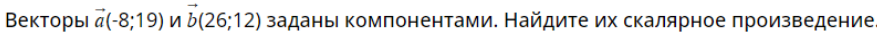

Рис. 2: Сгенерированная задача

1.2 Основные требования

В процессе генерации задачи, необходимо удовлетворить некоторым основным требованиям, которые и вносят значительный вклад в сложность скрипта.

1. **Случайные шаблоны** - перед генерацией задачи, необходимо задать сет предсгенерированных шаблонов с условием. Для генерации лучше всего использовать ChatGPT. Важно, **ни один из шаблонов не должен в точности повторять исходную формулировку** задачи. В процессе генерации, для каждой задачи необходимо выбрать случайный паттерн и подставить туда правильные параметры.

```
patterns = [
    "Даны векторы a→({a1}; {a2}) и b→({b1}; {b2}). Найдите их скалярное произведение.",
    "Векторы a→({a1}; {a2}) и b→({b1}; {b2}) заданы своими компонентами. Определите их скалярное произведение.",
    "Определите скалярное произведение векторов a→({a1}; {a2}) и b→({b1}; {b2}).",
    "Найдите скалярное произведение векторов a→({a1}; {a2}) и b→({b1}; {b2}).",
    "Векторы a→({a1}; {a2}) и b→({b1}; {b2}) заданы компонентами. Найдите их скалярное произведение."
]
```

Рис. 3: Пример паттернов

2. **Генерация параметров** - для генерации задачи необходимо выбрать некоторые параметры. Сделать это можно перебором в каком-то промежутке, или же абсолютно случайно. Главное требования - **все задачи должны быть уникальными**, а значит в случае случайного выбора необходимо вести set параметров, во избежание дубликатов.

Так же стоит отметить, что **все задачи должны иметь адекватные параметры**, сопоставимые по сложности с исходной задачей. Так в примере Рис. 1 самое сложное действие - это произведение 13 x 6. В этом плане произведение из Рис. 2 - 26 x 8 или 19 x 12 имеет повышенную сложность, но все еще сопоставимую. Но если бы необходимо было выполнять операцию на порядок сложнее: например трехнзначное число умножить на двузначное, то такие параметры бы считались неадекватными.

3. **HTML оформление** - сгенерированная задача должна быть красиво оформленна в HTML виде, **используя теги <math>**, если внутри используются формулы. Помимо этого нужно учитывать другие визуальные ньюансы. Например если имеется шаблон (a + b), и генерируются параметры от -10 до 10. То в случае прямой вставки отрицательных чисел вместо b, может получится например следующая строка (5 + -3). В таком случае необходимо отдельно обрабатывать место шаблона, чтобы получить строку вида (5 - 3). Аналогичные проблемы связаны с сокращением дробей и тд.

4. **Целый или конечный ответ** - так как задачи должны решаться онлайн без преподавателя, ответ должен быть **однозначно трактуемым для комьютера и человека**, а также его должно быть легко вписать в поле для ввода. Так например ответ корень из 2 не подойдет, тк может быть проблемотично вписать его в поле для ответа.

   Поэтому предполагается, что в большинстве задач ответом являетяся целое число - что сильно уменьшает количество возможных вариант при генерации. Так же в некоторых задачах, например про вероятность - ответ не может быть целым. Поэтому предполагается конечный десятичный вид. Например: 0.125 или 0.3 и так далее.

5. **Определение сложности** - для каждой задачи необходимо определять сложность, а также делать случайную выборку из сгенерированного пула в определенном соотношении по сложности. Например, всего возможно сгенерировать 1000 разлчиных задач. В финальной выборке должно быть 50 задач. Из них 60 процентов легких, 30 процентов средних и 10 процентов сложных.

   Сложность задачи оценивает ее **относительную сложность в рамках одного условия**, но с разными параметрами. Чтобы определить сложность задачи, предлагается полагаться на здравый смысл. Так если основным действием в задаче является взятие корня, то очевидно что произвести данную операцию над числами 4, 9, 16, 25 и тд. до 100 - очень просто, тк это квадраты первых 10-ти чисел. Таким задачам предлагается давать сложность легко. Если же квадрат берется для степени двойки или 5-ки, то задача будет средней, тк данные числа на слуху у школьников. Например корень 256 или 625. Если же это что-то большое, как 361 и тд. то данная задача точно будет сложной.

По такой же логике необходиом оценивать сложность и для других задач / операций.

Важно, иногда может оказаться, что нельзя сгенерировать больше 10 простых задач. В таком случае нужно взять все простые задачи и добрать какое-то количество средних и сложных, чтобы сохранить пропорцию.

1.3 Выход скрипта

Основным артефактом работы скрипта - является JSON файл, с массивом сгенерированных задач. Пример сгенерированного файла с массивом из одного элемента:

В данном JSON объекте присутсвуют как обязательные, так и необязательные поля. Дальше будут описаны все обязательные и некоторые дополнительные поля.

1. **uid** - обязательное поле, произвольная строчка из 10 символов. Допускаются только **заглавные** английские символы или цифры.

2. **task_num** - обязательное поле, порядковый номер задачи в тесте. Выдается вместе с исходной задачей.

3. **task_subject** - обязательное поле, предмет задачи. Всегда принимает значение **math**.

4. **task_types** - обязательное поле, массив типов задачи. Выдается вместе с исходной задачей.

5. **fipi_uid** - обязательное поле, уникальный номер задачи с сайта ФИПИ. Выдается вместе с исходной задачей.

6. **task_text** - обязательное поле, текстовое описание задачи в читаемом для машины тексте. Необходимо для того, чтобы удобно читать задание из базы данных, пересылать ее в телеграмме, отправлять в wolfram для решения или ChatGPT. Например некоторые математические символы могут пропасть при копировании. Например корень или степень. Поэтому предлагается в данном поле записывать их буквенно, как sqrt() и т.д.

7. **task_html** - обязательное поле, условие задачи оформленное в формате html. Корневым тегов является . Для описание математических формул можно использовать тег <math> и остальные математические теги. Также могут пригодится таблицы и тд.

8. **task_answer** - обязательное поле, ответ на задачу. Тип поля - строка.

9. **task_solution** - обязательное поле, оформленное в HTML решение задачи, с подставленными частными параметрами.

10. **task_hints** - обязательное поле, массив оформленных в HTML подсказок для решения задачи. Иногда будет достаточно завернруть просто текст в тег. Данный массив должен содержать все формулы и свойства, которые могут пригодится при решении задачи.

11. **answer_type** - обязательное поле, тип ответа. Выдается вместе с исходной задачей. Скорее всего просто short.

12. **task_group_key** - обязательное поле, ключ группы задачи. Каждый генератор должен иметь уникальный код, обозначающий тип генерируемой задачи, ее уникальное свойство - о чем задача.

13. **task_group_label** - обязательное поле, имя группы задачи. Каждый генератор должен иметь уникальное имя, расшифровку ключа.

14. **task_images_svg** - дополнительное поле, массив содержимого svg картинок.

```
[    {
        "uid": "AY197DO5G6",
        "task_subject": "math",
        "task_text": "Отрезки AC и BD - диаметры окружности с центром O. Угол ACB раве
        "task_html": "<p>Окружность с центром в точке O имеет диаметры AC и BD. Величи
        "task_solution": "<p>Угол <math><mi>AOD</mi></math> является смежным для угла
        "task_answer": "68",
        "task_images_svg": [
            "<?xml version='1.0' encoding='UTF-8'?>\n<svg xmlns=\"http://www.w3.org/20
        ],
        "task_hints": [
            "<p>Центральный угол равен удвоенному вписанному углу, опирающемуся на ту
            "<p>Используйте свойство, смежных углов. Сумма смежных углов - равняется 1
        ],
        "task_group_label": "Поиск центральногго угла, через вписанный и смежный",
        "task_types": [
            "7.1 Фигуры на плоскости"
        ],
        "task_group_key": "circle_task_a",
        "answer_type": "short",
        "task_num": 1,
        "fipi_uid": "80A34C",
        "difficulty": "medium"
    }]
```


Listing 1: tasks.json

1.4 Промпт ChatGPT

Для простого начала, предлагается использовать следующий промпт для ChatGPT. Данный запрос позволит быстро сгенерировать основу Python скирпта.

1.4.1 Генерация задачи

Первое сообщение: Помоги мне написать Python скрипт для генерации следующей задачи:

<Условие задачи>

Мне необходимо получить 1) Условие задачи 2) Решение задачи 3) Ответ задачи 4) Сложность задачи [Добавь какое-то условие для автоматической проверки на сложность. Например если все операции проводятся с числами до 10, то это легкая задача. Сделай возможность редактировать функцию оценки сложности]

Второе сообщение: Давай хранить сгенерированную задачу в JSON формате с следующими полями:

1. uid - уникальный номер из 10 случайных, заглавных букв или цифр.
2. task_subject - всегда равняется math.
3. task_text - условие задачи без использования спец символов. Например вместо знака корень используй функцию sqrt(). Это нужно для того, чтобы было удобно читать условие из базы данных.
4. task_html - условие задачи оформленное с использованием HTML. Необходимо для красивой записи формул. Используй теги <p> <math> <mi> <mo> <ms> и тд.
5. task_answer - ответ на задачу в виде строки.
6. task_solution - решение задачи с подставленными параметрами, оформленное с помощью HTML.
7. task_hints - массив подсказок к задаче, которые являются строками, так же оформленными в формате HTML для красоты. Должен содержать различные формулы которые используются при решении или свойства. Пример подсказки: "Используйте формулу для вычисления длины гипотенузы ...".
8. answer_type - всегда равняется short.
9. difficulty - одно из трех: easy, medium, hard.

Сгенерируй 1000 задач, возьми 40 случайных и сохрани в файл tasks.json - массив JSON-ов задач.

Третье сообщение: Давай добавим вариативности за счет использования случайных пресетов для условия задачи для task_html. Сгенерируй

5 равнозначных текстов для условий, при этом ни один текст не должен повтоярть исходный. И дальше во время генерации выбирай случайный и подставляй туда параметры. Условия должны быть на русском языке, а кодировка utf-8.

1.4.2 Генерация картинки

Используя библиотеку from lxml import etree, напиши скрипт на Python для генерации похожей картинки.

<Вставьте картинку из задачи>

<Опишите какие параметры к картинке должны подаваться на вход>

1.5 Полезные функции

1.5.1 Генерация JSON файла

```
def save_tasks_to_json(tasks, filename='tasks.json'):
    with open(filename, 'w', encoding='utf-8') as f:
        json.dump(tasks, f, ensure_ascii=False, indent=4)
```

1.5.2 Генерация HTML-ей из массива JSON-ов

```
import os

def generate_html_files(tasks, output_folder='html'):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    else:
        for file_name in os.listdir(output_folder):
            file_path = os.path.join(output_folder, file_name)
            if os.path.isfile(file_path):
                os.unlink(file_path)

    for i, task in enumerate(tasks):
        task_html = task['task_html']
        task_images_svg = task['task_images_svg'][0]

        prev_link = ''
        next_link = ''
        if i > 0:
            prev_uid = tasks[i-1]['uid']
            prev_link = f'<a href="{prev_uid}.html">Previous Task</a>'
        if i < len(tasks) - 1:
            next_uid = tasks[i+1]['uid']
            next_link = f'<a href="{next_uid}.html">Next Task</a>'
```

```
html_content = f'''
<html>
<head>
    <meta charset="UTF-8">
    <title>Task {task["uid"]}</title>
</head>
<body>
    {task_html}
    <div>{task_images_svg}</div>
    <div>
        {prev_link} | {next_link}
    </div>
</body>
</html>
'''

file_name = os.path.join(output_folder, f'{task["uid"]}.html')
with open(file_name, 'w', encoding='utf-8') as html_file:
    html_file.write(html_content)
```

1.6 Вспомогательные файлы

Репозиторий с шаблонами
HTML + JS для тестирования