Программирование: Рассеянные токены

Предыстория

Янис устал платить за подписку на ChatGPT и хочет разработать собственную большую языковую модель (LLM), чтобы помочь себе и разработчикам Lemon AI.

> ***Большие языковые модели (известные по-английски как Large Language Models или LLM)*** *— это нейронные сети, которые генерируют текст. Они кодируют язык в виде* ***токенов*** *— фрагментов текста, таких как слова, части слов или символы (например,* `Hello`*,* `AI`*,* `olymp`*,* `iad`*,* `!` *— это токены, кодирующие* `Hello AI olympiad!`*).*

Традиционный подход к созданию LLM заключается в использовании моделей на основе трансформеров.

> ***Трансформер*** *— это революционная архитектура, представленная в статье «Attention is All You Need» Васвани и др. в 2017 году. С тех пор она стала базовой архитектурой для многих задач NLP (обработки естественного языка), включая генерацию текста.*

Однако Янис знает о недостатке трансформеров: они являются авторегрессивными по своей природе, что ограничивает скорость и эффективность генерации текста.

> ***Авторегрессивные модели*** *должны генерировать токены по одному в порядке: после генерации токена он добавляется к результату, а следующий токен генерируется на основе обновленного результата. Это ограничивает генерацию последовательностью.*

Янис хочет исследовать **диффузионные модели**, которые могут генерировать токены в произвольном порядке для более быстрого параллельного вывода.

> ***Диффузионные модели*** *не являются авторегрессивными и могут генерировать токены в любом порядке, а не только слева направо.*

Однако Янис знает о проблеме диффузионных моделей — они часто сводятся к авторегрессивному генерации: генерируют токены последовательно, слева направо. Чтобы этого избежать, Янис ограничивает порядок генерации: сначала генерируются несмежные токены, а затем остальные.

Прежде чем приступить к созданию Diffusion LLM, Янис хочет создать прототип и измерить, насколько он эффективен.

У него есть очень специальный графический процессор, который потребляет больше энергии по мере увеличения значений токенов на входе и выходе. Чтобы минимизировать затраты, он хочет вычислить **накопленные затраты на электроэнергию** (как описано ниже).

Янис придумал следующую задачу:

Задача

У нас есть последовательность из n токенов: $P = [p_1, p_2, \ldots, p_n]$, где каждый токен p_i имеет энергозатраты, равные его значению. Нам нужно найти оптимальный порядок генерации, который минимизирует **накопленные энергозатраты** и предотвращает **авторегрессионный коллапс**.

Накопленная стоимость энергии для последовательности $[a_1, a_2, \ldots, a_n]$ равна:

```
sum = a₁ + (a₁ + a₂) + (a₁ + a₂ + a₃) + ... + (a₁ + a₂ + ... + aₙ)
```

Например, $[1, 3, 9]$ имеет накопленные затраты $= 1 + (1 + 3) + (1 + 3 + 9) = 18$

Ограничение, предотвращающее **авторегрессионный коллапс**, заключается в следующем:

- Разделите токены на две группы по четности их положения в исходной последовательности: нечетные позиции (1-я, 3-я, 5-я, ...) и четные позиции (2-я, 4-я, 6-я, ...)
- Все токены одной четности должны быть выведены до токенов другой четности.

Вывод должен представлять собой перестановку (переупорядочение) исходной последовательности P.

Входные данные

Первый аргумент содержит целое число n ($1 \le n \le 10^5$) — длину последовательности.

Второй аргумент содержит n целых чисел $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 10^9$), разделенных пробелами — элементы последовательности.

Вывод

Выведите n целых чисел, разделенных пробелами, представляющих оптимальную действительную перестановку.

Если существует несколько оптимальных решений, вы можете вывести любое из них.

Ограничения

- Количество тестовых примеров ≤ 100
- Сумма n по всем тестовым примерам $\le 5 \times 10^5$
- Ограничение по времени выполнения: 5 секунд
- Ограничение по памяти: 512 МБ

Примеры

Входные данные:

```
3
5 1 4
```

Выходные данные:

```
1 4 5
```

Этот ответ следует чередующемуся шаблону: сначала выбирается токен в четной позиции (2-й: значение 1), затем токены в нечетных позициях (1-й и 3-й: значения 5 и 4, отсортированные как 4, 5). Накопленная стоимость составляет 1 + (1+4) + (1+4+5) = 16, что является оптимальным результатом.

Оценивание

Оценка = пройденные тесты / общее количество тестов

Дальнейшее изучение

Если вам интересно, как работают LLM, посмотрите это видео от 3Blue1Brown: http://youtube.com/watch?v=LPZh9BOjkQs

Если вы хотите узнать больше, продолжите просмотр его серии видео о нейронных сетях после видео о LLM! В плейлисте есть гостевое видео от Welch Labs о том, как работают трансформеры: https://youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi