Distribuidor de capacidade

Modelos largos de linguagem (*large language models* ou LLMs) são modelos de inteligência artificial (*artificial intelligence* ou AI) que analizam linguagem natural e são treinados com uma vasta quantidade de textos. Textos são decompostos em **tokens**, e o processo de inferência envolve vários rounds onde uma sequência de tokens chamada **input** são processados por uma sequência de transformadores generativos pré-treinados (*Generational Pre-trained Transforms* ou GPTs) a fim de calcular uma lista de quais os próximos tokens mais prováveis e suas probabilidades. O próximo token mais provável é adicionado ao input e o processo se repete. A sequência de tokens gerada a partir do input inicial é chamada de **output**.

A principal forma de medir o uso de LLMs é o *throughput* do total de tokens processados (input + output). A unidade é **tokens por minuto** (*tokens per minute* ou TPM).

Você é um desenvolvedor de uma plataforma de AI que provê serviços de LLM para clientes. Para cada cliente é designada uma **capacidade** (*capacity*) medida em TPM. Cada cliente pode criar um **projeto** dentro da plataforma. Dentro de cada projeto o cliente pode criar aplicações de LLMs, como agentes de AI, data pipelines com etapas de inferência, chatbots, visão computacional, entre outros.

Cada projeto acaba por ter um **uso** (*usage*) de LLMs, e um **limite** (*limit*), medidos em TPM. Quando o uso do projeto está acima do limite, as aplicações de LLM dentro do projeto reduzem seu *throughput*, o que as tornam mais lentas e menos responsivas. O **excesso** é o quanto o uso está acima do limite.

Para cada projeto, o cliente pode configurar:

- Um **limite mínimo** (*maximum limit* ou $minLimit$), o menor valor para o limite daquele projeto. A soma dos limites mínimos não pode ser maior que a capacidade do cliente. O intuito é preservar usabilidade mínima.
- Um **limite máximo** (*minimum limit* ou $maxLimit$), o maior valor para o limite daquele projeto. O intuito é conseguir controlar o gasto de tokens de um único projeto.
- Uma **importância** (*importance*), medido em uma unidade arbitrária. O limite de um projeto deve ser **proporcional** à importância do projeto desde que não viole os limites máximo e mínimo. O intuito é dar ao cliente controle para priorizar um projeto sobre outro quando estes competem por recursos.

Seu trabalho como desenvolvedor dessa plataforma de AI é criar um algoritmo que distribui dinamicamente a capacidade do cliente pelos seus projetos, ou seja, que calcula

o limite de cada projeto em determinado instante baseado no uso daquele instante. O algoritmo deve respeitar as condições acima e deve minimizar o excesso de forma ótima. O algoritmo deve garantir que a soma do mínimo entre o limite e o uso de cada projeto não ultrapasse a capacidade do cliente. Ou seja, $\sum_{i=0}^{n-1} \min(usage[i], limit[i]) \leq capacity$.

Esboço do código:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class CapacityDistributor {

    public record Project(int minLimit, int maxLimit, int importance) {}

    static List<Integer> distribute(List<Project> projects, List<Integer>
usage, int capacity) {
        return List.of();
    }

    public static void main(String[] args) {
        List<Project> projects = List.of(
            new Project(100, 1000, 1),
            new Project(100, 1000, 1),
            new Project(100, 1000, 3),
            new Project(100, 1000, 5),
            new Project(0, 100, 5));
        List<Integer> usage = List.of(
            0,
            800,
            800,
            100,
            400);
        List<Integer> expectedLimits = List.of(
            200,
            200,
            600,
            1000,
            100);
        List<Integer> actualLimits = distribute(projects, usage, 1000);
        System.out.println("Wanted: " + expectedLimits);
        System.out.println("Got: " + actualLimits);
    }
}
```