Blueprint do Projeto: Compressão de Pesos de LLM (v2)

Fase 1: Análise Exploratória e Caracterização da Fonte (Concluída)

- **1.1 Engenharia Reversa:**
  - ☒ Identificar estrutura: Header JSON + Payload Binário contíguo.
  - ☒ Identificar tipo de dados: `BF16` (2 bytes: 1 sinal+expoente, 1 mantissa).
- **1.2 Análise Estatística:**
  - ☒ Calcular Entropia Global: ~6.22 bits/byte.
  - ☒ Calcular Entropia Condicional: ~5.36 bits/byte.
  - ☒ **Byte-Splitting (Descoberta Chave):**
    - Entropia MSB (Expoente): **2.71 bits/byte** (Alta redundância).
    - Entropia LSB (Mantissa): **7.96 bits/byte** (Ruído quase aleatório).

Fase 2: Benchmarking de Referência (Em Progresso)

Objetivo: Estabelecer as métricas a bater.

- **Objetivo 2.1: Testes com Compressores Padrão:**
  - ☒ Executar `gzip` (níveis -1 e -9), `bzip2`, `xz` e `zstd` sobre o ficheiro original.
  - ☒ Registar: Tamanho Final, Tempo de Compressão/Descompressão, Pico de RAM.
- **Objetivo 2.2: Testes Específicos (Opcional):**
  - ☐ Testar ferramentas especializadas em floats (`fpzip` ou `zfp`) apenas para comparação teórica, se houver tempo.

Fase 3: Desenvolvimento do Codec "Split-Stream" (O Core)

Estratégia Definida: Arquitetura Híbrida baseada na separação MSB/LSB.

3.1 Módulo de Pré-processamento (Splitter): * [x] **Implementado:** Leitura por blocos (1MB) e separação vetorial MSB/LSB no `encoder_core.cpp`.

- **3.2 Compressão do Canal MSB (O "Cérebro"):**
  - ☒ **Teste de Predição Delta ($x_n - x_{n-1}$):** Falhou (Entropia subiu para 3.28). Causa: Bit de sinal do BF16.
  - ☒ **Teste de Predição XOR ($x_n \oplus x_{n-1}$):** Falhou (Entropia 3.11). Causa: Falta de correlação sequencial nos expoentes.
  - ☒ **Decisão:** Usar dados **Raw** (Entropia 2.70).
  - ☒ **Codificação de Entropia:**
    - Implementado **Huffman Estático** (Modo Fast): ~633 MB.

    - ∗ Implementado **Aritmética Estática** (Modo Best): ~631 MB.
- **3.3 Compressão do Canal LSB (Tarefa Pendente - Colega):**
  - ☒ **Implementar RLE Simples:** Detetar apenas sequências de zeros (comum em *sparsity*).
  - ☒ **Fallback:** Se o RLE aumentar o tamanho, manter cópia direta (*Raw Copy*).
- **3.4 Empacotamento e Descodificação:**
  - ☒ **Formato de Ficheiro (.sc):** Definido como `[Header Tamanho][Tabela Freq][Bitstream MSB][Dados LSB]`.
  - ☒ **Decoder:** Falta criar o `decoder.cpp` para inverter o processo e reconstruir o `.safetensors` original (prova de conceito).

Fase 4: Otimização e Pontos de Operação

Objetivo: Criar as variantes "Fast" vs "Best" exigidas no enunciado.

- **Objetivo 4.1: Tuning de Parâmetros:**
  - ☒ **Modo "Fast":** Split + Huffman Estático + LSB Raw.
  - ☒ **Modo "Best":** Split + Aritmética Adaptativa + LSB (tentativa LZ).
- **Objetivo 4.2: Gestão de Memória:**
  - ☐ Refinar o tamanho dos *chunks* de leitura para garantir que o compressor funciona em máquinas com pouca RAM.

Fase 5: Relatório e Apresentação

- **Objetivo 5.1: Escrita Técnica:**
  - ☐ Documentar a implementação do *Splitter* e do *Preditor*.
  - ☐ Comparar os resultados finais com o Benchmark da Fase 2.
- **Objetivo 5.2: Apresentação:**
  - ☐ Preparar slides focados na decisão de arquitetura ("Porquê separar os bytes?").