**🏗️ 1️⃣ Sugerir 3 algoritmos com características distintas**

1. **Bubble Sort** – (ordenamento simples)

* Muitas operações de comparação e troca, mas baixo uso de memória.
* Pouco paralelismo, muita dependência entre instruções.

1. **Fast Fourier Transform (FFT)** – (processamento matemático)

* Operações matemáticas intensivas, bom uso de paralelismo.
* Acessos regulares à memória, mas mais instruções de ponto flutuante.

1. **Busca Binária em vetor grande** – (acesso à memória)

* Poucas instruções, mas muitos acessos à memória.
* Eficiência depende da cache e da latência da memória.

Esses três são bem distintos: um é "control-heavy" (Bubble Sort), outro é "compute-heavy" (FFT), e o terceiro é "memory-heavy" (Busca).

**🔧 2️⃣ Sugerir 3 parâmetros do gem5 para variar**

🔸 **Tamanho da Cache L1 (2kB, 4kB, 8kB, 16kB)**

* **Por quê:** Aumentar o tamanho diminui faltas de cache e beneficia acessos repetitivos ou com padrões imprevisíveis.
* **Efeito esperado:**
  + **Bubble sort:** acesso sequencial e previsível, melhora moderada.
  + **FFT:** acesso misturado, ganha consideravelmente ao segurar mais dados.
  + **Binary search:** acesso disperso e imprevisível, grande ganho.

🔸 **Tamanho do Buffer de Fetch (4, 8, 16, 32 bytes)**

* **Por quê:** O buffer de fetch armazena instruções pré-buscadas antes da decodificação; um buffer maior permite maior “lookahead” e mantém a pipeline alimentada em códigos sequenciais ou com poucas interrupções de controle.
* **Efeito esperado:**
  + **Bubble sort:** melhora moderada, pois o acesso é muito sequencial e simples; buffers maiores suavizam eventuais stalls.
  + **FFT:** ganho considerável, dado o loop intenso e a necessidade de manter a pipeline cheia de instruções matemáticas.
  + **Binary search:** pouco ganho, pois branches imprevisíveis esvaziam o buffer com frequência.

🔸 **Associatividade da Cache L1 (1-way, 2-way, 4-way, 8-way)**

* **Por quê:** Menos vias aumentam conflitos de mapeamento, penalizando padrões dispersos; mais vias reduzem conflitos.
* **Efeito esperado:**
  + **Bubble sort:** sequencial, quase não sofre com conflitos.
  + **FFT:** acesso misto, impacto moderado com menos vias.
  + **Binary search:** disperso, sofre muito com baixa associatividade.

**🔒 3️⃣ Configuração Fixa (Base)**

Escolha uma configuração inicial, com:  
✅ Cache L1 de **16kB** (padrão 32kB)  
✅ Buffer de Fetch de **32 bytes** (padrão 64 bytes)  
✅ Cache L1 **8-way** (padrão 8-way)

**📊 Planejamento dos testes**

* Para cada programa (Bubble, FFT, Busca):
  + Simular com a configuração fixa.
  + Simular com:
    - Tamanho da L1 variando (2kB, 4kB, 8kB, 16kB)
    - Tamanho do Buffer de Fetch (8, 16, 32, 64)
    - Associat. da L1 variando (1-way, 2-way, 4-way, 8-way)
* Coletar **Tempo de execução** e **IPC**.

**🗒️ Resultados**

* Para cada parâmetro, fazer 2 gráficos:
  + X = níveis do parâmetro
  + Y = IPC ou tempo de execução
  + Linhas = 3 algoritmos
* Explicar cada gráfico com foco:
  + Por que o IPC e o tempo mudam?
  + Qual algoritmo se beneficia mais ou menos e por quê?
  + Custo-benefício das mudanças de configuração.