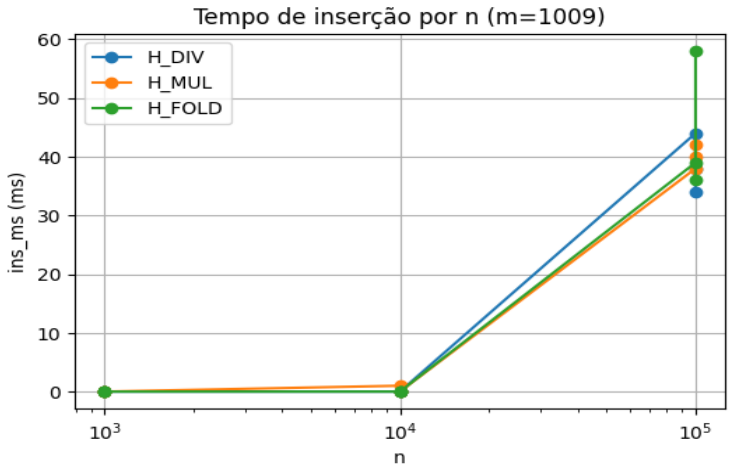
# Tabela Hash por Encadeamento Separado

## Metodologia

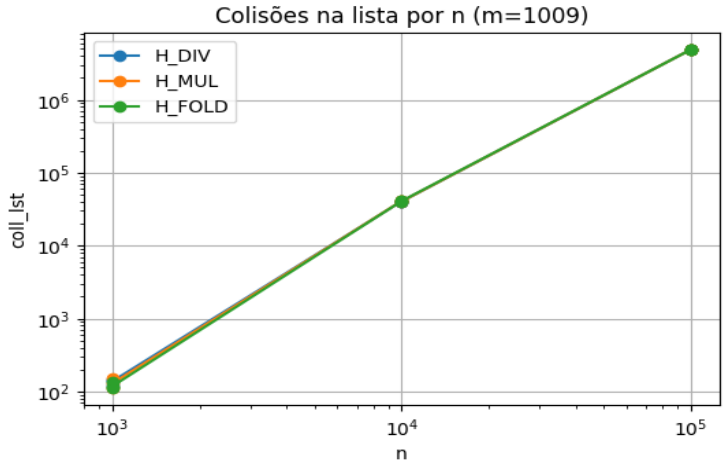
Usa-se exatamente os parâmetros (m ∈ {1009, 10007, 100003}, n ∈ {1000, 10000, 100000}, função de hashing ∈ {H_DIV, H_MUL, H_FOLD} e seed ∈ {137, 271828, 314159}). Utilizamos seed devidamente, registramos métricas de colisão e inserção. “Distribuições mais uniformes reduzem o custo médio no encadeamento separado.”

## Resultados

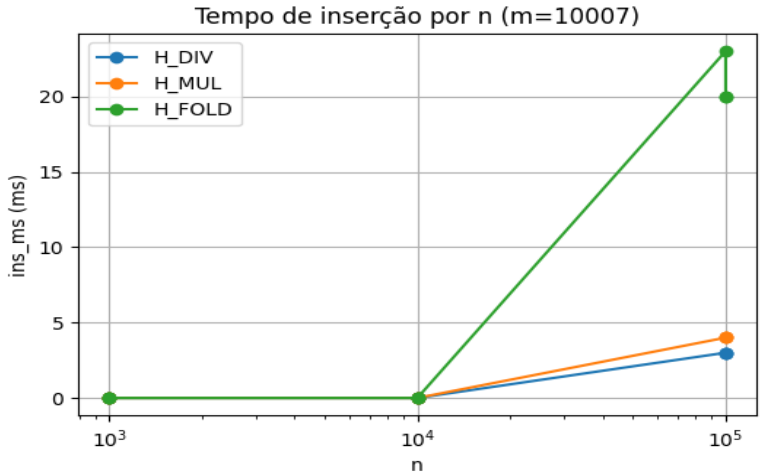
**1 - Tempo de inserção por n (m = 1009)**



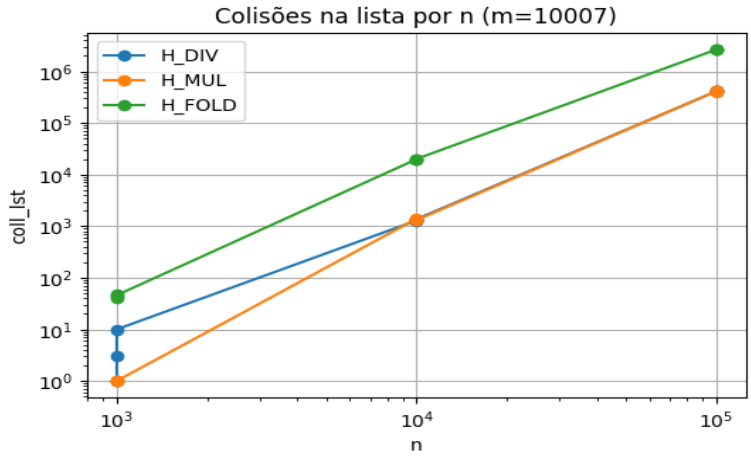

**2 - Colisões na lista por n (m = 1009)**



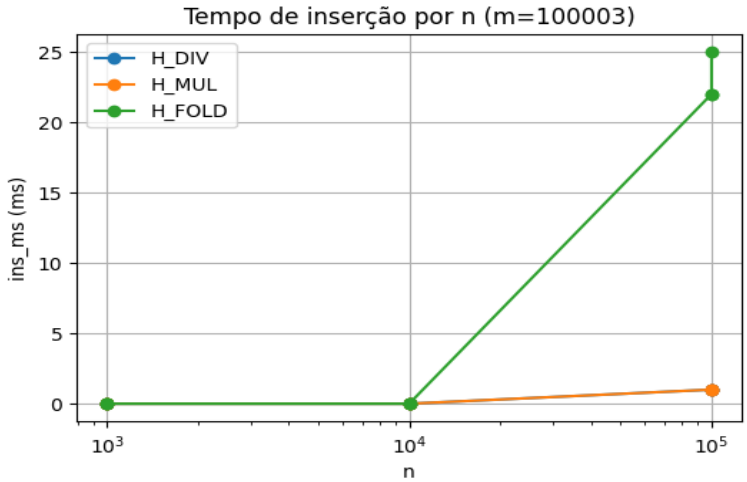

**3 - Tempo de inserção por n (m = 10007)**



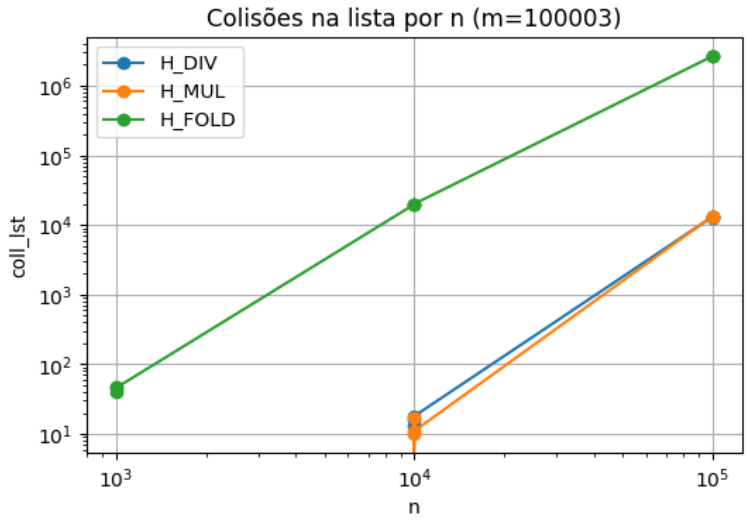

**4 - Colisões na lista por n (m = 10007)**




**5 - Tempo de inserção por n (m = 100003)**




**6 - Colisões na lista por n (m = 100003)**




## Conclusões -

**CARGA MAIOR ---**

Quanto mais alto o fator de carga (n>m), maiores as listas e mais frequentes as colisões. Note os gráficos de m = 1009: quando n = 100000 as listas ficam bem longas. Quando m é maior (10007 ou 100003) o problema fica menos aparente, porque os baldes dão mais espaço para as chaves.

**EFETO DA DISTRIBUIÇÃO --**

H_FOLD se destaca nas execuções por gerar bem mais colisões na lista, especialmente para m = 10007 e m = 100003. H_DIV e H_MUL, por outro lado, ficaram parecidos na maior parte dos testes, indicando que esses dois métodos distribuíram as chaves de forma mais igualitária.

**BUSCAS — HITS VS MISSES**

Quando a busca encontra a chave (hit), o custo cresce mais ou menos com o comprimento médio da lista até a posição da chave, então funções que mantêm listas curtas tendem a ter cmp_hits menores. Quando a chave não está, a busca normalmente percorre a lista inteira do balde, então cmp_misses fica bem maior em baldes longos.