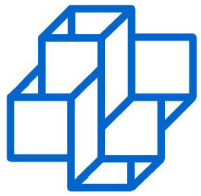


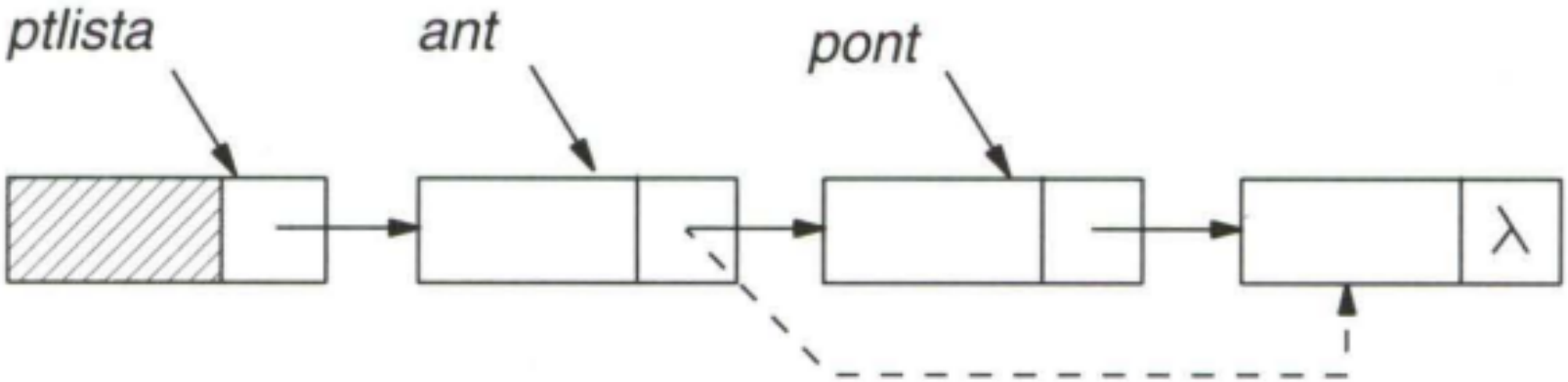
Laboratório Nacional de Computação Científica

Análise de complexidade para Listas Ligadas Simples e Duplas

Martha Hilda Timoteo Sanchez, Rafael Lemes Beirigo

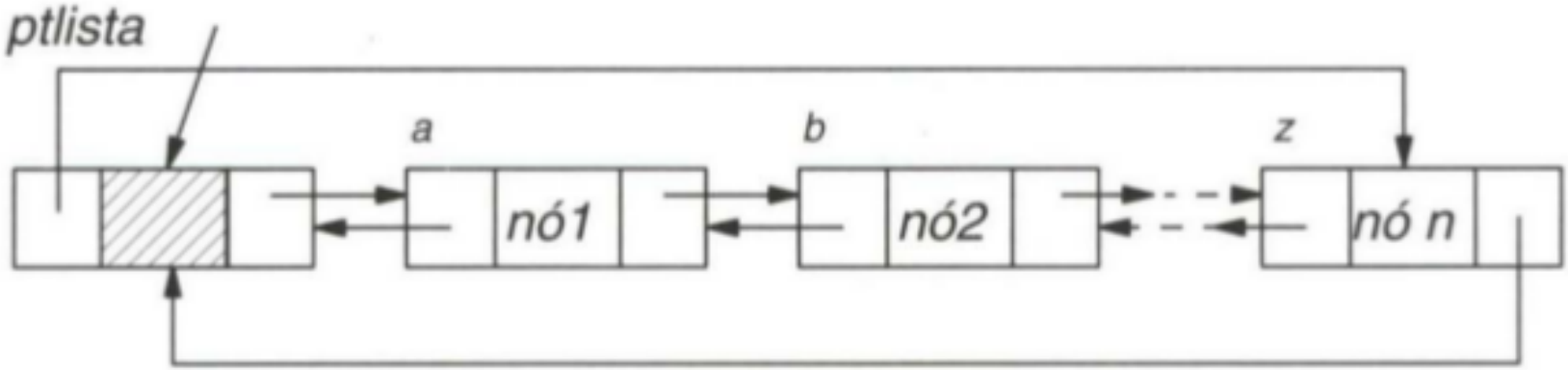
July 11, 2015

Um ponteiro: para o nó posterior



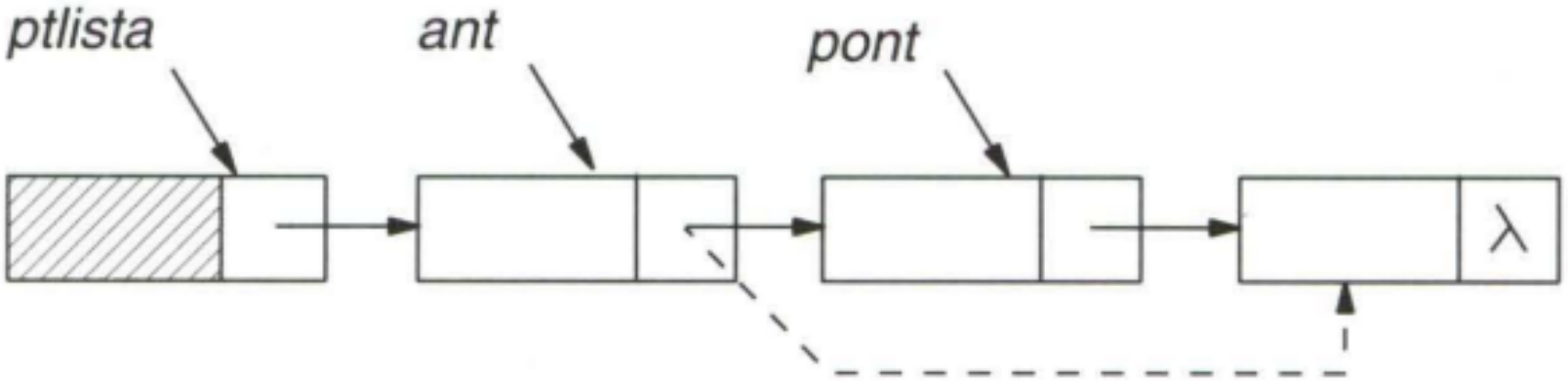
- Percurso unidirecional
- Remoção necessita de dois ponteiros

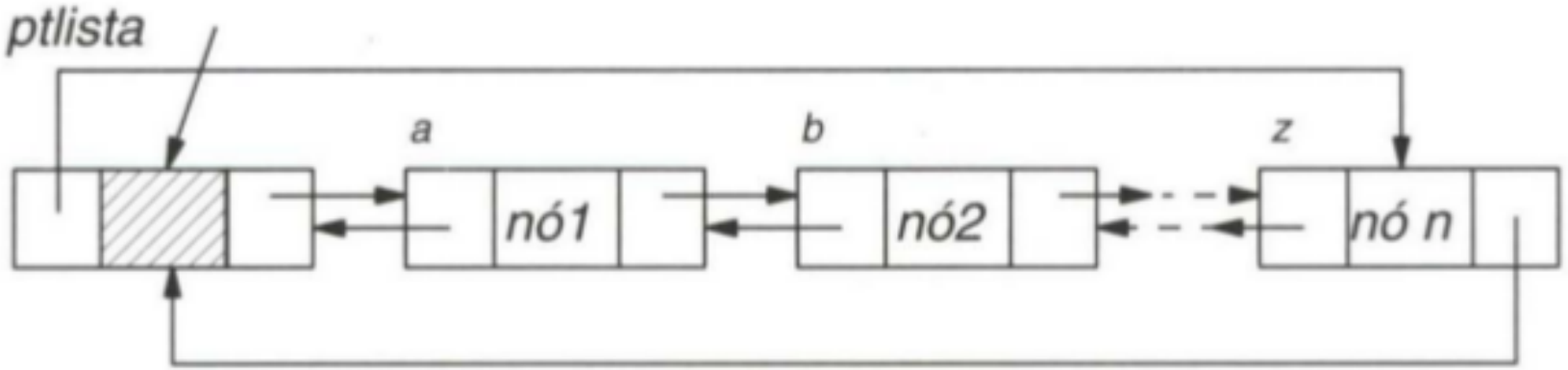
Dois ponteiros: para os nós anterior e posterior



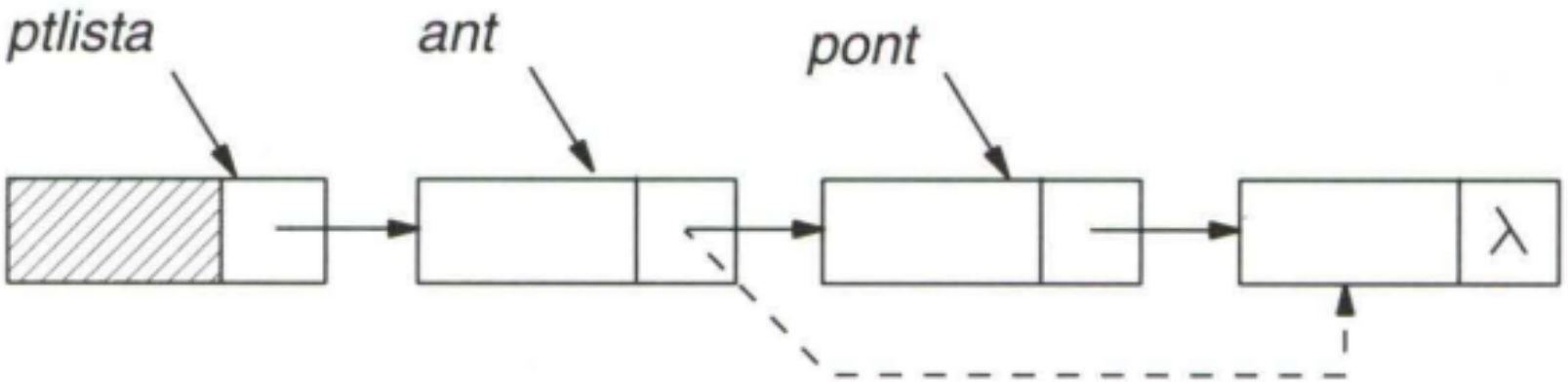
- Percurso bidirecional
- Remoção necessita de um ponteiro

Controlado no momento da inserção



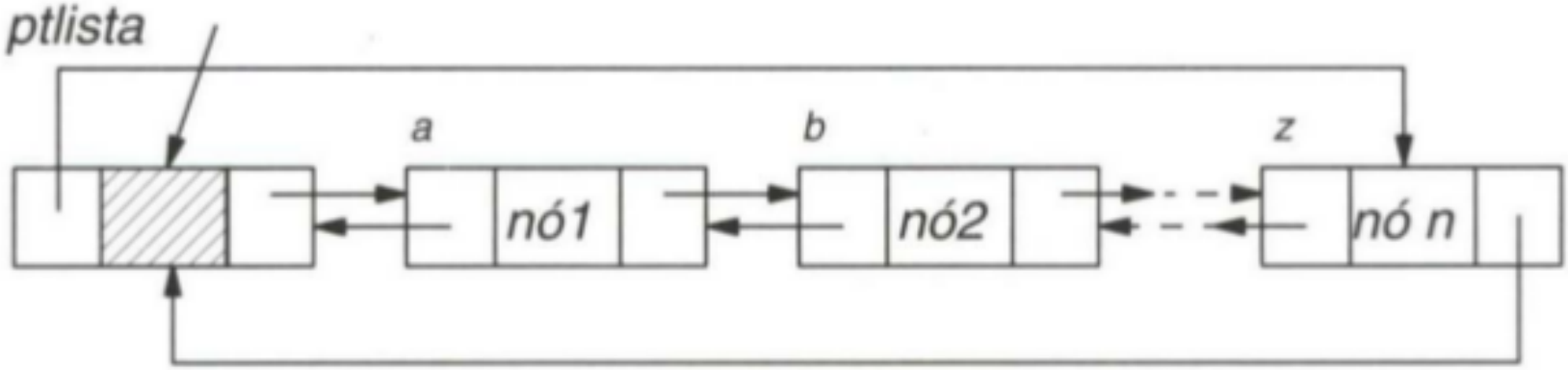


Lista Simples



- (+) Menor custo de memória
- (-) Remoção necessita de dois ponteiros
- Complexidade do pior caso
  - Percorre a lista completa para inserir/remover (busca)
  - $\mathcal{O}(n)$

Lista Dupla



- (+) Inserção imediata na última posição
- (-) Consome o dobro de memória para ponteiros
- Complexidade do pior caso
  - Percorre a lista até o penúltimo nó para inserir/remover (busca)
  - $\mathcal{O}(n-1) = \mathcal{O}(n)$

Algoritmo

```
for (qtd = 1; qtd <= MAX; qtd *= 2) { // mais pontos
    for (i = 0; i < (qtd / 2); ++i)    // (1e10)
        l=insercao_enc(rand() % MAX, l);

    time(&inicio_conta);
    for (i = 1; i <= CANT_TESTE; ++i) {
        num = rand() % MAX;
        l=insercao_enc(num,l);
        l=remocion_enc(num,l);
    }
    time(&fim_conta);
    total = fim_conta - inicio_conta;
}
```

Algoritmo

```
for (qtd = 1; qtd <= MAX; qtd *= 2) {
    for (i = 0; i < (qtd / 2); ++i) // completar a lista
        l=insercao_enc(rand() % MAX, l);

    time(&inicio_conta);
    for (i = 1; i <= CANT_TESTE; ++i) {
        num = rand() % MAX;
        l=insercao_enc(num,l);
        l=remocion_enc(num,l);
    }
    time(&fim_conta);
    total = fim_conta - inicio_conta;
}
```

Algoritmo

```
for (qtd = 1; qtd <= MAX; qtd *= 2) {
    for (i = 0; i < (qtd / 2); ++i)
        l=insercao_enc(rand() % MAX, l); // insere número
                                         // aleatórios
    time(&inicio_conta);
    for (i = 1; i <= CANT_TESTE; ++i) {
        num = rand() % MAX;
        l=insercao_enc(num,l);
        l=remocion_enc(num,l);
    }
    time(&fim_conta);
    total = fim_conta - inicio_conta;
}
```

Algoritmo

```
for (qtd = 1; qtd <= MAX; qtd *= 2) {
    for (i = 0; i < (qtd / 2); ++i)
        l=insercao_enc(rand() % MAX, l);

    time(&inicio_conta);
    for (i = 1; i <= CANT_TESTE; ++i) { // repetição foi
        num = rand() % MAX;             // necessária
        l=insercao_enc(num,l);          // (1e6)
        l=remocion_enc(num,l);
    }
    time(&fim_conta);
    total = fim_conta - inicio_conta;
}
```

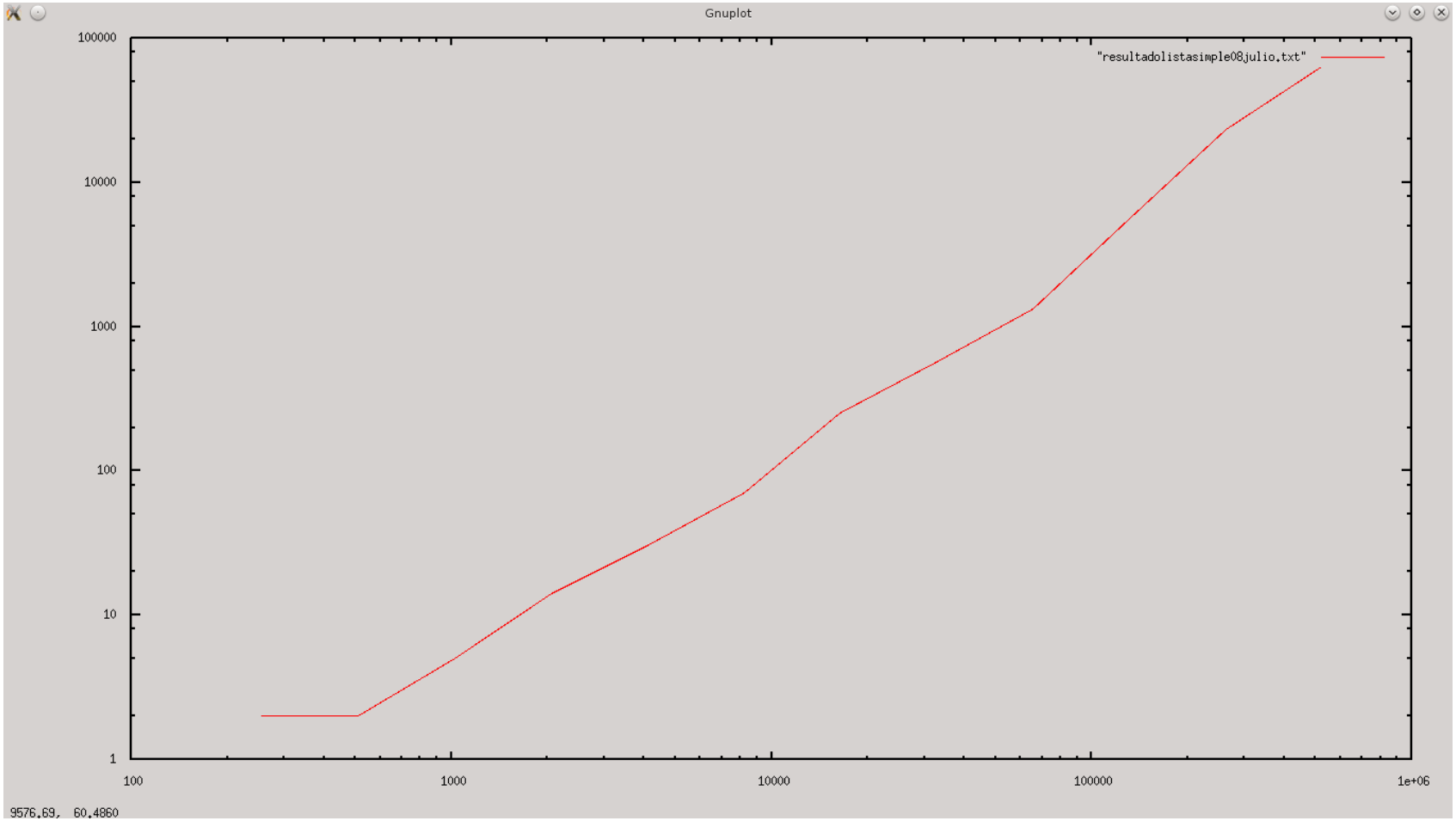
Lista Simples – pontos iniciais: rapidez *vs.* precisão

n	t
1	0
2	0
4	0
8	0
16	0
32	0
64	1
128	0

Lista Simples – a partir de $n = 256$

n	t
256	2
512	2
1024	5
2048	14
4096	30
8192	71
16384	253
32768	569
65536	1338
131072	5649
262144	23473
524288	63852

Lista Simples – Gráfico



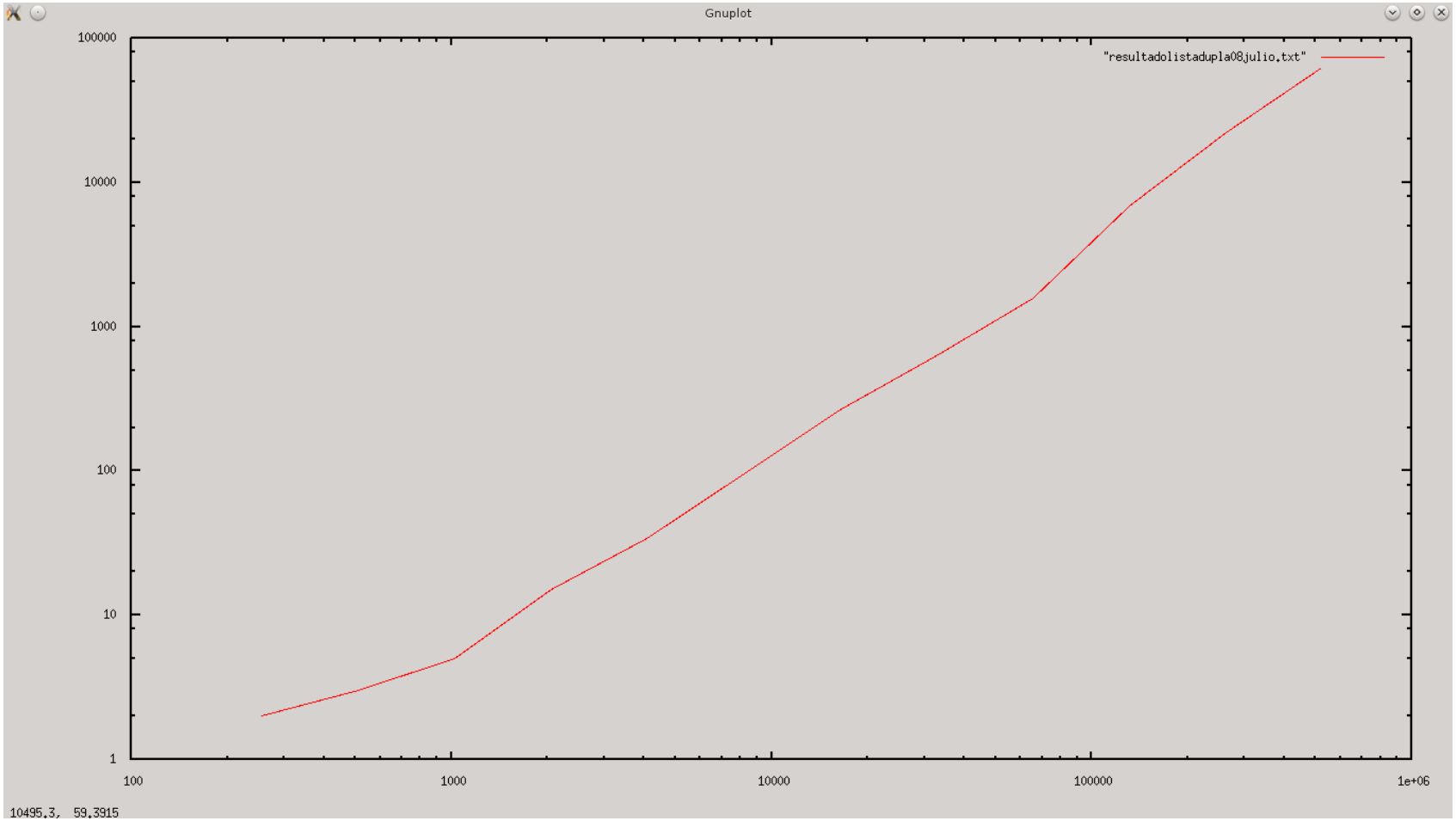
Lista Dupla – pontos iniciais: rapidez *vs.* precisão

n	t
1	0
2	0
4	0
8	1
16	0
32	0
64	0
128	1

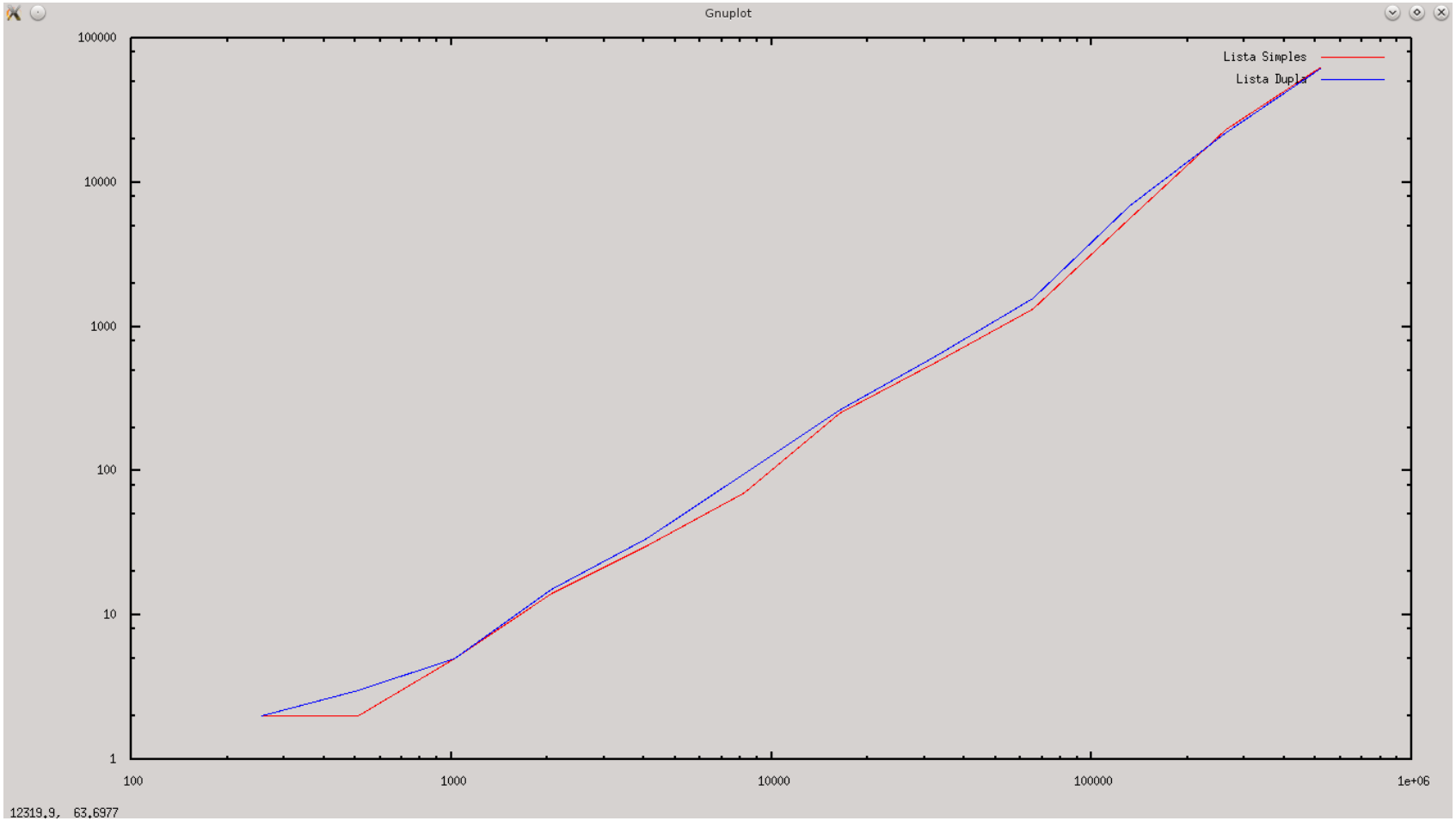
Lista Dupla – a partir de $n = 256$

n	t
256	2
512	3
1024	5
2048	15
4096	34
8192	96
16384	267
32768	633
65536	1560
131072	6936
262144	22018
524288	61744

Lista Dupla – Gráfico



Análise: desempenho equiparável (inesperado)



Possíveis razões

- Complexidade teórica considera pior caso, i.e., inserção sempre na última (simples) e penúltima (dupla) posições
- Experimento realizou inserções de números aleatórios
  - Posições aleatórias
- Resultado corresponde, portanto, à complexidade do caso médio, onde se espera que o desempenho seja equiparável