Ternary Search

O **Ternary Search** é um algoritmo de busca que divide a lista em três partes, em vez de duas como no **Binary Search**. Ele compara o elemento de busca com dois pontos de divisão e elimina uma parte da lista em cada iteração. Isso pode, teoricamente, reduzir o número de comparações em algumas situações.

Como Funciona o Ternary Search:

1. O algoritmo começa com a lista ordenada.
2. Divide a lista em três partes, calculando dois pontos de divisão: `mid1` e `mid2`.
3. Compara o valor de busca com os elementos nos índices `mid1` e `mid2`:
    - Se o valor de busca for igual a `mid1`, o índice `mid1` é retornado.
    - Se o valor de busca for igual a `mid2`, o índice `mid2` é retornado.
    - Se o valor de busca for menor que `mid1`, o algoritmo continua a busca na primeira parte da lista.
    - Se o valor de busca for maior que `mid2`, o algoritmo busca na terceira parte da lista.
    - Caso contrário, a busca continua na parte do meio.

Diferença entre Ternary Search e Binary Search:

- **Binary Search** divide a lista em duas partes e faz uma comparação.
- **Ternary Search** divide a lista em três partes e realiza duas comparações.
- O **Binary Search** tem complexidade de tempo O(logn)O(\log n)O(logn), enquanto o **Ternary Search** tem a mesma complexidade de O(logn)O(\log n)O(logn), mas com uma constante maior devido às duas comparações a cada divisão.

Desempenho do Ternary Search vs. Binary Search:

1. Em Listas Pequenas:

- Para listas pequenas, a diferença de desempenho entre **Binary Search** e **Ternary Search** é mínima. Ambos os algoritmos têm uma complexidade de tempo de O(logn)O(\log n)O(logn), mas o **Ternary Search** requer um pouco mais de comparações a cada iteração (duas comparações em vez de uma).

2. Em Listas Grandes:

- Para listas grandes, **Ternary Search** pode ser ligeiramente mais lento que o **Binary Search** devido ao custo adicional das comparações. Embora ambos tenham complexidade O(logn)O(\log n)O(logn), o **Binary Search** tende a ser mais eficiente, já

que ele faz uma única comparação por iteração, enquanto o **Ternary Search** faz duas comparações por iteração.

3. Casos em que o Ternary Search é mais eficiente:

- O **Ternary Search** pode ser mais eficiente em cenários onde a lista é extremamente estável e as comparações a mais, embora não otimizem diretamente o tempo, podem reduzir o número de passos recursivos. Contudo, o **Binary Search** tende a ser mais popular e eficiente na maioria dos casos devido à sua simplicidade e baixo custo computacional.

Conclusão:

- Ambos os algoritmos têm complexidade de O(logn)O(\log n)O(logn), mas o **Binary Search** é geralmente mais eficiente em termos de tempo, pois faz menos comparações a cada iteração.
- O **Ternary Search** pode ser interessante em situações específicas, mas na prática, o **Binary Search** é preferido pela simplicidade e eficácia.