Análise do Uso da Fila Sequencial no Radix Sort

1. Introdução

No problema de **ordenação por distribuição**, precisamos organizar números inteiros em "baldes" de acordo com cada dígito, recolhendo-os na ordem correta a cada passo.
Neste trabalho, os baldes foram implementados como **Filas Sequenciais**, ou seja, vetores circulares com tamanho fixo.

2. Vantagens da Fila Sequencial

1. **Simplicidade de implementação**: a fila sequencial é fácil de implementar usando arrays, sem necessidade de ponteiros ou gerenciamento complexo de memória.
2. **Acesso rápido aos elementos**: como os elementos são armazenados em um vetor, o acesso por índice é muito rápido.
3. **Boa performance para tamanho limitado**: quando o número máximo de elementos é conhecido, não há overhead de alocação dinâmica.
4. **Previsibilidade**: o uso de um vetor fixo evita fragmentação de memória e facilita o controle de índices no algoritmo.

3. Desvantagens da Fila Sequencial

1. **Tamanho fixo**: a fila sequencial depende de um tamanho máximo pré-definido. Se o número de elementos exceder esse limite, a fila não consegue armazenar novos elementos.
2. **Desperdício de memória**: se o vetor for muito grande e a fila raramente atingir a capacidade máxima, há memória alocada que não é usada.
3. **Gerenciamento circular**: para manter a fila circular, é preciso controlar cuidadosamente os índices de início e fim; erros podem causar sobreposição de elementos ou acesso indevido.
4. **Menor flexibilidade que filas encadeadas**: diferentemente de filas encadeadas, não é possível crescer dinamicamente sem realocação do vetor.

4. Conclusão

A **Fila Sequencial** é adequada para o Radix Sort quando:

- O número de elementos é conhecido previamente.
- Deseja-se simplicidade e acesso rápido.

No entanto, para aplicações que exigem crescimento dinâmico ou números muito grandes, uma **Fila Encadeada** seria mais flexível, apesar de exigir mais complexidade de implementação.