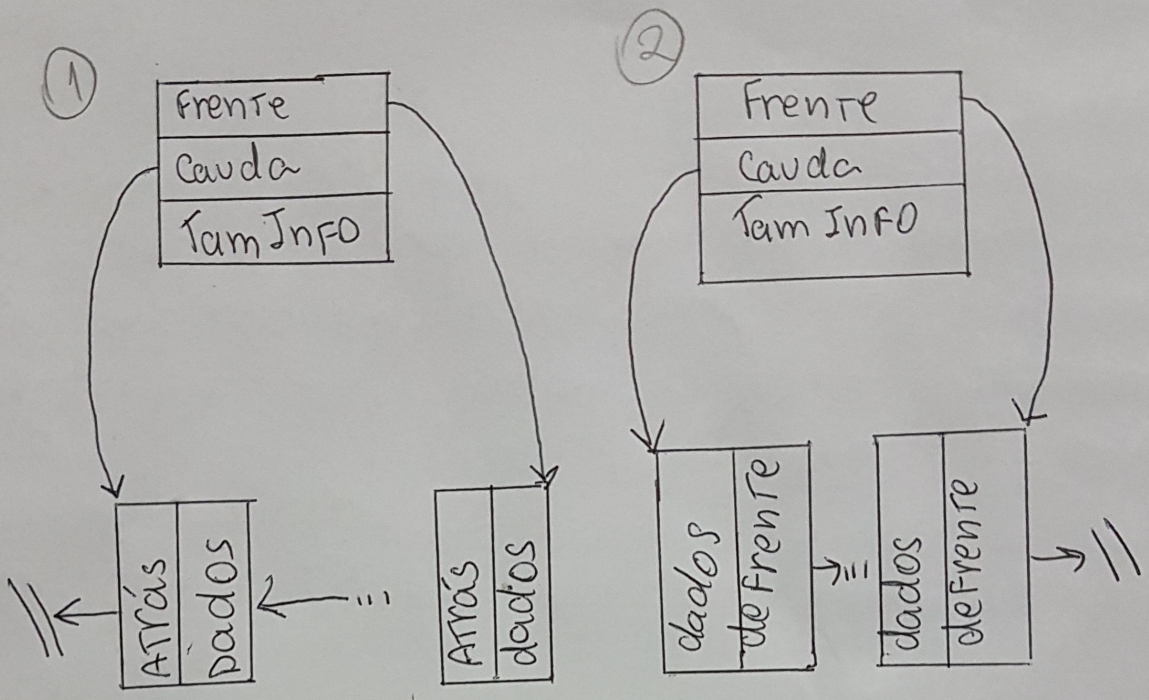
Fila Simplesmente Encadeada



1) Criação:
Ambos modos de implementação terão a eficiência equivalente.

2) Buscas:
Ambos implementações terão a mesma desempenho em memória e processamento porém, em tempo de procura, cada uma pode ser mais rápida que a outra, dependendo da localização do elemento. Se o elemento estiver mais próximo do fim da fila, a ② implementação será mais rápido, já, se o elemento estiver próximo do início da fila, a ① implementação será mais veloz. E por fim, se o elemento estiver no meio da fila, ambos serão iguais. 1.1

3) Destruição:
Ambos terão a mesma eficiência

3) TESTE SE ESTA VAZIA OU CHEIA:
Ambas terão a mesma desempenho

4) Inserção Ambas vão terão que arrumar os ponteiros.
A única diferença na inserção é que, na primeira implementação terá um processo a mais, a última estrutura deve apontar para NULL. Porém isso é "insignificante em termos de processamento".
Sequencia de passos para inserir elemento na implementação (1)

1: Inserir 2: atrás → elemento_inserido 3: Cauda → elemento inserido
4: elemento_inserido → NULL

Sequencia de passos para inserir elemento na implementação (2)
1: Inserir 2: elemento_inserido → elemento_dfpronte 3: cauda → elemento_inserido

5) Remoção
A primeira implementação é mais eficiente já que demanda menos processamento para cada inserção. Em grande escala, pode fazer grande diferença. Na primeira implementação, a primeira estrutura da fila já possui o próximo elemento armazenado na variável atrás. ou seja, em uma remoção, já temos o endereço da proxima estrutura. Ja na segunda implementação, não temos armazenado o endereço da estrutura do próximo elemento da fila, apenas o da frente, tendo que percorrer toda a fila através de ponteiros para saber o endereço para que a variável frente deverá apontar em caso de remoção

Índice de comentários

1.1 Para as filas que trabalhamos as buscas ocorrem apenas nas extremidades buscaNaCauda(...) ou buscaNaFrente(...), então, em termos das buscas não há diferença.