Hash

Em geral você tem chaves que você quer guardar em alguma estrutura, de modo que a busca seja bem rápida. Por exemplo, os nomes dos alunos podem ser chaves - e a partir deles você quer achar no seu banco informações diversas, tipo as notas deles ou whatever.

Pra isso, você vai ter uma funçãozinha (usando diminutivos à la Claudson) chamada hash. Ela faz umas contas lá e retorna um número. Tipo, digamos que hash("marcos") = 8. Sempre que a gente calcular vai ser 8, não é aleatório, tá.

A ideia é ter um array, e guardar as informações sobre "marcos" na posição 8. Assim, toda vez que você tiver que procurar, é só calcular o hash (costuma custar pouco) e ir direto na posição certa. É pra isso que hash serve.

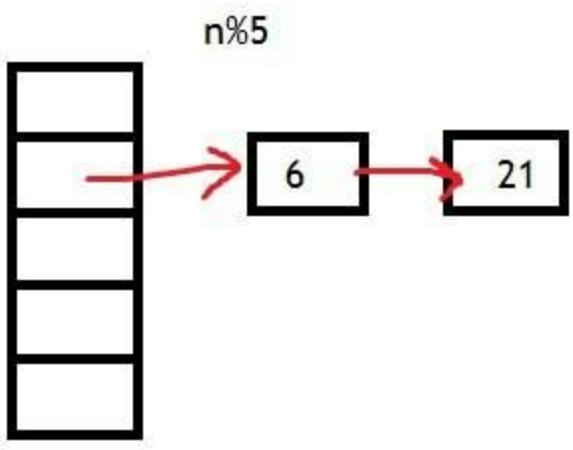
Na prova, as chaves vão ser números (pra ficar mais fácil) e a função de hash vai ser geralmente n mod p, também pra facilitar as coisas.

Imagina porém que hash("joaquim") = 8. Quando a gente for inserir, "marcos" já vai estar lá.

Cada tipo de tabela de hash tem uma forma diferente de tratar esse problema. É por isso que existem 5 tipos, eles só lidam com a colisão (dois hashs iguais) de formas diferentes.

Tabelas de hash não permitem duplicata. Se ele mandar você inserir duas vezes, você simplesmente diz "o professor está bêbado" e só insere uma.

Encadeamento exterior

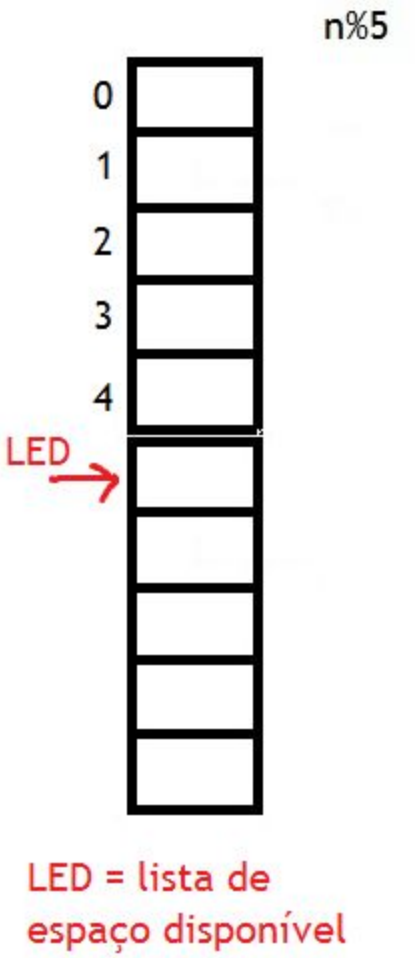


A tabela de hash é um array. Se a função pode gerar valores de 0 a n, então a tabela vai ter n+1 espaços. Cada espaço vai ter uma lista encadeada. Se dois colidirem, é só adicionar na lista e pronto.

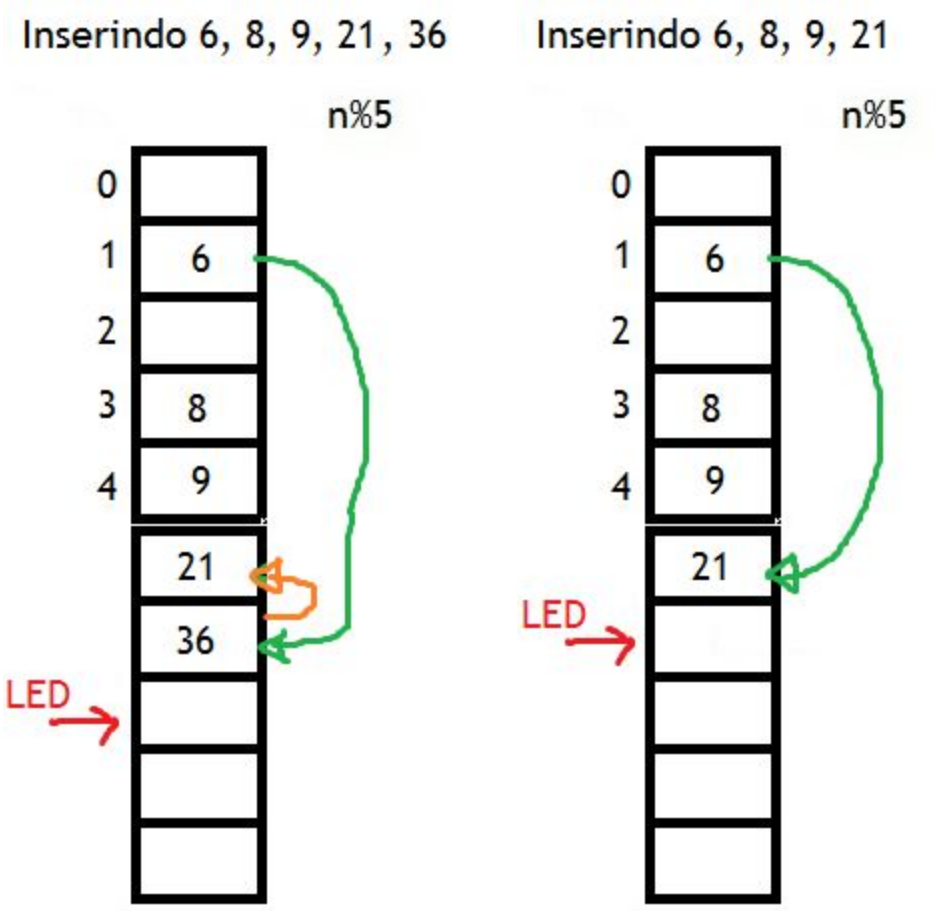
A remoção é idêntica a de uma lista encadeada. É só apagar o item que queremos apagar, fazer o anterior a ele apontar para o seu próximo e pronto.

Encadeamento Interior com Área de Colisão

Você vai criar um array maior, com espaço sobrando depois dos 5 espaços que teria pra mod5. Você vai ter um ponteiro pro primeiro espaço dessa área extra, que vamos chamar de área de colisão.

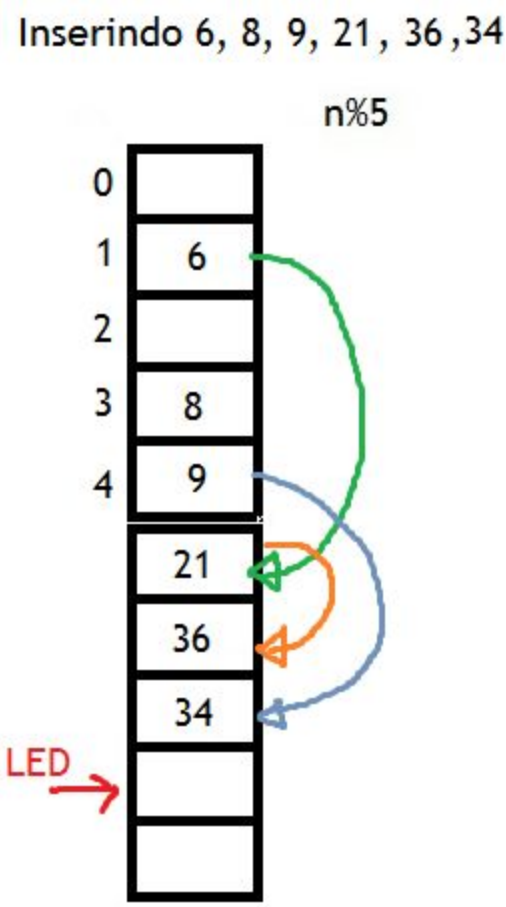


Todos esses espaços na tabela tem um ponteiro para o próximo, que é inicializado com NULL. Quando duas chaves tiverem o mesmo hash, você vai colocar a chave nova no espaço disponível, e vai usar o ponteiro da antiga para apontar para a nova.
Exemplo:



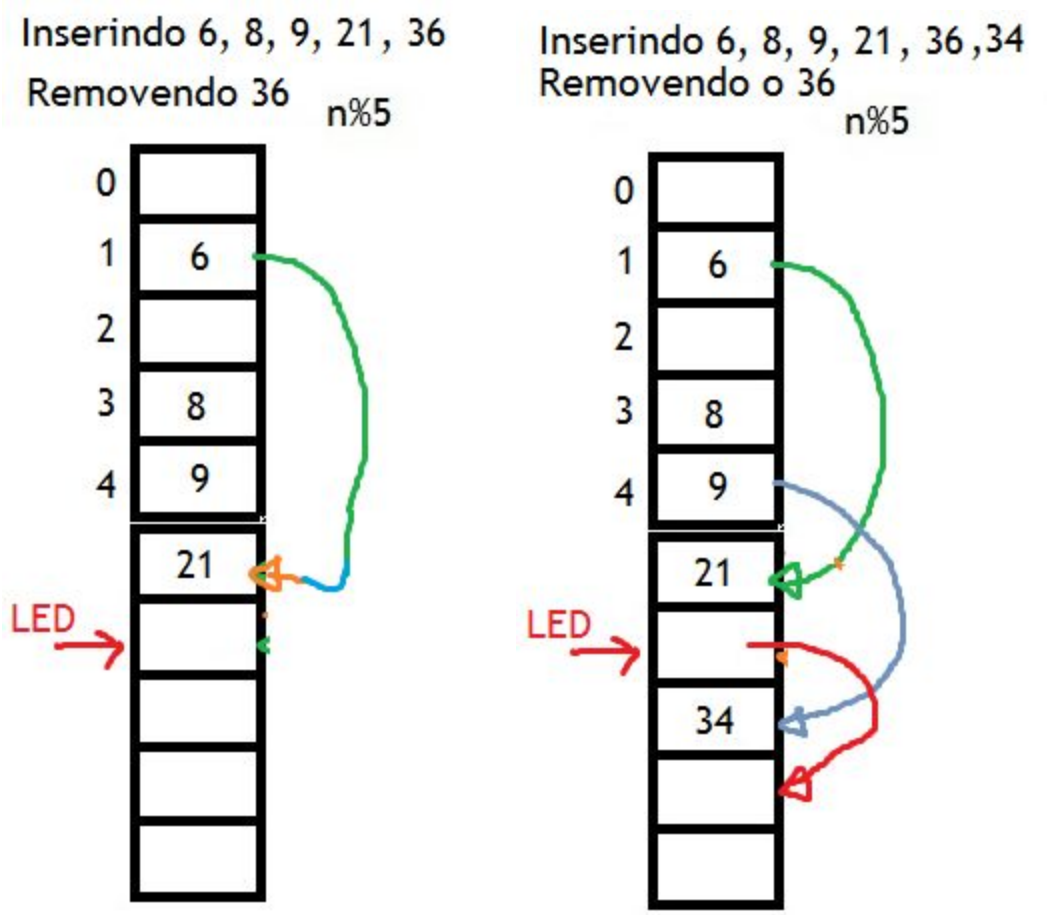
Lembre sempre de atualizar o ponteiro para a lista disponível.

Existem duas formas de inserir. Essa primeira demonstrada é a mais fácil de entender e visualizar, mas é mais chatinha de implementar. Na prática, na hora de criar o algoritmo, é mais fácil apontar sempre a chave que ficou no lugar “certo” para a chave nova, e depois passar a chave nova para onde a outra apontava. Por exemplo, na hora de inserir o 36, em vez de apontar do 21 para o 36, nós apontaríamos do 6 pro 36 e do 36 para o 21.



Na prática, os dois estão certos e se você não entender o segundo, não precisa se preocupar.

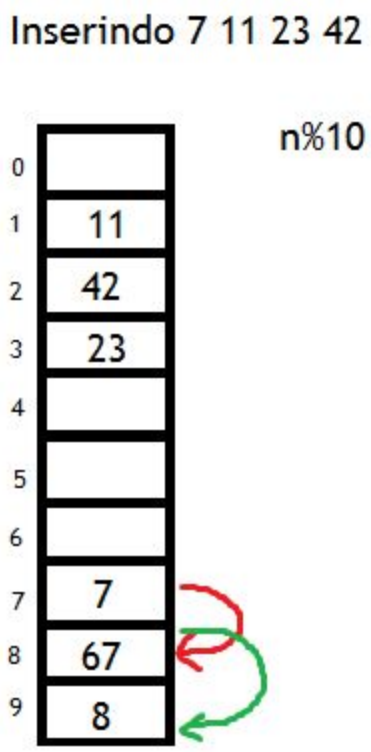
Remover é simples. Se o cara estiver dentro da área “normal”, você substitui ele por alguém para quem ele aponta, e rearruma os ponteiros. Se ele não aponta pra ninguém, é só remover. Lembrando sempre de atualizar a LED. Exemplo: remover o 36. Você só precisa apontar o 6 (que apontava para o 36) para o 21 (para quem o 36 apontava).



Exemplos de remoção.

Encadeamento interior sem área de colisão

Virou bagunça. Mais uma vez, todos os espaços têm um ponteiro para o próximo. Se houver colisão, você usa o próximo espaço disponível. Simples assim.

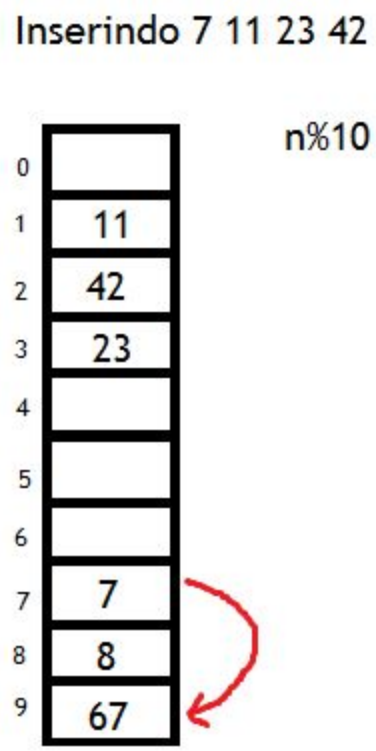


Perceba que, ao pesquisar 8, vamos ter de passar primeiro pelo 67 (que não tem nada a ver com 8!) antes de chegar no 8. Pois é. A esse probleminha damos o nome de coalescência. A próxima tabela contorna esse problema.

Encadeamento interno sem área de colisão e sem coalescência

É basicamente a mesma coisa que o anterior, mas no momento em que você for inserir o 8, você vai verificar: o número que está aí realmente devia estar aí? Como esse não é o lugar de direito do 67, o 8 vai ficar nesse lugar, e o 67 vai ter de ser colocado na próxima casa disponível.

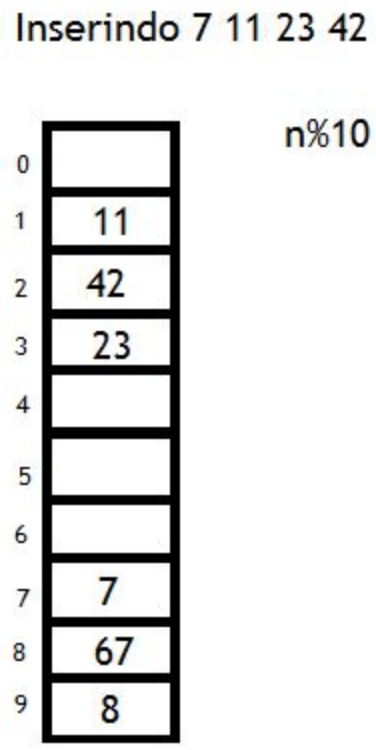
O exemplo em aula foi por volta de: imagine que você está entrando num avião e sua passagem diz que seu lugar é o 12. Se você chega lá e já tem alguém, e a passagem dessa pessoa diz 7, você vai dizer “Moço, esse lugar é meu” e a pessoa vai sair dali. Mas se a pessoa também tiver uma passagem com o número 12, então a companhia aréa provavelmente se enrolou, mas você vai ter paciência e vai sentar em outro lugar.



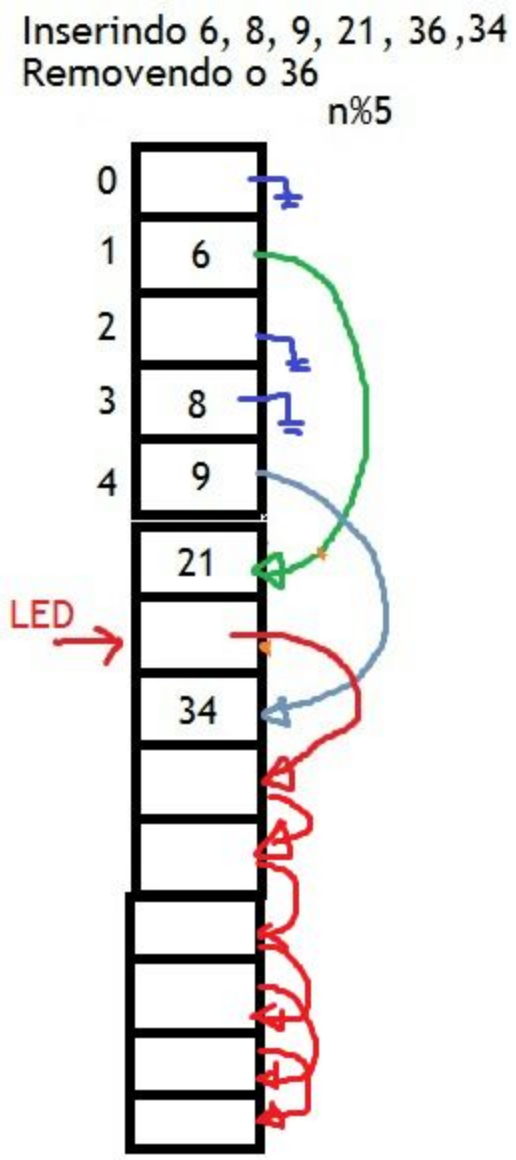
Exemplo de como ficaria sem coalescência

Endereçamento Aberto

É muito parecido com os dois anteriores, mas não usa ponteiros. Se hash[i] não é o que você queria encontrar, basta ir seguindo pelo array (i++) até encontrar o que você queria OU um espaço em branco (que significa que o que você quer não está lá).



Observações: na tabela com área de colisão, fica implícito que cada espaço na área de colisão aponta para o próximo. Todos os ponteiros (de praticamente todas as tabelas) são inicializados com NULL. Por exemplo:



Outro detalhe interessante é que a área de colisão pode ser tão grande quanto você quiser, nesse caso específico (Encadeamento interno com área de colisão). Geralmente se faz com o tamanho da tabela “normal” de cima.