ESTRUTURA DE DADOS

AGENDA

- Tabela de dispersão (hash table)

TABELA DE DISPERSÃO (HASH TABLE)

- Já vimos: Busca eficiente com complexidade O(logn)
- Que complexidade seria melhor? Por que?

$$O(1)$$

FUNÇÃO DE DISPERSÃO

- Uma matrícula m
- N espaços no vetor
- Sempre que possível ter as seguintes propriedades:
  - Eficientemente avaliada
  - Espalhar bem as chaves de busca

Ex: $m\%N$

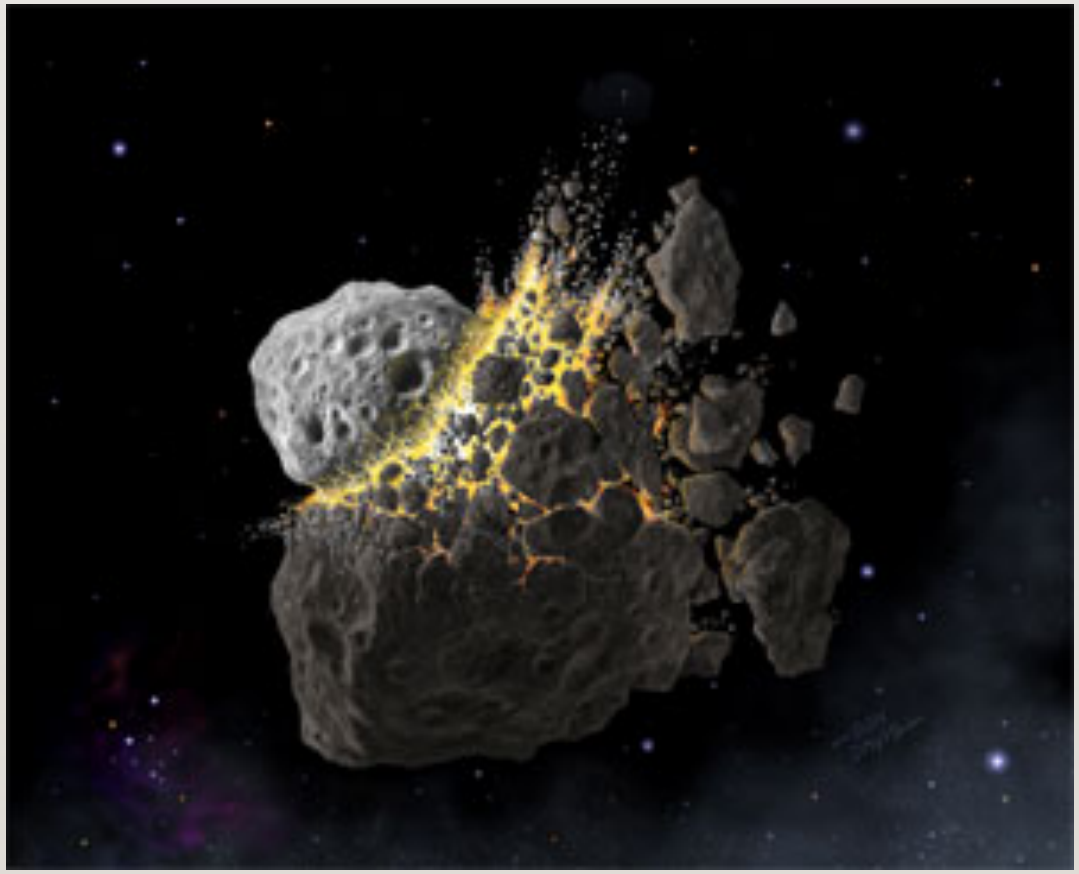
FUNÇÃO DE DISPERSÃO

```
#define N 101

static int hash (int mat){
        return (mat%N);
}
```

COLISÃO E TRATAMENTO

- O que é colisão?



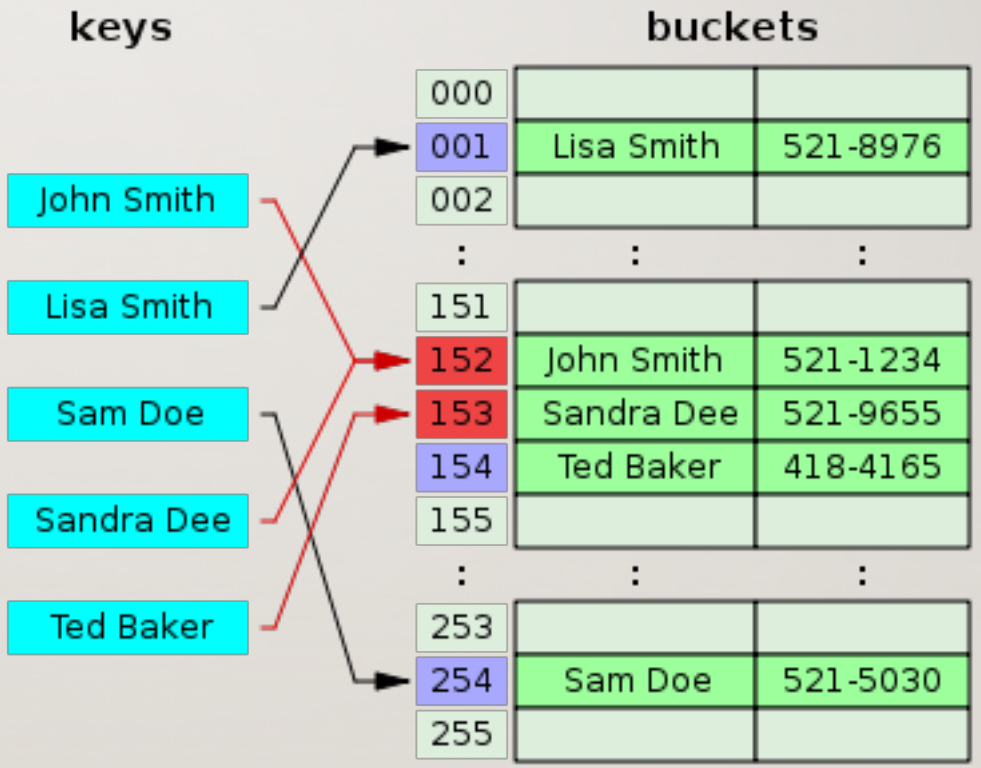
COLISÃO E TRATAMENTO

- N deve ser preferencialmente primo para diminuir colisões
- Vetores muito cheios podem perder desempenho. Empiricamente é adequado manter o tamanho ocupado em no máximo 75%
  - 50% traz bons resultados
  - Menor de 25% pode apresentar perda de desempenho

Ex: $m\%N$

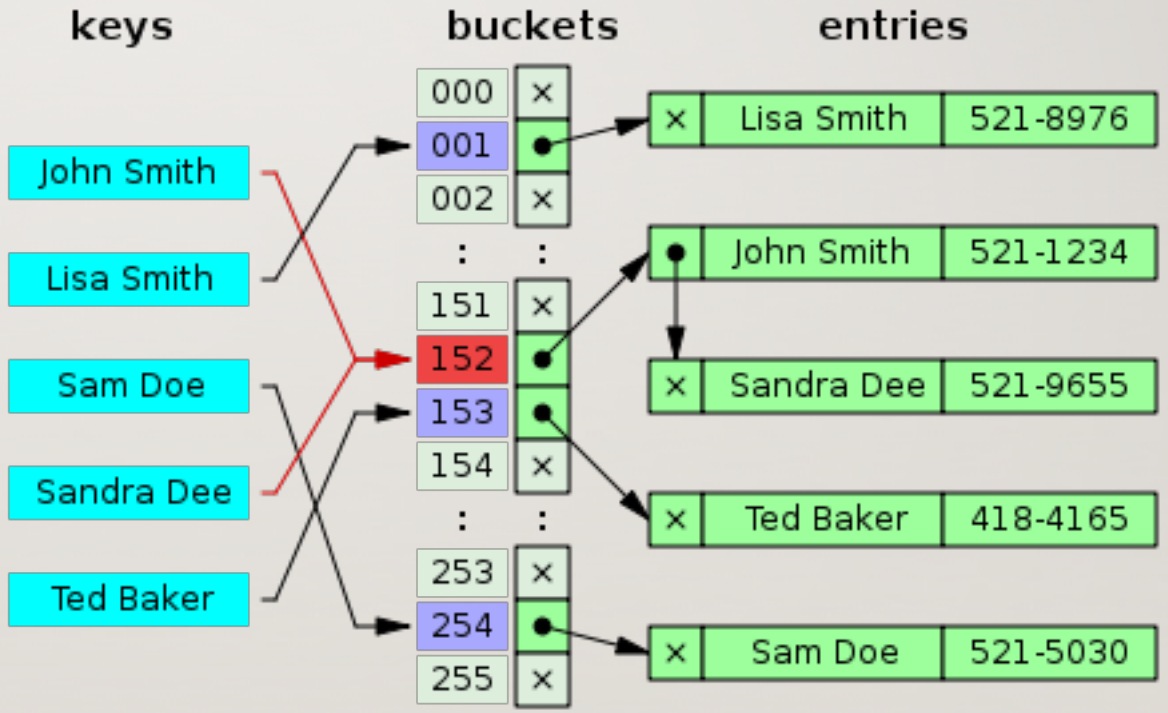
COLISÃO E TRATAMENTO

- Posição consecutiva livre: (h(x) + 1)%N
- Segunda função de hash: (h(x) + h`(x))%N



COLISÃO E TRATAMENTO

- Lista encadeada



EXEMPLO DE ESTRUTURA

```
typedef struct hash Hash;
hash {
        int n; //número de elementos na tabela
        int dim;  //dimensão da tabela
        Aluno** v;
}

int hash (Hash* tab, int mat){
        return (mat%tab->dim);
}
```

ATIVIDADES

- Implemente as demais operações da tabela de dispersão do arquivo hash.c. Tire os comentários das linhas a medida que implementa para testar se elas estão funcionando.

- https://github.com/gvanerven/estdados/blob/master/apoio/hash.c