MÉTODOS DE BUSCA

SCC0601 – Introdução à Ciência da Computação II

Prof. Lucas Antiqueira

Introdução

- Importância em estudar busca
  - Busca é uma tarefa muito comum?
- Vários métodos e estruturas de dados podem ser empregados para se fazer busca
  - Quais?
- Certas formas de organização de dados podem tornar o processo de busca mais eficiente

Introdução

- O problema da busca (ou pesquisa)

  "*Dado um conjunto de elementos, onde cada um é identificado por uma chave, o objetivo da busca é localizar, nesse conjunto, o elemento que corresponde a uma chave específica*"

Termos Relacionados

- Tabela: termo genérico, pode ser qualquer estrutura usada para armazenamento e organização dos dados

- Uma tabela é um conjunto de elementos, chamados registros

Termos Relacionados

- Existe uma **chave** associada a cada registro, usada para diferenciar os registros entre si.
  - Chave interna: chave está contida dentro do registro, em uma localização específica
  - Chave externa: essas chaves estão contidas em uma tabela de chaves separada, que inclui ponteiros/referências para os registros
  - Chave primária: para todo arquivo existe pelo menos um conjunto exclusivo de chaves
    - Dois registros não podem ter o mesmo valor de chave
  - Chave secundária: são as chaves não primárias
    - São chaves que não precisam ter valores exclusivos
    - Para que servem?

Termos Relacionados

- Algoritmo de busca
  - É um algoritmo que aceita um argumento **a** e tenta encontrar um registro na tabela cuja chave seja **a**

Termos Relacionados

- Operações na tabela
  - Inserção: adicionar um novo elemento à tabela
  - Remoção: retirar um elemento da tabela
  - Leitura: procurar um elemento na tabela e, se achá-lo, torná-lo disponível para uso

Tipos de Busca

- A tabela pode ser:
  - Um vetor de registros
  - Uma lista encadeada
  - Uma árvore
  - ...
- A tabela pode ficar:
  - Totalmente na memória (*busca interna*)
  - Totalmente no armazenamento auxiliar (busca externa)
  - Dividida entre ambos

Tipos de Busca

- Algumas **técnicas de busca em memória interna** são
  - Busca Sequencial
  - Busca Binária
  - Busca por Interpolação
  - Busca em Árvores
  - Hashing

- O objetivo é encontrar um dado registro com o **menor custo**
  - Cada técnica possui suas vantagens e desvantagens

Busca Sequencial

- A busca sequencial é a forma mais simples de busca
  - É aplicável a uma tabela organizada como um vetor ou como uma lista encadeada

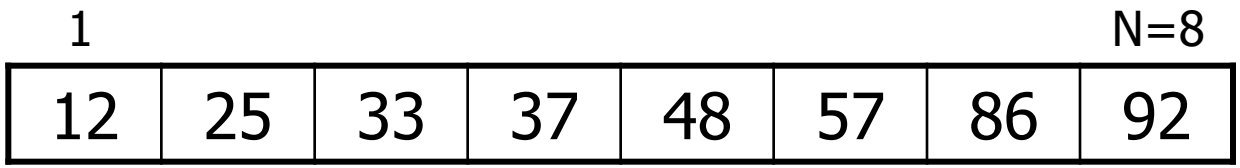
Busca Sequencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

Busca Sequencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

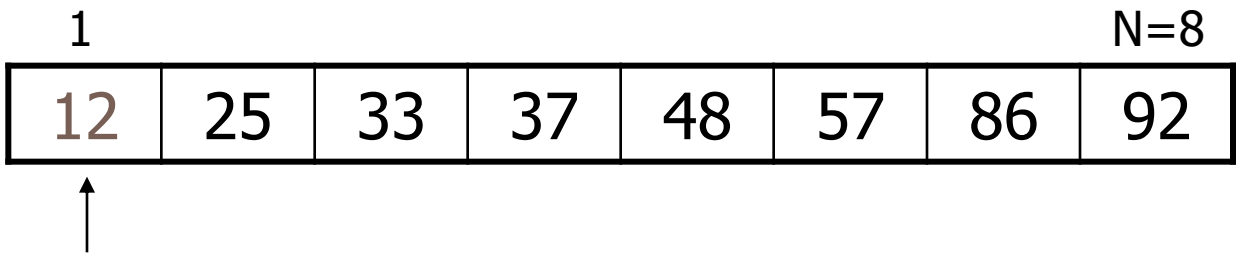
Procure por 48



Busca Sequencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

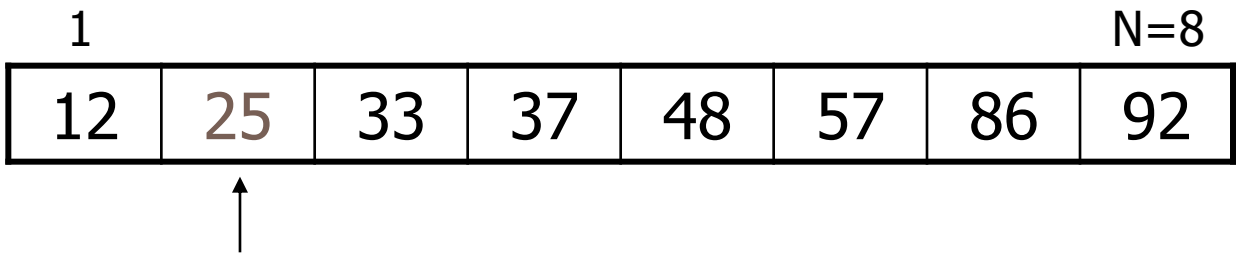
Procure por 48



Busca Sequencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

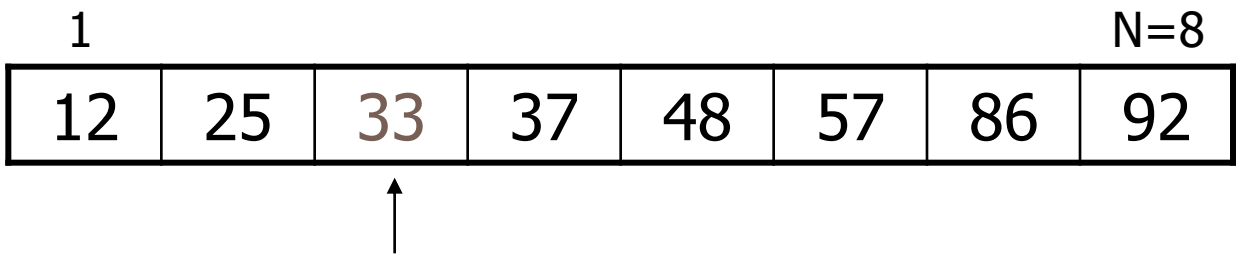
Procure por 48



Busca Sequencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

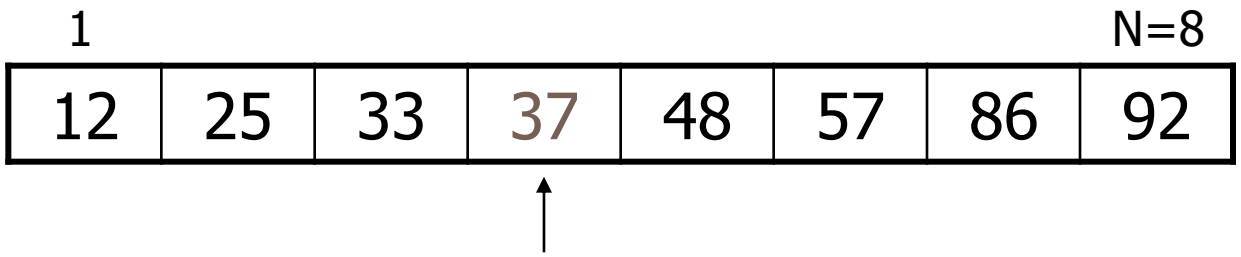
Procure por 48



Busca Sequencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

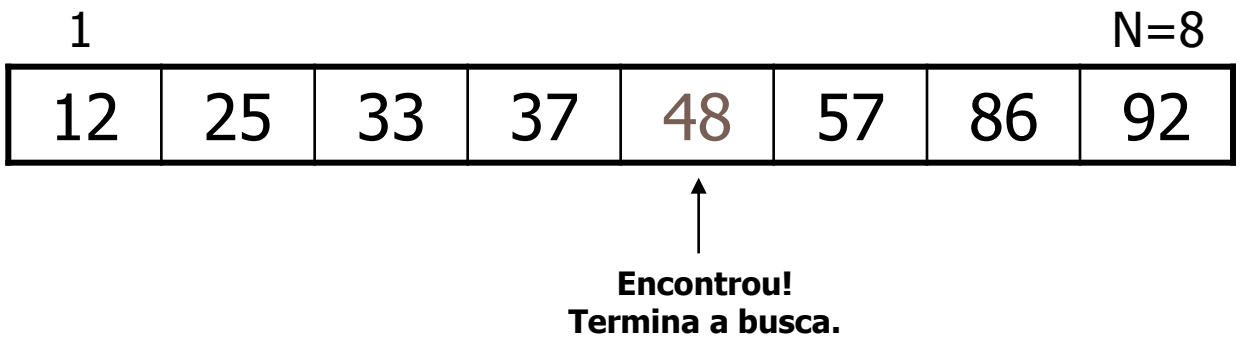
Procure por 48



Busca Sequencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

Procure por 48



Busca Sequencial

- Implementação
  - Algoritmo de busca sequencial em um vetor v de n posições (0 até n-1), sendo x a chave procurada

```
int busca_sequencial(int v[], int n, int x) {
   int i;
   for (i=0; i<n; i++)
      if (v[i]==x)
            return i;
   return -1;
}
```

Busca Sequencial

- Uma maneira de tornar o algoritmo mais eficiente é usar um elemento do tipo **sentinela**
  - **Sentinela**: elemento de valor x (o buscado) no final da tabela
  - Qual a vantagem de se usar um nó sentinela?

Busca Sequencial

- Uma maneira de tornar o algoritmo mais eficiente é usar um elemento do tipo **sentinela**
  - **Sentinela**: elemento de valor x (o buscado) no final da tabela
  - O sentinela garante que o elemento será encontrado, o que elimina um teste, melhorando a performance do algoritmo

Busca Sequencial

- Implementação
  - Busca sequencial com sentinela (vetor deve ser declarado com uma posição a mais)

```
int busca_sequencial_com_sentinela(int v[], int n, int x) {
      int i;
      v[n] = x;
      for(i=0; x!=v[i]; i++)
               ;
      if (i < n)
               return i;
      else
               return -1;
}
```

Busca Sequencial

- Limitações do vetor
  - Tamanho fixo
    - Pode desperdiçar ou faltar espaço
- Alternativa
  - Lista dinâmica e encadeada
    - O que muda na busca sequencial?

Busca Sequencial

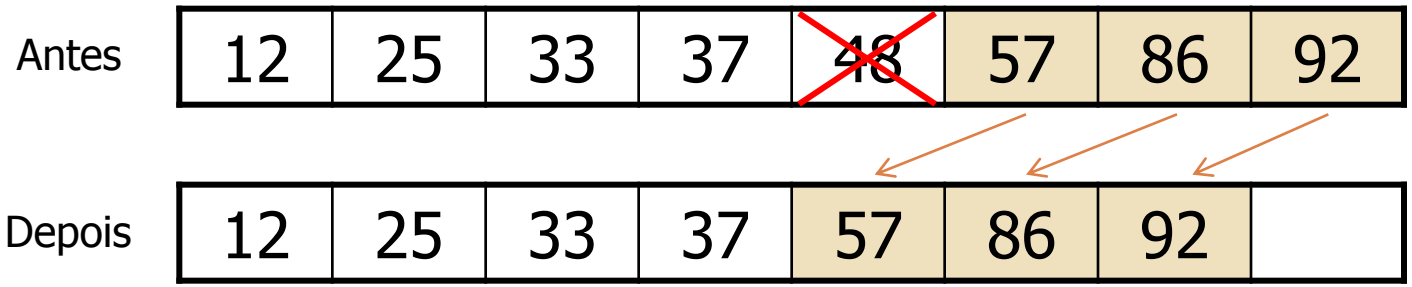
- Exercício para casa
  - Escrever em C uma função de busca de um elemento em uma lista encadeada alocada dinamicamente

Busca Sequencial

- Complexidade da busca sequencial
  - Se o registro procurado for o primeiro: 1 comparação
  - Se o registro procurado for o último: N comparações
  - Se for igualmente provável que o argumento apareça em qualquer posição da tabela, teremos, em média: N/2 comparações
  - Se a busca for mal sucedida: N comparações
  - Logo, a busca sequencial, no pior caso e na média, é O(n)

Busca Sequencial

- Para um vetor não ordenado
  - Inserção no final
  - Remoção implica movimento dos registros posicionados após o registro removido



Busca Sequencial

- Para aumentar a eficiência
  - Rearranjar continuamente a tabela de modo que os registros mais acessados sejam deslocados para o início
    1. Método mover-para-frente: sempre que uma pesquisa obtiver êxito, o registro recuperado é colocado no início da lista
    2. Método da transposição: um registro recuperado com sucesso é trocado com o registro imediatamente anterior
  - Ambos baseiam-se no fenômeno observado chamado recuperação recorrente de registros

Busca Sequencial

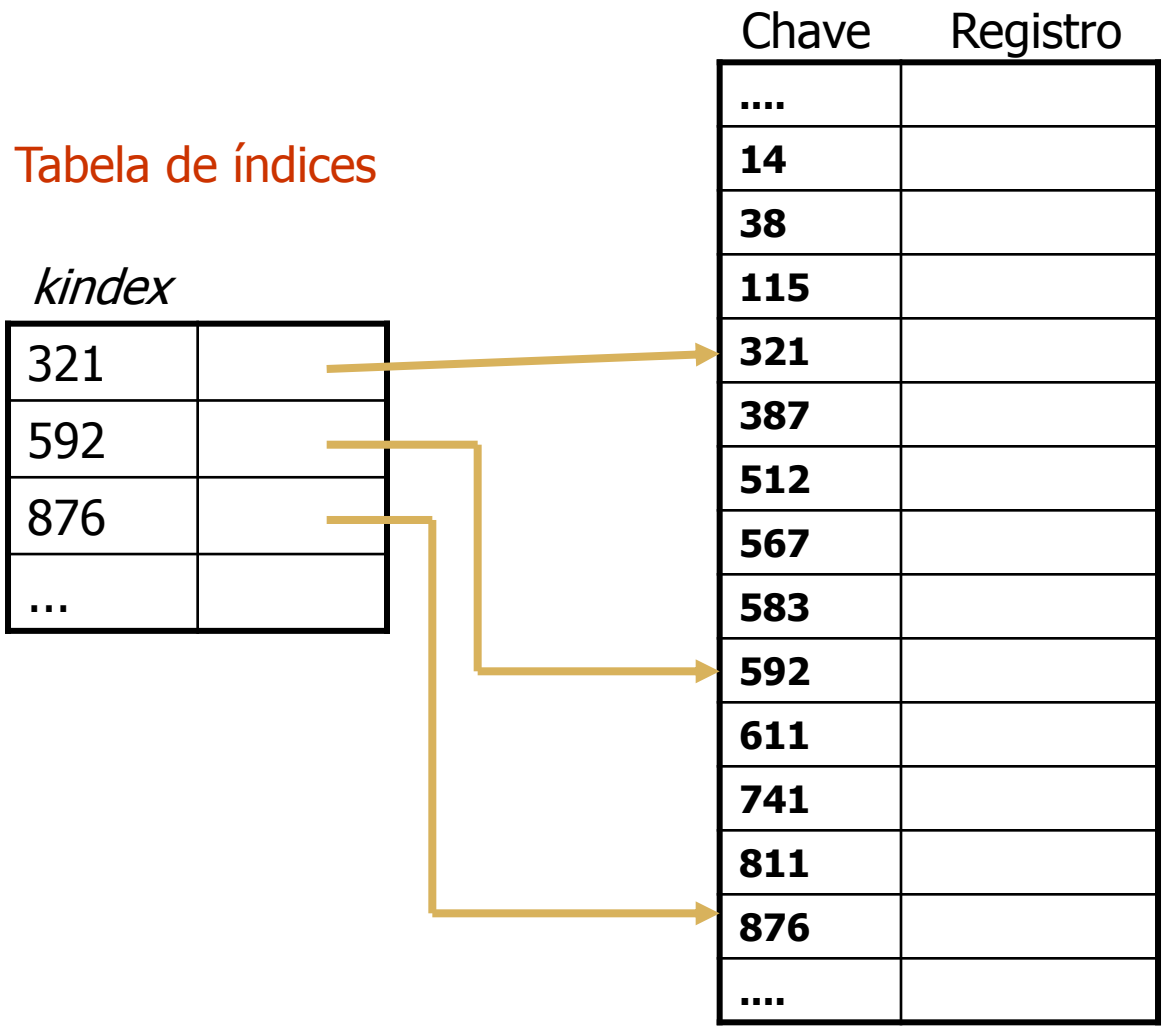
- Busca sequencial em tabela ordenada
  - A eficiência da operação de busca melhora se as chaves dos registros estiverem ordenadas
    - Se a busca for mal sucedida, serão realizadas na média N/2 comparações, pois interrompe-se a busca assim que uma chave maior do que a procurada é encontrada
  - Desvantagem do método?

Busca Sequencial

Busca sequencial indexada

- Existe uma tabela auxiliar, chamada tabela de índices, além do próprio arranjo ordenado
- Cada elemento na tabela de índices contém uma chave (*kindex*) e um indicador do registro no arquivo que corresponde a *kindex*
  - Faz-se a busca a partir do ponto indicado no índice, sendo que a busca não precisa ser feita desde o começo
- Pode ser implementada como uma lista encadeada ou como um vetor
  - O indicador da posição na tabela pode ser um ponteiro ou uma variável inteira, respectivamente

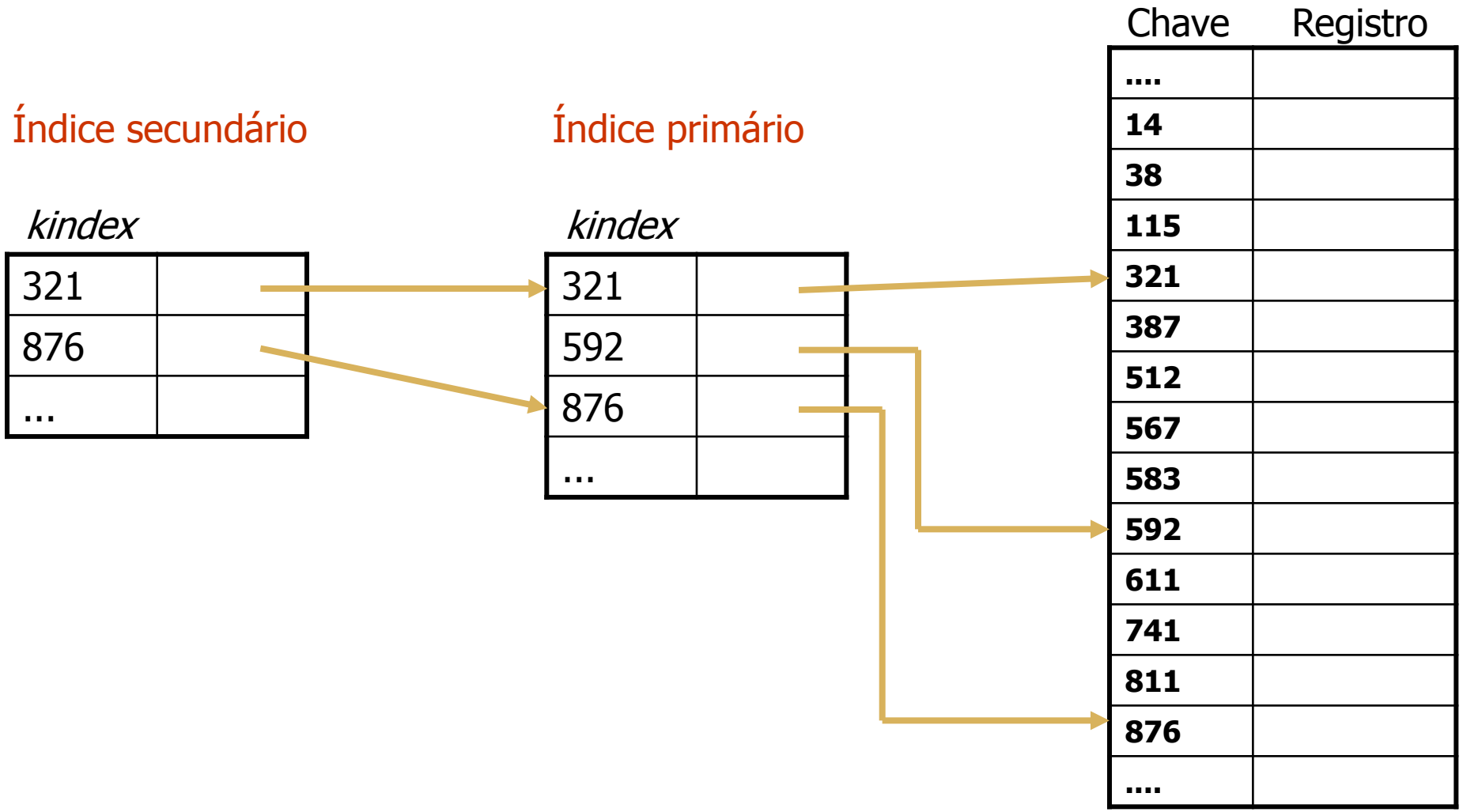
Busca Sequencial



Busca Sequencial

- Busca sequencial indexada
  - Se a tabela for muito grande, pode-se ainda usar uma **tabela de índices secundária**
    - O índice secundário é um índice para o índice primário

Busca Sequencial



Busca Sequencial

- Vantagem
  - Não precisa acessar todos os registros anteriores até encontrar o desejado
    - O tempo de busca diminui consideravelmente
- Desvantagens
  - A tabela e os índices têm que estar ordenados
  - Exige espaço adicional para armazenar o(s) índice(s)
  - Algo mais?

Busca Sequencial

- Vantagem
  - Não precisa acessar todos os registros anteriores até encontrar o desejado
    - O tempo de busca diminui consideravelmente
- Desvantagens
  - A tabela e os índices têm que estar ordenados
  - Exige espaço adicional para armazenar o(s) índice(s)
  - Cuidados com inserção e remoção

Busca Sequencial

- Remoção

1. Remove-se o elemento e rearranja-se a tabela inteira e o(s) índice(s)

ou

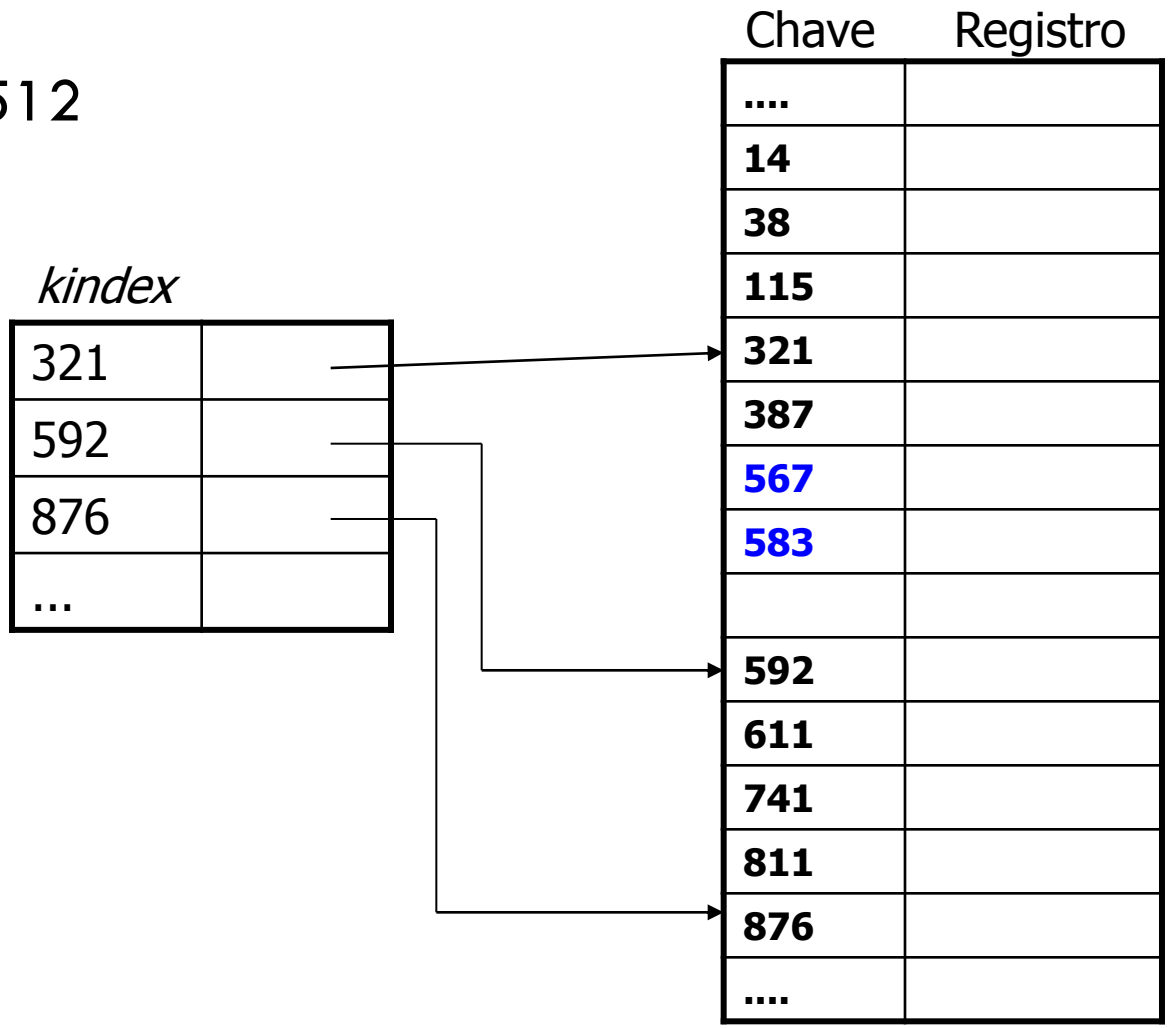
2. Marca-se a posição do elemento removido, indicando que ela pode ser ocupada por um outro elemento futuramente

Busca Sequencial

- Inserção
  - Se houver espaço vago na tabela, rearranjam-se os elementos localmente.
    - Por que localmente?
  - Se não houver espaço vago, rearranjar a tabela a partir do ponto apropriado, e reconstruir o(s) índice(s)

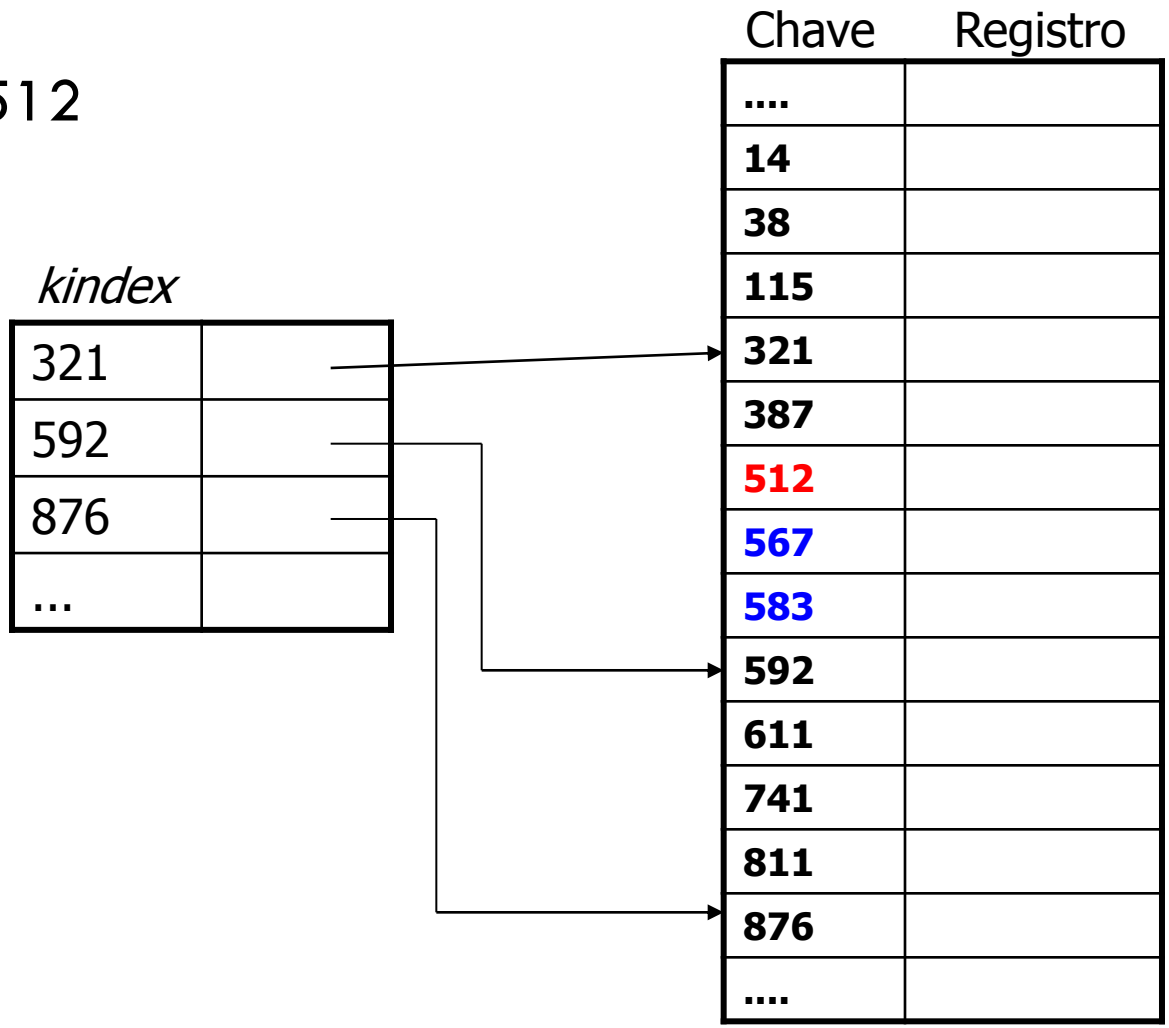
Busca Sequencial

- Inserção do elemento 512 com espaço vago
  - 567 e 583 descem
  - 512 é inserido



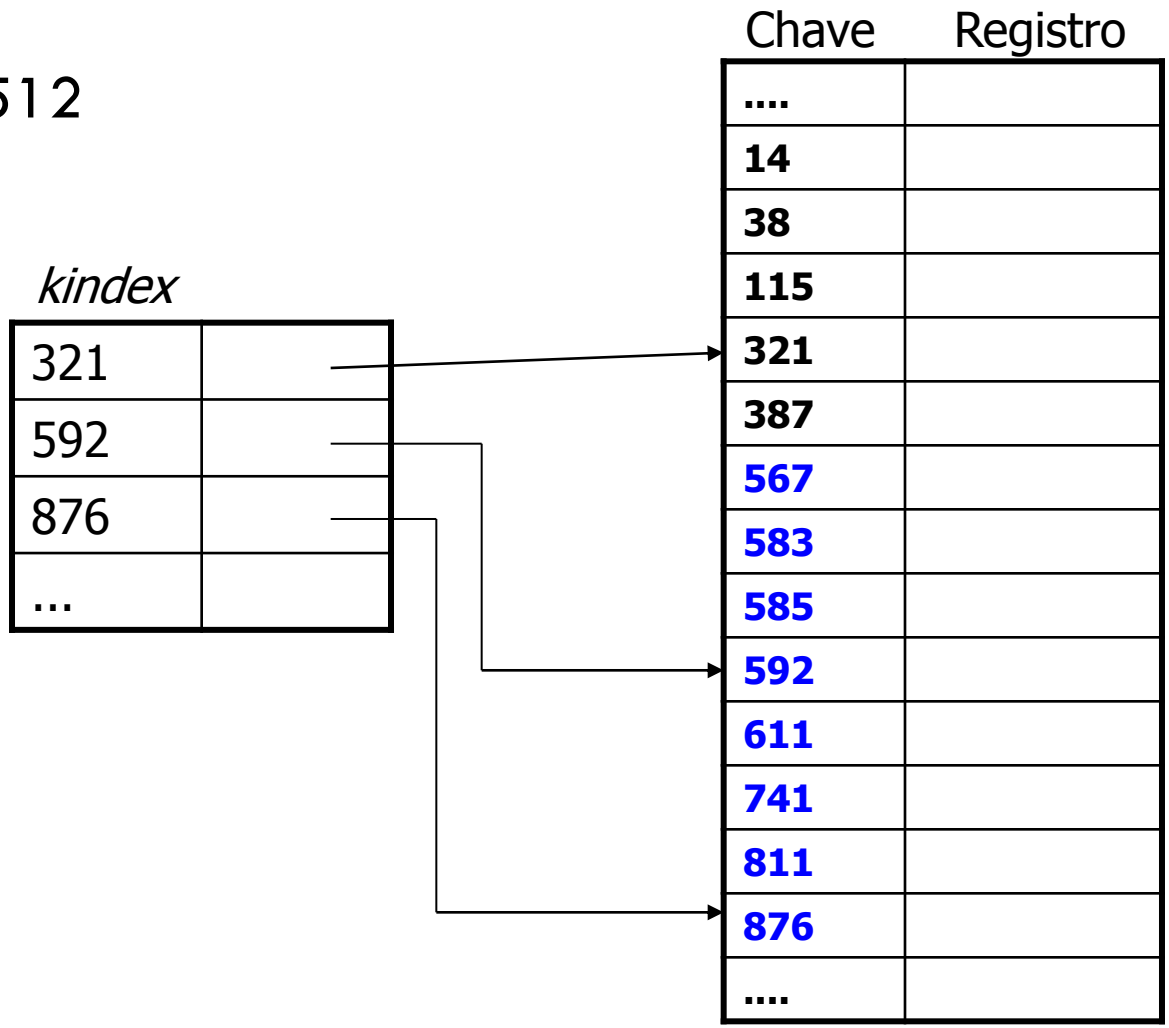
Busca Sequencial

- Inserção do elemento 512 com espaço vago
  - 567 e 583 descem
  - 512 é inserido



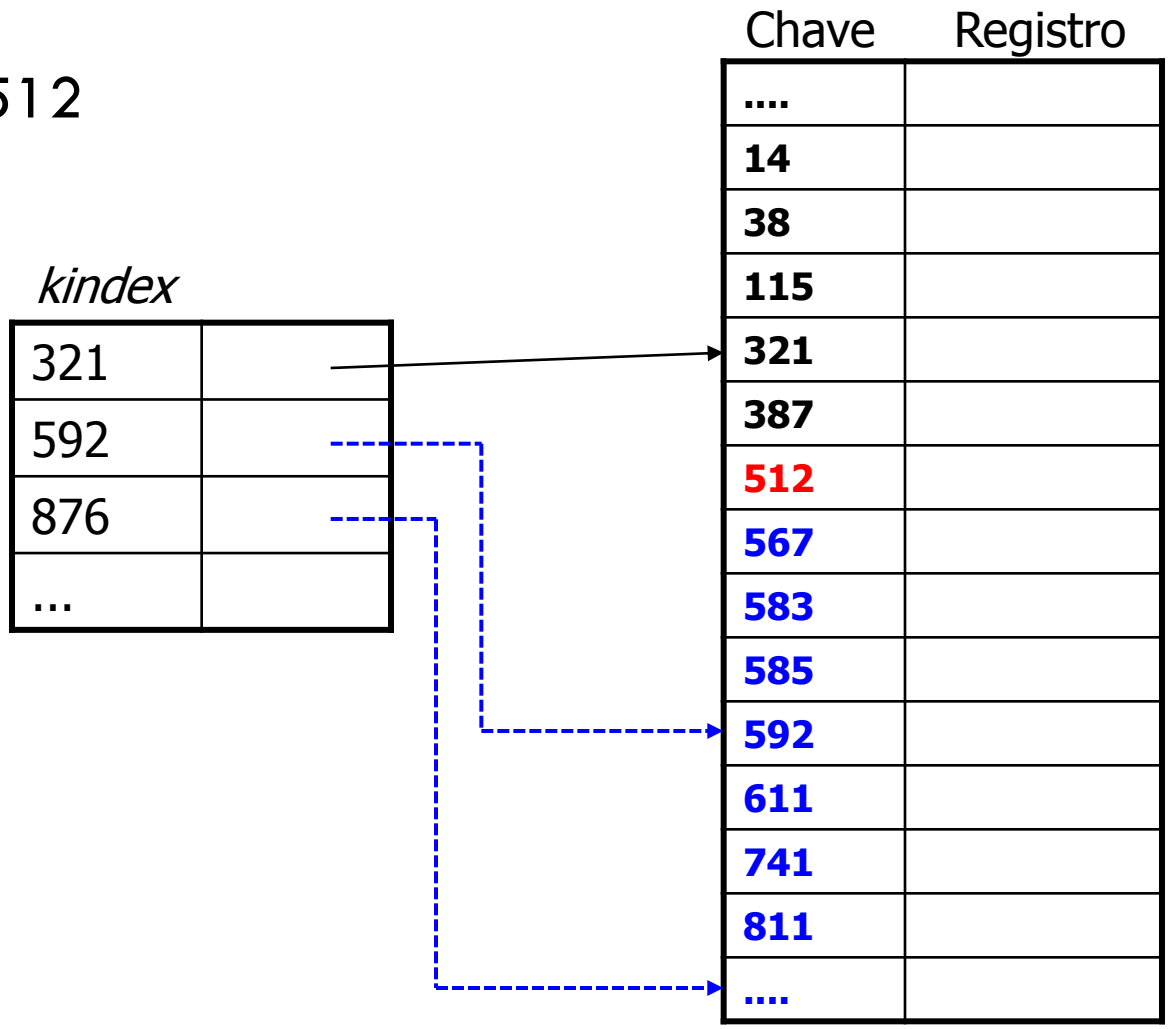
Busca Sequencial

- Inserção do elemento 512 sem espaço vago
  - Elementos a partir de 567 descem
  - 512 é inserido
  - Índice é re-construído



Busca Sequencial

- Inserção do elemento 512 sem espaço vago
  - Elementos a partir de 567 descem
  - 512 é inserido
  - Índice é re-construído



Busca Sequencial

- Como montar o índice primário
  - Se a tabela não estiver ordenada, ordene-a
  - Divide-se o número de elementos da tabela pelo tamanho do índice (ni) desejado: n/ni
  - Para montar o índice, recuperam-se da tabela os elementos

    0, 0+n/ni, 0+2*n/ni, ...
  - Cada par de elementos do índice representa n/ni elementos da tabela

Busca Sequencial

- Exemplo
  - Divide-se o número de elementos da tabela pelo tamanho do índice desejado
    - Se a tabela tem 1.000 elementos e deseja-se um índice primário de 10 elementos, faz-se 1.000/10=100
  - Para montar o índice, recuperam-se da tabela os elementos 0, 0+n/ni, 0+2*n/ni, ...
    - O índice primário é montado com os elementos das posições 0, 100, 200, etc. da tabela
  - Cada par de elementos do índice representa n/ni elementos da tabela
    - Cada par fornece os limites de um grupo de 100 elementos da tabela

Busca Sequencial


```
typedef struct {
   int kindex, pos;
} idx;

void cria_indice(idx index[], int ni, int v[], int n) {
   int pos, i=0;
   while (i < ni) {
        pos = i * n / ni;
        index[i].kindex = v[pos];
        index[i].pos = pos;
        i++;
   }
}
```

Busca Sequencial

- Para montar um índice secundário, aplica-se raciocínio análogo sobre o índice primário

- Em geral, não são necessários mais do que 2 índices

Busca Sequencial

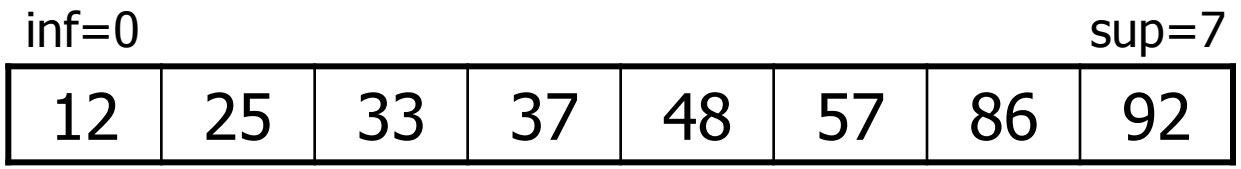
- Exercício
  - Escrever em C uma sub-rotina de busca sequencial indexada em um vetor com índice primário

Busca Binária

- O elemento buscado é comparado ao elemento do meio do arranjo (já ordenado)
  - Se igual, busca bem-sucedida
  - Se menor, busca-se na metade inferior do arranjo
  - Se maior, busca-se na metade superior do arranjo

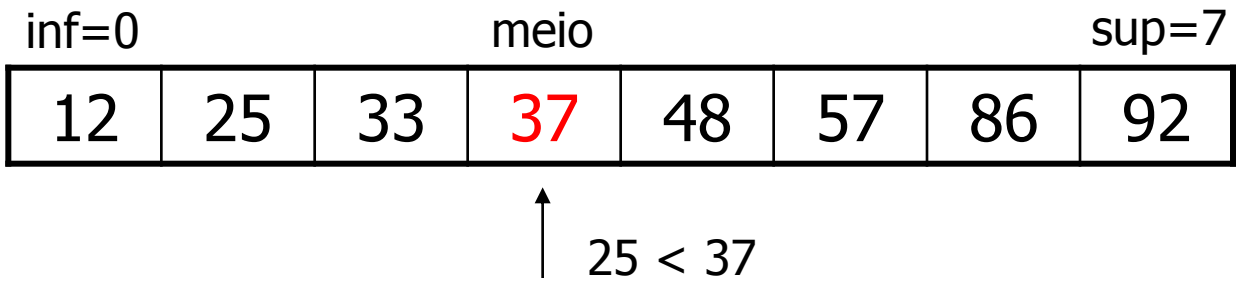
Busca Binária

- Busca 25



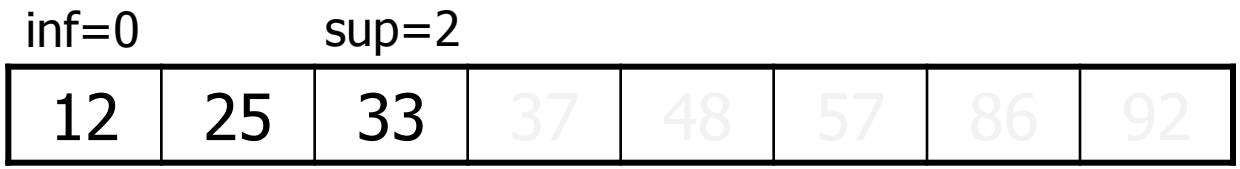
Busca Binária

- Busca 25



Busca Binária

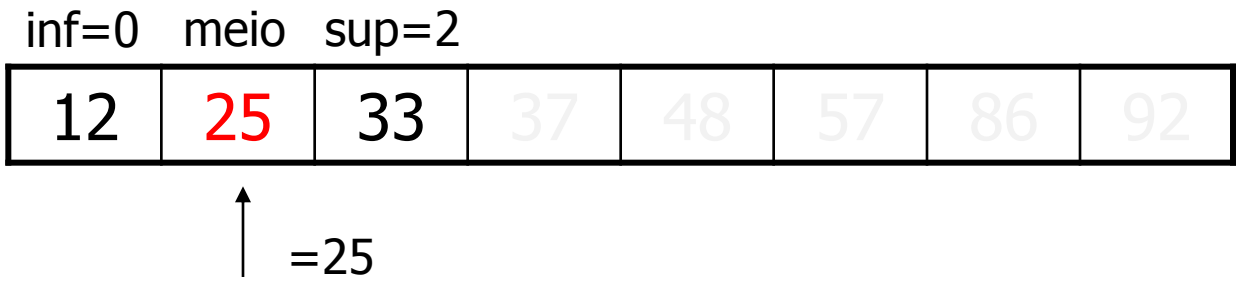
- Busca 25



Em cada passo, o tamanho do arranjo em que se busca é dividido por 2

Busca Binária

- Busca 25



Busca Binária

Versão recursiva:

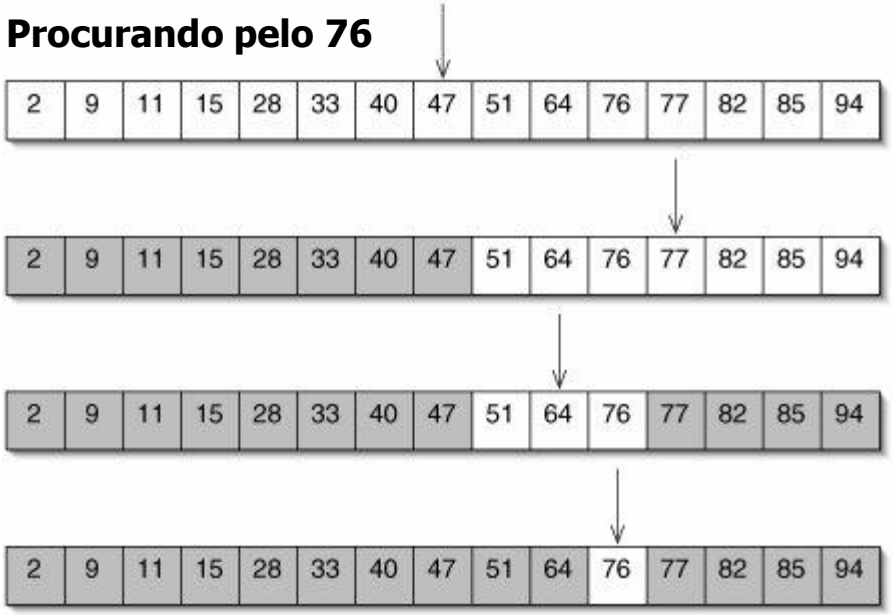
```
int busca_binaria_rec(int v[], int inf, int sup, int x) {
   int meio;
   if (inf <= sup) {
        meio = (inf + sup) / 2;
        if (x == v[meio])
                return 1;
        else if (x < v[meio])
                return busca_binaria_rec(v, inf, meio-1, x);
        else
                return busca_binaria_rec(v, meio+1, sup, x);
   } else
        return 0;
}
```

Busca Binária

- Exercício
  - Implemente a versão não recursiva

Busca Binária

- Complexidade
  - $O(\log_2 n)$ no pior caso, pois cada comparação reduz o número de possíveis candidatos por um fator de 2



Busca Binária

- Vantagens
  - Eficiência da busca
  - Simplicidade da implementação

- Desvantagens
  - Nem todo arranjo está ordenado
  - Exige o uso de um vetor para armazenar os dados
  - Inserção e remoção de elementos são ineficientes
    - Realocação de elementos

Busca Binária

- A busca binária pode ser aplicada em uma tabela indexada
  - Em vez de pesquisar o índice sequencialmente, pode-se usar uma busca binária

Busca por Interpolação

- Se as chaves estiverem uniformemente distribuídas (além de ordenadas), esse método pode ser ainda mais eficiente do que a busca binária

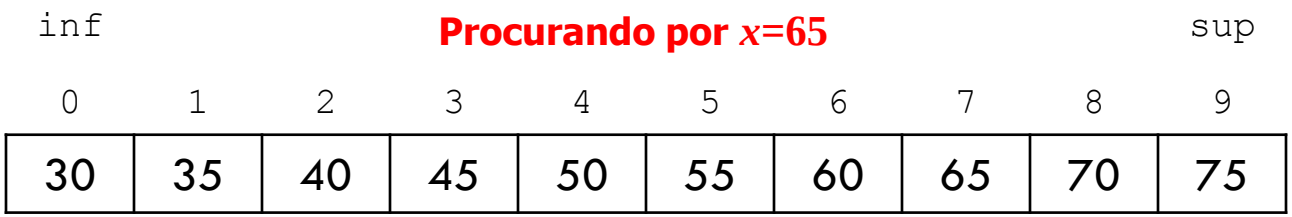
- Com chaves uniformemente distribuídas, pode-se esperar que x esteja aproximadamente na posição

$$meio = \inf + (\sup - \inf)\left[\frac{x - v[\inf]}{v[\sup] - v[\inf]}\right]$$

- O algoritmo é o mesmo da busca binária, muda-se apenas o cálculo do "meio"

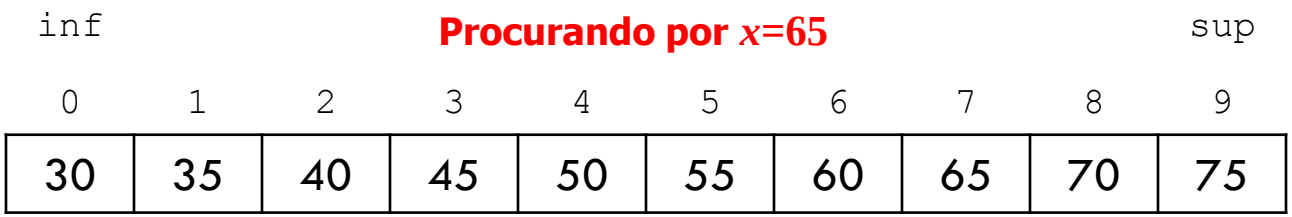
Busca por Interpolação

$$meio = \inf + (\sup - \inf)\left[\frac{x - v[\inf]}{v[\sup] - v[\inf]}\right]$$



Busca por Interpolação

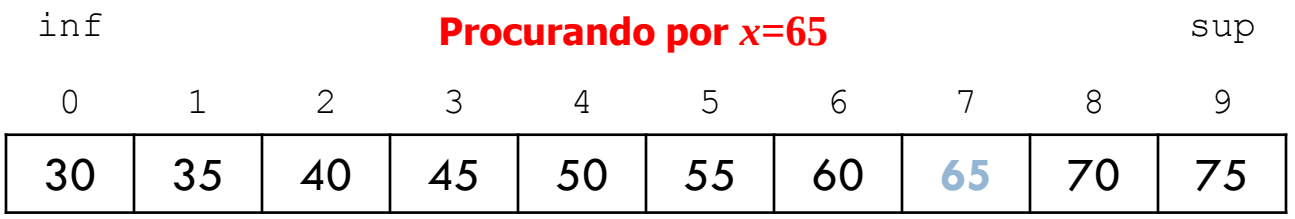
$$meio = \inf + (\sup - \inf)\left[\frac{x - v[\inf]}{v[\sup] - v[\inf]}\right]$$



$$meio = 0 + (9 - 0)\left[\frac{65 - 30}{75 - 30}\right] = 9\left[\frac{35}{45}\right]$$

Busca por Interpolação

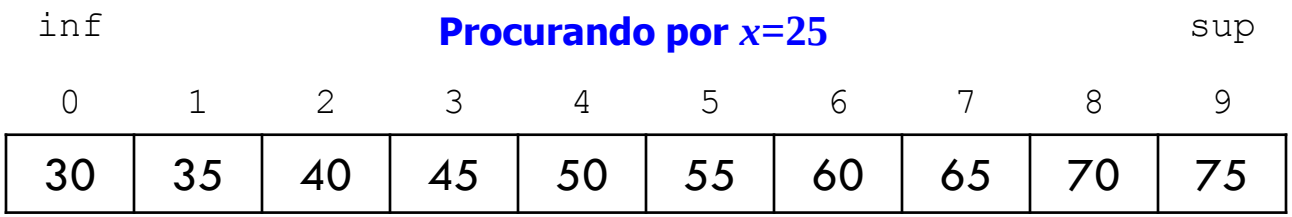
$$meio = \inf + (\sup - \inf)\left[\frac{x - v[\inf]}{v[\sup] - v[\inf]}\right]$$



$$meio = 0 + (9 - 0)\left[\frac{65 - 30}{75 - 30}\right] = 9\left[\frac{35}{45}\right] = 7$$

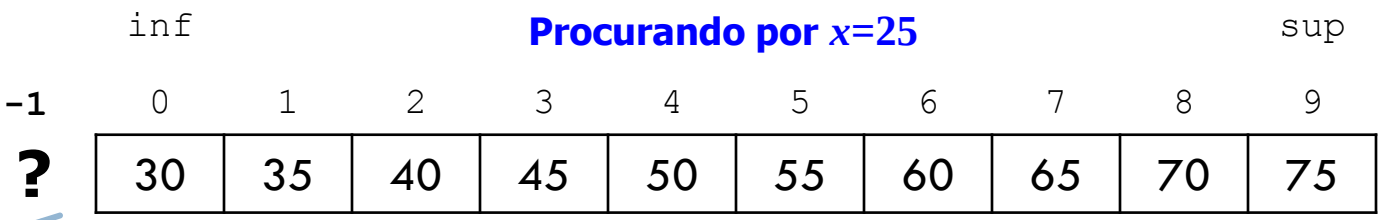
Busca por Interpolação

$$meio = \inf + (\sup - \inf)\left[\frac{x - v[\inf]}{v[\sup] - v[\inf]}\right]$$



Busca por Interpolação

$$meio = \inf + (\sup - \inf)\left[\frac{x - v[\inf]}{v[\sup] - v[\inf]}\right]$$



$$meio = 0 + (9 - 0)\left[\frac{25 - 30}{75 - 30}\right] = -9\left[\frac{5}{45}\right] = -1$$

Busca por Interpolação

- Complexidade
  - Na média $O(\log_2 (\log_2 n))$, se as chaves estiverem distribuídas de maneira aproximadamente uniforme
  - Se as chaves não estiverem uniformemente distribuídas, a busca por interpolação tem um pior caso $O(n)$

Busca por Interpolação

- Em situações práticas, as chaves tendem a se aglomerar em torno de determinados valores, não sendo, portanto, uniformemente distribuídas
  - Exemplo: há uma quantidade maior de nomes começando com “S” do que com “Q”
  - Utilizar a busca por interpolação somente quando houver garantia de uma distribuição de chaves próxima da uniforme

Créditos

Aula baseada nos materiais dos profs.
Rudinei Goularte e Thiago A. S. Pardo