Técnicas de Busca e Ordenação (TBO)

Tabelas de Símbolos

Departamento de Informática (DI)
Centro Tecnológico (CT)
Universidade Federal do Espírito Santo (UFES)

(Material baseado nos slides do Professor Eduardo Zambon)

Introdução

- A partir de agora vamos focar no outro ponto fundamental do curso: busca.
- Podemos abstrair boa parte das aplicações de busca através do conceito de tabela de símbolos.
- **Aula de hoje**: apresentação da estrutura abstrata tabela de símbolos e suas implementações elementares.
- **Objetivos**: compreender as aplicações e o funcionamento de uma tabela de símbolos.

Referências

Chapter 12 – Symbol Tables and Binary Search Trees
R. Sedgewick

Parte I

Tabelas de Símbolos

Tabelas de símbolos

Abstração de um par chave-valor (*key-value*).

Operações fundamentais:

- Inserir um valor com a chave especificada.
- Dada uma chave, buscar pelo valor correspondente.

Ex.: *DNS lookup*.

- Inserir o nome de domínio com um endereço IP.
- Dado o domínio, encontrar o IP correspondente.

domain name	IP address
www.cs.princeton.edu	128.112.136.11
www.princeton.edu	128.112.128.15
www.yale.edu	130.132.143.21
www.harvard.edu	128.103.060.55
www.simpsons.com	209.052.165.60
↑ key	↑ value

Tabelas de símbolos: aplicações

application	purpose of search	key	value
dictionary	find definition	word	definition
book index	find relevant pages	term	list of page numbers
file share	find song to download	name of song	computer ID
financial account	process transactions	account number	transaction details
web search	find relevant web pages	keyword	list of page names
compiler	find properties of variables	variable name	type and value
routing table	route Internet packets	destination	best route
DNS	find IP address	domain name	IP address
reverse DNS	find domain name	IP address	domain name
genomics	find markers	DNA string	known positions
file system	find file on disk	filename	location on disk

Tabelas de símbolos: contexto

Também conhecidas como: mapas, dicionários, *arrays* associativos.

Generalizam os *arrays*: chaves não precisam estar entre 0 e $N - 1$.

Suporte das linguagens:

- Bibliotecas externas: C, Visual Basic, ML, bash, ...
- Bibliotecas *built-in*: Java, C#, C++, Scala, ...
- Suporte nativo: Awk, Perl, PHP, Tcl, JavaScript, Python, Ruby, Lua, ...

- Em PHP, todo *array* é associativo.
- Em JavaScript, todo objeto é um *array* associativo.
- Em Lua, tabela é a única estrutura de dados primitiva.

Veja os vídeos do Brian Kernighan sobre *arrays* associativos.

Tabelas de símbolos: API básica

```
// Create an empty symbol table.
void ST_init(int maxN);

// Put key-value pair into the table: a[key] = val; .
void ST_put(Key key, Value val);

// Value paired with key: a[key] .
Value ST_get(Key key);

// Is there a value paired with key?
bool ST_contains(Key key);

// Remove key (and its value) from table.
void ST_delete(Key key);

// Is the table empty?
bool ST_empty();

// Number of key-value pairs in the table.
int ST_size();

// Clean up the table memory.
void ST_finish();
```

Tabelas de símbolos: convenções

Convenções sobre chaves e valores:

- Tipos `Key` e `Value` possuem valores nulos especiais: `NULL_Key` e `NULL_Value`, respectivamente.
- Usuário não deve inserir valores nulos.
- Função `ST_get()` retorna `NULL_Value` se a chave não estiver presente.
- Funções de busca de chaves (`ST_min()`, adiante) retorna `NULL_Key` se não houver uma chave adequada na tabela.
- Função `ST_put()` escreve o valor novo sobre o antigo.
- Fácil de implementar `contains()`:

```
bool ST_contains(Key key)
{ return ST_get(key) != NULL_Value; }
```

- Permite implementar versão preguiçosa de `delete()`:

```
void ST_delete(Key key)
{ ST_put(key, NULL_Value); }
```

Um programa cliente simples

Construir uma tabela de símbolos associando o valor *i* com o *i*-ésimo caractere da entrada.

```
void visit(Key key, Value val) {
    printf("%c %i\n", key, val);
}
int main(void) {
    ST_init(N);
    for (int i = 0; i < N; i++) {
        char key;
        scanf("%c ", &key);
        ST_put(key, i);
    }
    ST_traverse(visit);
    ST_finish();
}
```

output

```
A  8
C  4
E  12
H  5
L  11
M  9
P  10
R  3
S  0
X  7
```

keys	S	E	A	R	C	H	E	X	A	M	P	L	E
values	0	1	2	3	4	5	6	7	8	9	10	11	12

Função `ST_traverse` caminha em ordem pelas chaves.

Parte II

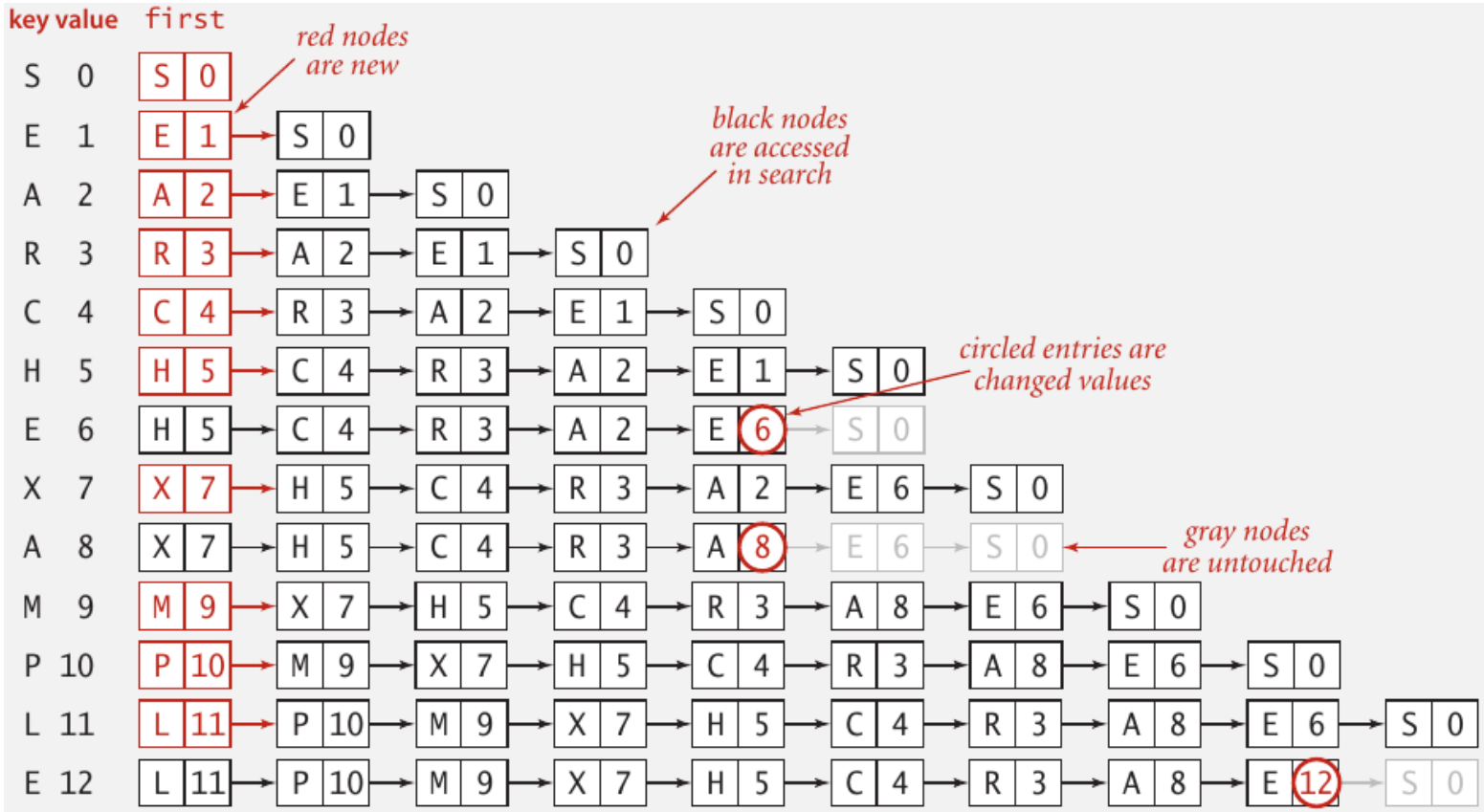
Implementações Elementares

Tabela de símbolos como uma lista encadeada

Estrutura: mantém uma lista encadeada não ordenada de pares chave-valor.

Busca: Varre todas as chaves até encontrar.

Inserção: Faz uma busca, se não encontrar insere no começo.



Implementações elementares: sumário

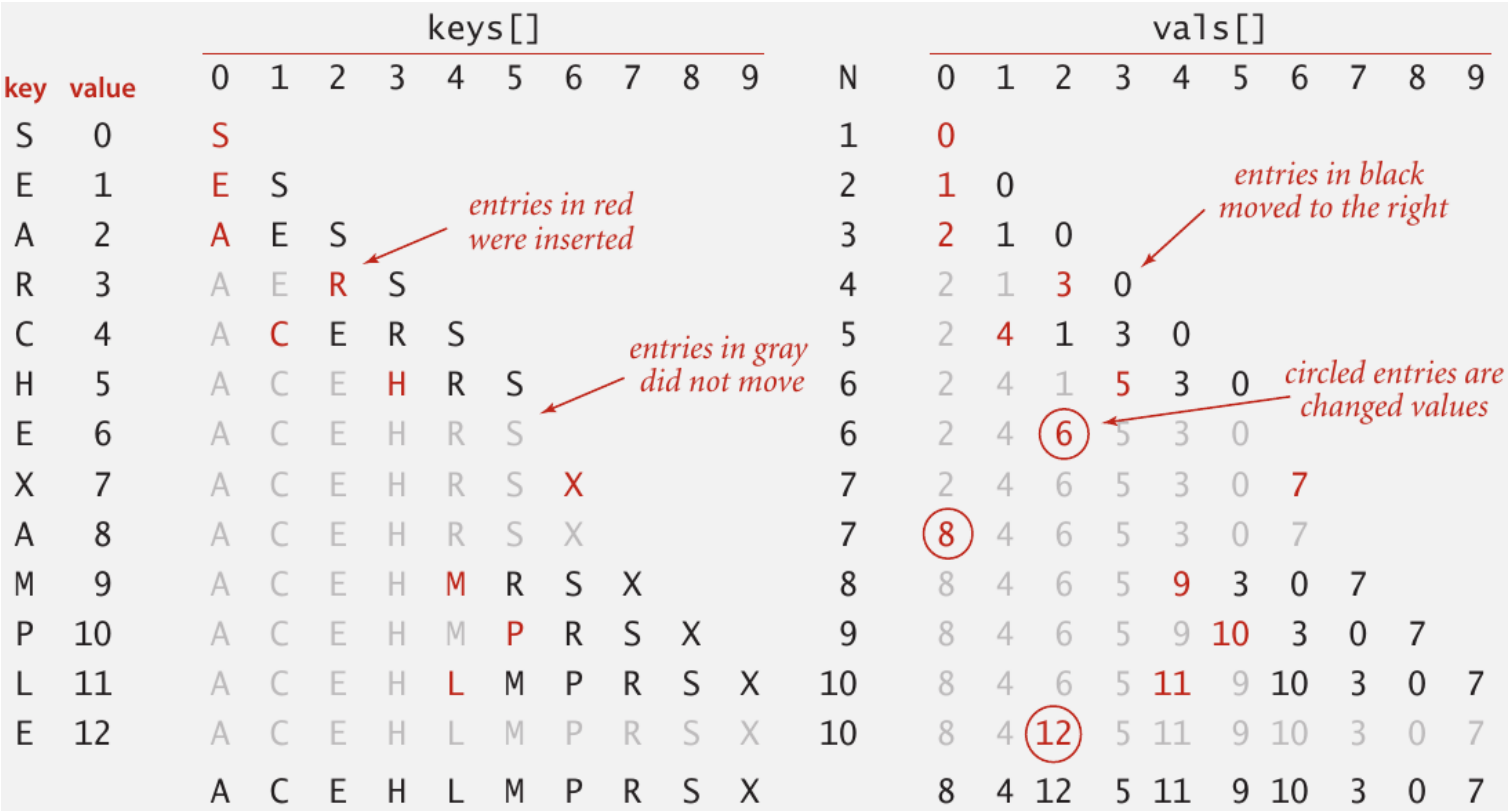
ST implementation	guarantee		average case	
	search	insert	search hit	insert
sequential search (unordered list)	N	N	$N / 2$	N

Desafio: Implementações eficientes para ambas busca e inserção.

Tabela de símbolos como um *array* ordenado

Estrutura: mantém um *array* ordenado de pares chave-valor. (Ou dois *arrays* separados).

Problema: Para inserir, precisa mover todas as chaves maiores.



key	value	keys[0]	keys[1]	keys[2]	keys[3]	keys[4]	keys[5]	keys[6]	keys[7]	keys[8]	keys[9]	N	vals[0]	vals[1]	vals[2]	vals[3]	vals[4]	vals[5]	vals[6]	vals[7]	vals[8]	vals[9]
S	0	S										1	0									
E	1	E	S									2	1	0								
A	2	A	E	S								3	2	1	0							
R	3	A	E	R	S							4	2	1	3	0						
C	4	A	C	E	R	S						5	2	4	1	3	0					
H	5	A	C	E	H	R	S					6	2	4	1	5	3	0				
E	6	A	C	E	H	R	S					6	2	4	6	5	3	0				
X	7	A	C	E	H	R	S	X				7	2	4	6	5	3	0	7			
A	8	A	C	E	H	R	S	X				7	8	4	6	5	3	0	7			
M	9	A	C	E	H	M	R	S	X			8	8	4	6	5	9	3	0	7		
P	10	A	C	E	H	M	P	R	S	X		9	8	4	6	5	9	10	3	0	7	
L	11	A	C	E	H	L	M	P	R	S	X	10	8	4	6	5	11	9	10	3	0	7
E	12	A	C	E	H	L	M	P	R	S	X	10	8	4	12	5	11	9	10	3	0	7
		A	C	E	H	L	M	P	R	S	X		8	4	12	5	11	9	10	3	0	7

Vantagem: procura por uma chave pode usar busca binária.

Implementações elementares: sumário

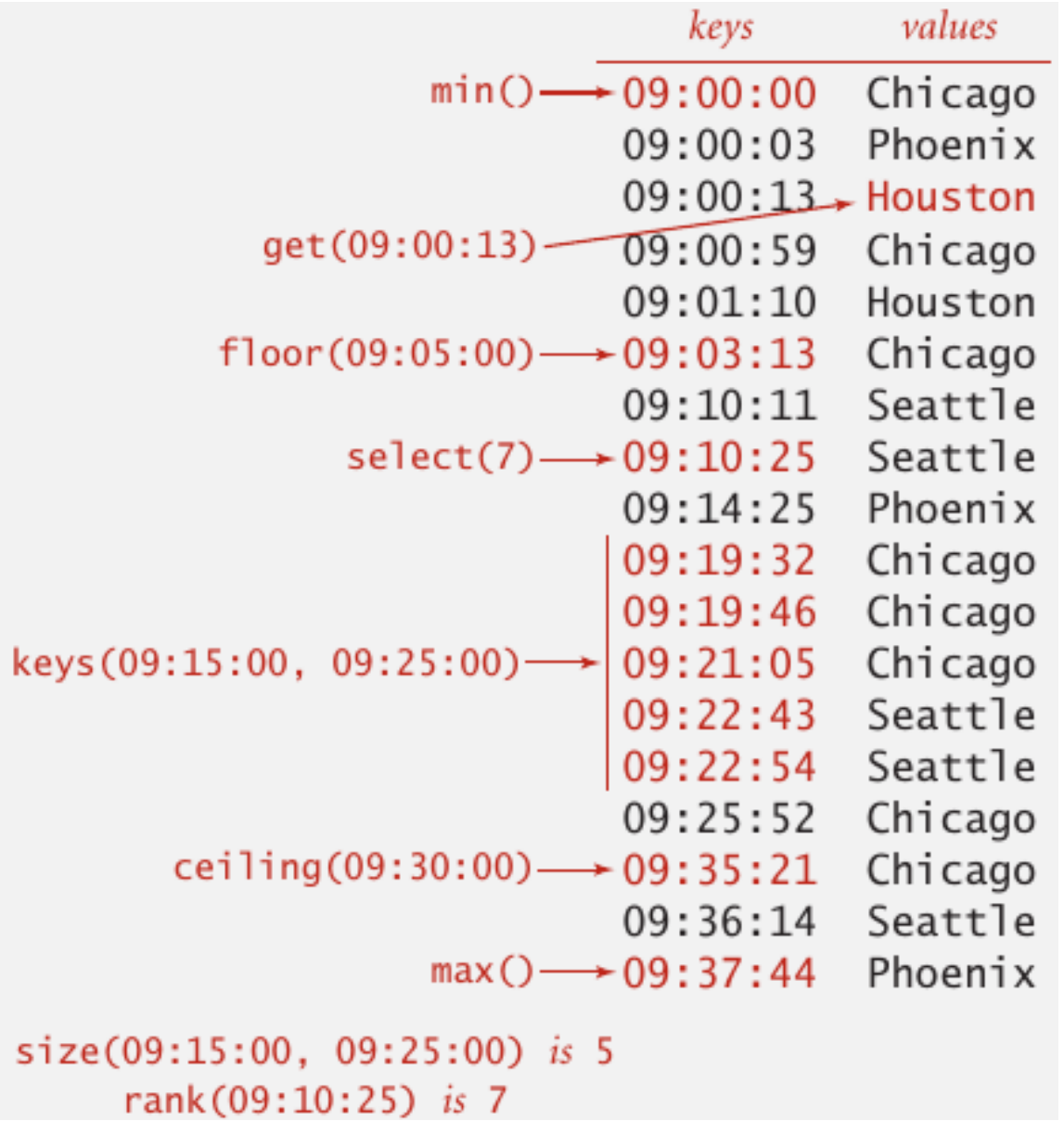
ST implementation	guarantee		average case	
	search	insert	search hit	insert
sequential search (unordered list)	N	N	$N / 2$	N
binary search (ordered array)	$\log N$	N	$\log N$	$N / 2$

Desafio: Implementações eficientes para ambas busca e inserção.

Parte III

Operações Ordenadas

Exemplo de API de operações ordenadas



Tabelas de símbolos: API de operações ordenadas

```
// Smallest key.
Key ST_min();

// Largest key.
Key ST_max();

// Largest key less than or equal to key.
Key ST_floor(Key key);

// Smallest key greater than to equal to key.
Key ST_ceiling(Key key);

// Number of keys less than key.
int ST_rank(Key key);

// Delete smallest key.
void ST_delmin();

// Delete largest key.
void ST_delmax();

// Visit all the key-value pairs in the order of their keys.
void ST_traverse(void (*visit)(Key,Value));
```

Implementações elementares: sumário

	sequential search	binary search
search	N	$\log N$
insert / delete	N	N
min / max	N	1
floor / ceiling	N	$\log N$
rank	N	$\log N$
select	N	1
ordered iteration	$N \log N$	N