Tabelas de símbolos de árvores binárias de busca

- Além do Sedgewick (**sempre** leiam o Sedgewick), veja
  - `http://www.ime.usp.br/~pf/mac0122-2002/aulas/symbol-table.html`
  - `http://www.ime.usp.br/~pf/algoritmos/aulas/bint.html`
  - `http://www.ime.usp.br/~pf/algoritmos/aulas/binst.html`

Tabelas de símbolos de árvores binárias de busca

- *Tabelas de símbolos*: estruturas de dados para se manter um conjunto de itens que admitem as operações básicas de *inserção* de itens e *busca de um item com uma dada chave*.
- Implementação com *árvores binárias de busca* (ABBs)

Tabelas de símbolos

```
/* prog12.1.c - ST.h */
void STinit(int);
 int STcount();
void STinsert(Item);
Item STsearch(Key);
void STdelete(Item);
Item STselect(int);
void STsort(void (*visit)(Item));
```

Um cliente (levemente modificado); exemplo de execução

```
yoshi@erdos:~/IME/www/2003ii/mac122/exx/ST_basico$ a.out
We hold these truths to be self-evident, that all men are
created equal, that they are endowed by their Creator with
certain unalienable Rights, that among these are Life,
Liberty and the Pursuit of Happiness.

Creator
Happiness.
Liberty
Life,
Pursuit
Rights,
We
all
among
and
are
be
by
```

Um cliente (levemente modificado); exemplo de execução

```
certain
created
endowed
equal,
hold
men
of
self-evident,
that
the
their
these
they
to
truths
unalienable
with

35 keys, 30 distinct keys
yoshi@erdos:~/IME/www/2003ii/mac122/exx/ST_basico$
```

Tabelas de símbolos com ABBs

- Existem duas variantes de ABBs: balanceadas (vários tipos) e não-balanceadas
- Nesta disciplina veremos apenas ABBs não-balanceadas (ABBs balanceadas ficam para MAC323)

Árvores

- Árvores binárias T: (*i*) Λ ("árvore vazia") ou (r, T_L, T_R), (*ii*) $\square$ [nó externo] ou (r, T_L, T_R)
- Árvores binárias de busca: árvores binárias cujos nós estão associados a chaves de forma que valha a "propriedade de árvore binária de busca", a saber, as chaves associadas aos nós na subárvore esquerda (resp., direita) de um nó arbitrário x são menores ou iguais (resp., maiores ou iguais) à chave do nó x.

Uma implementação de uma TS com uma ABB

```
/* prog12.7.c */
#include <stdlib.h>
#include "Item.h"

typedef struct STnode *link;
struct STnode { Item item; link l, r; int N };

static link head, z;

link NEW(Item item, link l, link r, int N)
  { link x = malloc(sizeof *x);
    x->item = item; x->l = l; x->r = r; x->N = N;
    return x;
  }
```

Uma implementação de uma TS com uma ABB

```
[...]
void STinit()
  { head = (z = NEW(NULLitem, 0, 0, 0)); }

int STcount() { return head->N; }

Item searchR(link h, Key v)
  { Key t = key(h->item);
    if (h == z) return NULLitem;
    if eq(v, t) return h->item;
    if less(v, t) return searchR(h->l, v);
                  else return searchR(h->r, v);
  }
Item STsearch(Key v)
  { return searchR(head, v); }
```

Uma implementação de uma TS com uma ABB

```
[...]
link insertR(link h, Item item)
  { Key v = key(item), t = key(h->item);
    if (h == z) return NEW(item, z, z, 1);
    if less(v, t)
         h->l = insertR(h->l, item);
    else h->r = insertR(h->r, item);
    (h->N)++; return h;
  }
void STinsert(Item item)
  { head = insertR(head, item); }
```

Ordenação e ABBs

```
/* prog12.8.c */
void sortR(link h, void (*visit)(Item))
  {
    if (h == z) return;
    sortR(h->l, visit);
    visit(h->item);
    sortR(h->r, visit);
  }
void STsort(void (*visit)(Item))
  { sortR(head, visit); }
```

Inserção sem recursão

```
/* prog12.9.c */
void STinsert(Item item)
  { Key v = key(item); link p = head, x = p;
    if (head == NULL)
      { head = NEW(item, NULL, NULL, 1); return; }
    while (x != NULL)
      { p = x; x->N++;
        x = less(v, key(x->item)) ? x->l : x->r;
      }
    x = NEW(item, NULL, NULL, 1);
    if (less(v, key(p->item))) p->l = x;
                          else p->r = x;
  }
```

ABBs aleatórias

Suponha que temos N inteiros distintos. Podemos construir uma ABB aleatória T_N escolhendo uma permutação destes inteiros uniformemente ao acaso dentre as $N!$ permutações possíveis e inserindo estes inteiros na ordem escolhida em uma ABB inicialmente vazia.

Propriedade 1. *A altura esperada de* T_N *é* $O(\log N)$.

Exercício importante

Exercício 2. *Gere ABBs aleatórias* T_N *para valores grandes de* N *e determine a altura das árvores geradas. Compare o resultado com o resultado enunciado na Propriedade 1. Estime a constante na cota* $O(\log N)$ *experimentalmente.*