Recursão

Além do Sedgewick (**sempre** leiam o Sedgewick), veja

- `http://www.ime.usp.br/~pf/algoritmos/aulas/recu.html`
- `http://www.ime.usp.br/~pf/algoritmos/aulas/enum.html`

Recursão

- Definições recursivas em matemática
- Provas por indução finita
- Algoritmos definidos de forma recursiva

Algumas definições recursivas

Suponha que definimos $f(n)$ para $n \geq 0$ da seguinte forma:

$$f(n) = \begin{cases} 1 & \text{se } n = 0 \\ nf(n-1) & \text{se } n > 0. \end{cases} \quad (1)$$

Temos então que os valores de $f(n)$ ($n \geq 0$) são

$$1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, \ldots \quad (2)$$

Algumas definições recursivas

Conhecemos $f(n)$; esta é a função *fatorial*:

$$f(n) = n! = n(n-1)\ldots 1 = \prod_{1 \leq k \leq n} k. \tag{3}$$

Algumas definições recursivas

Suponha que definimos $g(m, n)$ para m, $n \geq 0$, com m e n são simultaneamente nulos da seguinte forma:

$$g(m, n) = \begin{cases} m & \text{se } n = 0 \\ g(n, m \bmod n) & \text{se } n > 0. \end{cases} \tag{4}$$

- Tabela?

Algumas definições recursivas

Também conhecemos $g(m, n)$; esta é a função mdc:

$$g(m, n) = \mathrm{mdc}(m, n) = (m, n). \tag{5}$$

Algumas definições recursivas

Prova. Provamos (5) por indução em n. Se $n = 0$, a identidade (5) vale. Suponha agora que $n > 0$ e que (5) vale para valores menores de n. Fixe $m \geq 0$. Temos que

$$(m, n) = (n, m \bmod n) \tag{6}$$

[isto exige uma prova, que estamos omitindo]. Entretanto, o lado direito de (6) é $g(n, m \bmod n)$ por hipótese de indução, e este último valor é $g(m, n)$, por (4). Assim concluímos que $g(m, n) = (m, n)$, o que completa o passo de indução. A identidade (5) segue pelo princípio da indução finita. □

Um algoritmo recursivo

```
int gcd(int m, int n)
  {
    if (n == 0) return m;
    return gcd(n, m % n);
  }
```

Expressões em notação pré-fixa

Expressão `E` em notação *pré-fixa*:

$$\texttt{E} = \begin{cases} \texttt{<natural>} \\ \texttt{+ F G} \\ \texttt{* F G}, \end{cases} \tag{7}$$

onde `F` e `G` são expressões em notação pré-fixa. (Por simplicidade, ignoramos as operações `-`, `/`, `mod`.)

Expressões em notação pré-fixa

Valor $\mathsf{val}(\texttt{E})$ de uma expressão `E` em notação *pré-fixa*:

$$\mathsf{val}(\texttt{E}) = \begin{cases} \texttt{<natural>} & \text{se } \texttt{E = <natural>} \\ \mathsf{val}(\texttt{F}) + \mathsf{val}(\texttt{G}) & \text{se } \texttt{E = + F G} \\ \mathsf{val}(\texttt{F}) \times \mathsf{val}(\texttt{G}) & \text{se } \texttt{E = * F G.} \end{cases} \tag{8}$$

Avaliação de expressões em notação pré-fixa

```
/* prog5.4.c */
#include <stdio.h>

char *a; int i;
int eval()
  { int x = 0;
    while (a[i] == ' ') i++;
    if (a[i] == '+') { i++; return eval() + eval(); }
    if (a[i] == '*') { i++; return eval() * eval(); }
    while ((a[i] >= '0') && (a[i] <= '9'))
      x = 10*x + (a[i++]-'0');
    return x;
  }
```

Avaliação de expressões em notação pré-fixa

```
/* prog5.4.c */
[...]

int main(int argc, char *argv[])
{
  a=argv[1];
  printf("%d\n", eval());
  return 0;
}
```

Avaliação de expressões em notação pré-fixa

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.4 "+ 2 3"
5
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.4 "* + 2 3 * 2 4"
40
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.4 "* + 7 * * 4 6 + 8 9 5"
2075
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$
```

Manipulação de listas ligadas com recursão

[Listas sem cabeça e sem cauda]

```
/* prog5.5.c */
int count(link x)
  {
    if (x == NULL) return 0;
    return 1 + count(x->next);
  }
```

Manipulação de listas ligadas com recursão

```
/* prog5.5.c */
[...]
void traverse(link h)
  {
    if (h == NULL) return;
    processe(h); /* funcao de processamento da celula h */
    traverse(h->next);
  }
void traverseR(link h)
  {
    if (h == NULL) return;
    traverseR(h->next);
    processe(h); /* funcao de processamento da celula h */
  }
```

Manipulação de listas ligadas com recursão

```
/* prog5.5.c */
[...]
link delete(link x, Item v)
  {
    if (x == NULL) return NULL;
    if (eq(x->item, v)) /* teste de igualdade */
      { link t = x->next; free(x); return t; }
    x->next = delete(x->next, v);
    return x;
  }
```

Torres de Hanói

```
/* prog5.7.c */
#include <stdio.h>
#include <stdlib.h>
void hanoi(int N, int d)
  { if (N == 0) return;
    hanoi(N-1, -d);
    printf("%d ", d*N); /* shift(N, d); */
    hanoi(N-1, -d);
  }
```

Torres de Hanói

```
/* prog5.7.c */
[...]
int main(int argc, char *argv[])
{ int N=atoi(argv[1]);
  hanoi(N,1);
  putchar('\n');
  return 0;
}
```

Torres de Hanói

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.7 1
1
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.7 2
-1 2 -1
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.7 3
1 -2 1 3 1 -2 1
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.7 4
-1 2 -1 -3 -1 2 -1 4 -1 2 -1 -3 -1 2 -1
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.7 5
1 -2 1 3 1 -2 1 -4 1 -2 1 3 1 -2 1 5 1 -2 1 3 1 -2 1 -4 1 -2 1 3 1 -2 1
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$
```

Exercício

Sugestão: estude e faça os exercícios em

- `http://www.ime.usp.br/~pf/algoritmos/aulas/recu.html`
- `http://www.ime.usp.br/~pf/algoritmos/aulas/enum.html`

Exercícios

Eis um par de exercícios das páginas acima:

Exercício 1 (Subset Sum)**.** *Suponha que você emitiu cheques nos valores de* $p[1], \ldots, p[n]$ *ao longo de um mês. No fim do mês, o banco informa que um total* t *foi descontado de sua conta. Quais dos cheques foram descontados? Por exemplo, se* $p = \{61, 62, 63, 64\}$ *e* $t = 125$ *então só há duas possibilidades: ou foram descontados os cheques* 1 *e* 4 *ou foram descontados os cheques* 2 *e* 3*. Este é o problema da "soma de subconjunto", também conhecido como subset sum:*

Dado um número t *e um vetor* $p[1 \ldots n]$*, encontrar todas as subseqüências* $s[1 \ldots k]$ *de* $1 \ldots n$ *tais que* $p[s[1]] + \cdots + p[s[k]] = t$*.*

Escreva uma função que resolva este problema.

Exercícios

Exercício 2 (Combinações)**.** *Escreva uma função que imprima todas as subseqüências de* $1 \ldots n$ *que têm exatamente* k *elementos. (Isso corresponde, exatamente, aos subconjuntos de* $\{1, \ldots, n\}$ *que têm* k *elementos.)*