Universidade Federal de Goiás – Instituto de Informática

# Análise e Projeto de Algoritmos
## Relações de Recorrência

**Diogo Stelle**

diogostelle@inf.ufg.br / diogo.stelle@gmail.com

**2019**

INSTITUTO DE INFORMÁTICA
UFG

# Relações de Recorrência

- Definições Recorrentes
    - Uma definição onde o item a ser definido aparece como parte da definição é chamada de **definição recorrente** ou **definição por recorrência** ou ainda **definição por indução**.
    - Como definir algo em torno de si mesmo?
        - 1. Uma base, ou condição básica, onde alguns casos simples (pelo menos um) do item que está sendo definido são dados explicitamente.
        - 2. Um passo de indução ou recorrência, onde novos casos do item que está sendo definido são dados em função dos casos anteriores.

# Relações de Recorrência

- Definições Recorrentes
  - O item 1 nos dá o começo, fornecendo casos simples e concretos
  - O item 2 nos permite construir novos casos, a partir dos simples e ainda outros casos a partir desses novos e assim por diante.

# Relações de Recorrência

- Sequências definidas por Recorrência
    - Uma sequência S é uma lista de objetos que são numerados em uma determinada ordem.
        - Existe um primeiro objeto, um segundo e assim por diante.
        - S(k) denota o k-ésimo objeto da sequência.
        - Uma sequência é definida por recorrência nomeando-se o primeiro valor (ou alguns primeiros) na sequência e depois definindo os demais valores subsequentes em termos dos valores anteriores.

# Relações de Recorrência

- Sequências definidas por Recorrência
  - **Exemplo 1**: A sequência S é definida por recorrência por
    1. S(1) = 2
    2. S(n) = 2 * S(n -1) para n ≥ 2

  Pela proposição 1, S(1) = 2. Depois, pela proposição 2, o segundo objeto em S é S(2) = 2 S(1) = 2 (2) = 4. Novamente, pela proposição 2, S(3) = 2 S(2) = 2(4) = 8. Continuando desse modo, vemos que a sequência S é: 2, 4, 8, 16, 32...

  - Uma regra como a da proposição 2, que define um valor de uma sequência em termos de um ou mais valores anteriores é uma **relação de recorrência**.

# Relações de Recorrência

- Algoritmos Recursivos
  - Um algoritmo recursivo resolve um problema **reduzindo-o** para uma **instância do mesmo problema** com dados de entrada menores.

**Com Recursão**

```
int fatorial (int n){
  if (n == 0)
    return 1;
  else
    return n*fatorial(n
          -1);
}
```

**Sem Recursão**

```
int fatorial (int n){
  if (n == 0)
    return 1;
  else {
    int i, f = 1;
    for (i=2; i <= n;i
        ++)
      f = f * i;
    return f;
  }
}
```

# Relações de Recorrência

- Algoritmos Recursivos
  - Pode-se usar indução para provar que um algoritmo recursivo está correto, ou seja:
    - ele produz a saída desejada para todas as entradas possíveis.

# Relações de Recorrência

- Resolvendo Relações de Recorrência
    - Desenvolvemos um algoritmo recorrente para calcular o valor de $S(n)$ no exemplo 1, no entanto, existe uma maneira mais fácil para calcular $S(n)$

Lembre que

$$S(1) = 2$$

$$S(n) = 2 * S(n-1), \qquad \text{para } n \geq 2$$

Então,

$S(1) = 2 = 2^1$ $\qquad$ $S(2) = 4 = 2^2$

$S(3) = 8 = 2^3$ $\qquad$ $S(4) = 16 = 2^4$

# Relações de Recorrência

- Resolvendo Relações de Recorrência
  - Fazendo isso, podemos **conjecturar** que $S(n) = 2^n$
  - Ou seja, podemos substituir $n$ e calcular diretamente o valor de S(n).
  - Uma equação dessa forma é chamada de **solução fechada** para a relação de recorrência S(n) sujeita a S(1).
  - Quando pudermos encontrar uma solução fechada, dizemos que **resolvemos** a relação de recorrência.
  - Por fim, essa conjectura deve ser verificada por meio de **indução matemática**.

# Relações de Recorrência

- Resolvendo Relações de Recorrência
    - Uma das técnicas para se resolver uma relação de recorrência consiste na abordagem **“expandir, conjecturar e verificar”**. Vamos mostrar o passo a passo para a recorrência

$$S(1) = 2$$

$$S(n) = 2 * S(n - 1), \qquad \text{para } n \geq 2$$

    - **Passo 1)** Expandir a sequência S para n, (n-1), (n-2), ... , k, ...1

# Relações de Recorrência

- Resolvendo Relações de Recorrência

$$S(n) = 2 \cdot S(n-1) \qquad \text{(após 1 expansão)}$$
$$S(n) = 2 \cdot \left[2 \cdot S(n-2)\right] = 2^2 \cdot S(n-2) \qquad \text{(após 2 expansões)}$$
$$S(n) = 2^2 \cdot \left[2 \cdot S(n-3)\right] = 2^3 \cdot S(n-3) \qquad \text{(após 3 expansões)}$$
$$S(n) = 2^3 \cdot \left[2 \cdot S(n-4)\right] = 2^4 \cdot S(n-4) \qquad \text{(após 4 expansões)}$$
$$\vdots$$
$$S(n) = 2^k \cdot S(n-k) \qquad \text{(após } k \text{ expansões)}$$
$$\vdots$$
$$S(n) = 2^{n-1} \cdot S(n-(n-1)) \qquad \text{(após } n-1 \text{ expansões)}$$
$$S(n) = 2^{n-1} \cdot S(1) = 2^{n-1} \cdot 2$$
$$\boxed{S(n) = 2^n} \qquad \textbf{(solução em forma fechada)}$$

# Relações de Recorrência

- Resolvendo Relações de Recorrência
  - **Passo 2)** Após obter a solução fechada, **conjecturamos**:

    $$S(n) = 2^n \text{ para todo inteiro } n \geq 1$$

  - **Passo 3)** Verificar a conjectura através de indução matemática para provar a forma fechada.

# Relações de Recorrência

- Prova por indução
  - Passo base: $S(1) = 2 = 2^1$
  - Passo indutivo: assumir que $P(k)$ é verdadeiro, isto é,

$$S(k) = 2^k$$

  - Mostrar que se $S(k) \rightarrow S(k + 1)$, i. e., $S(k + 1) = 2^{k+1}$

$$S(k+1) = 2 * S(k)$$

$$S(k+1) = 2 * 2^k$$

$$S(k+1) = 2^{k+1}$$

# Exercícios

# Exercícios

- **1-** Na função P definida recursivamente abaixo, encontre P(2), P(3), P(4) e P(5).

$$\begin{cases} P(1) = 2 \\ P(n) = 2P(n-1), \quad para\, n > 1 \end{cases}$$

- **2-** Dê uma definição recursiva para $P(n) = a^n$, onde $a \in \mathbb{R}^*$ e $n \in \mathbb{Z}^*$.

# Exercícios

- **3-** Encontre uma solução em forma fechada para a seguinte relação de recorrência:

$$\begin{cases} T(1) = 1 \\ T(n) = T(n-1) + 3, \qquad para\ n > 1 \end{cases}$$

- **4-** O jogo de "Torre de Hanói" tem a seguinte relação de recorrência

$$\begin{cases} T(1) = 1 \\ T(n) = 2 \cdot T(n-1) + 1, \qquad para\ n > 1 \end{cases}$$

Utilize o método "expandir, conjecturar e verificar" para encontrar uma solução fechada que dê o número de movimentações necessárias para completar o jogo com n discos.

# Análise de Algoritmos Recursivos