ANÁLISE DE ALGORITMOS: PARTE 4

Prof. André Backes

Relações de Recorrências

- Função recursiva
  - Função que chama a si mesma durante a sua execução
- Exemplo: fatorial de um número **N.**
  - Para **N = 4** temos
    - 4! = 4 * 3!
    - 3! = 3 * 2!
    - 2! = 2 * 1!
    - 1! = 1 * 0!
    - 0! = 1

Relações de Recorrências

3

- Função recursiva
  - Matematicamente, o fatorial é definido como
    - N! = N * (N-1)!
    - 0! = 1
  - Implementação

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

Relações de Recorrências

4

- Recorrência ou Relação de Recorrência
  - Expressão que descreve uma função em termos de entradas menores da função
    - Exemplo: definição de um função recursiva
  - Muitos algoritmos se baseiam em recorrência
  - Ferramenta importante para a solução de problemas combinatórios
- Relação de recorrência do fatorial
  - T(n) = T(n-1) + n

Relações de Recorrências

5

- Complexidade da recorrência
  - Uma recursão usualmente não utiliza estruturas de repetição, apenas comandos condicionais, atribuições etc
  - Podemos erroneamente imaginar que essa funções possuem complexidade ***O(1)***

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

Relações de Recorrências

6

- Complexidade da recorrência
  - Saber a complexidade da recursão envolve resolver a sua relação de recorrência
  - T(n) = T(n-1) + n

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

Relações de Recorrências

7

- Complexidade da recorrência
  - Temos que encontrar uma **fórmula fechada** que nos dê o valor da função **T(n) = T(n-1) + n** em termos de seu parâmetro **n**
    - Geralmente obtido como uma combinação de polinômios, quocientes de polinômios, logaritmos, exponenciais etc.

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

Relações de Recorrências

8

- Considere a seguinte relação de recorrência
  - **T(n) = T(n-1) + 2n + 3**
- Para $n \in \{2, 3, 4, \ldots\}$, existem inúmeras funções **T** que satisfazem a recorrência
  - Depende do **caso base, T(1)**
  - Exemplos
    - **T(1) = 1**

n	*1*	*2*	*3*	*4*	*5*
T(n)	*1*	*8*	*17*	*28*	*41*

    - **T(1) = 5**

n	*1*	*2*	*3*	*4*	*5*
T(n)	*5*	*12*	*21*	*32*	*45*

Relações de Recorrências

9

- Problema
  - Para cada valor **i** e o intervalo $n \in \{2, 3, 4, \ldots\}$ existe uma (e apenas uma) função **T** que tem caso base **T(1) = i** e satisfaz a recorrência
  - **T(n) = T(n-1) + 2n + 3**

n	*1*	*2*	*3*	*4*	*5*
T(n)	*1*	*8*	*17*	*28*	*41*

n	*1*	*2*	*3*	*4*	*5*
T(n)	*5*	*12*	*21*	*32*	*45*

Relações de Recorrências

10

- Solução
  - Precisamos encontrar uma **fórmula fechada** para a recorrência
  - Podemos expandir a relação de recorrência **T(n)=T(n-1) + 2n + 3** até que se possa detectar um comportamento no seu caso geral

Relações de Recorrências

11

- Para entender essa técnica de expansão, considere a seguinte recorrência
  - **T(n) = T(n-1) + 3**
  - Essa relação de recorrência representa um algoritmo que possui 3 operações mais uma chamada recursiva

Relações de Recorrências

12

- Expandindo a recorrência **T(n) = T(n-1) + 3**
  - Se aplicarmos o termo **T(n-1)** sobre a relação **T(n)**. Com isso, obtemos
    - **T(n-1) = T(n-2) + 3**
  - Se aplicarmos o termo **T(n-2)** sobre a relação **T(n)**, teremos
    - **T(n-2) = T(n-3) + 3**

Relações de Recorrências

13

- Expandindo a recorrência **T(n) = T(n-1) + 3**
  - Se continuarmos esse processo, teremos a seguinte expansão
    - **T(n) = T(n-1) + 3**
    - **T(n) = (T(n-2) + 3) + 3**
    - **T(n) = ((T(n-3) + 3) + 3) + 3**
  - Perceba que a cada passo um valor 3 é somado a expansão e o valor de **n** é diminuído em uma unidade

Relações de Recorrências

14

- Expandindo a recorrência **T(n) = T(n-1) + 3**
  - Podemos resumir essa expansão para usando a seguinte equação
    - **T(n) = T(n-k) + 3k**
  - Resta saber quando esse processo de expansão termina
    - Isso ocorre no **caso base**

Relações de Recorrências

15

- Expandindo a recorrência **T(n) = T(n-1) + 3**
  - O **caso base** ocorre quando **n-k = 1** ou seja, **k=n-1**
  - Substituindo, temos
    - **T(n) = T(n-k) + 3k**
    - **T(n) = T(1) + 3(n-1)**
    - **T(n) = T(1) + 3n - 3**

Relações de Recorrências

16

- Expandindo a recorrência **T(n) = T(n-1) + 3**
- Obtemos **T(n) = T(1) + 3n - 3**
  - **T(1)** é o **caso base**: recursão termina
  - Logo, seu custo é constante: ***O(1)***
- Complexidade da recorrência
  - **T(n) = 3n - 3 + *O(1)***
  - Ou seja, **linear**: ***O(n)***

Relações de Recorrências

17

- Outro exemplo: considere a seguinte recorrência
  - **T(n) = T(n/2) + 5**
  - Essa relação de recorrência representa um algoritmo que possui 5 operações mais uma chamada recursiva que divide os dados sempre pela metade **(n/2)**

Relações de Recorrências

18

- Neste caso, a recorrência existe apenas para valores de **n** que representem uma potência de 2
  - $n \in \{2^1, 2^2, 2^3, \ldots\}$
- Considerando $\mathbf{n = 2^k}$, podemos reescrever a recorrência como
  - $\mathbf{T(2^k) = T(2^{k-1}) + 5}$

Relações de Recorrências

19

- Expandindo a recorrência $T(2^k) = T(2^{k-1}) + 5$
  - Se aplicarmos o termo $T(2^{k-1})$ sobre a relação $T(2^k)$. Com isso, obtemos
    - $T(2^{k-1}) = T(2^{k-2}) + 5$
  - Se aplicarmos o termo $T(2^{k-2})$ sobre a relação $T(2^k)$, teremos
    - $T(2^{k-2}) = T(2^{k-3}) + 5$

Relações de Recorrências

20

- Expandindo a recorrência $T(2^k) = T(2^{k-1}) + 5$
  - Se continuarmos esse processo, teremos a seguinte expansão
    - $T(2^k) = T(2^{k-1}) + 5$
    - $T(2^k) = (T(2^{k-2}) + 5) + 5$
    - $T(2^k) = ((T(2^{k-3}) + 5) + 5) + 5$
  - Perceba que a cada passo um valor *5* é somado a expansão e o valor de **k** é diminuído em uma unidade

Relações de Recorrências

21

- Expandindo a recorrência $\mathbf{T(2^k) = T(2^{k-1}) + 5}$
  - Ao final da expansão, teremos
    - $\mathbf{T(2^k) = T(2^{k-k}) + 5k}$
    - $\mathbf{T(2^k) = T(2^0) + 5k}$
    - $\mathbf{T(2^k) = T(1) + 5k}$
  - Podemos resumir essa expansão usando a seguinte equação, a qual já considera o seu caso base
    - $\mathbf{T(2^k) = T(1) + 5k}$

Relações de Recorrências

22

- Expandindo a recorrência $\mathbf{T(2^k) = T(2^{k-1}) + 5}$
  - Temos que substituir o custo do **caso base**, ***O(1)***
  - Complexidade da recorrência
    - $\mathbf{T(2^k) = O(1) + 5k}$
  - Devemos lembrar que substituímos **n** por $\mathbf{2^k}$ no início da expansão, de modo que $\mathbf{n = 2^k}$

Relações de Recorrências

23

- Expandindo a recorrência $\mathbf{T(2^k) = T(2^{k-1}) + 5}$
  - Aplicando o logaritmo em $\mathbf{n = 2^k}$, temos que $\mathbf{k=\log_2 n}$
  - Substituindo, temos
    - $\mathbf{T(2^k) = O(1) + 5k}$
    - $\mathbf{T(n) = O(1) + 5 \log_2 n}$
- Complexidade da recorrência
  - $\mathbf{T(n) = O(1) + 5 \log_2 n}$
  - Ou seja, **logarítmica**: $\boldsymbol{O(\log_2 n)}$

Material Complementar

24

- Vídeo Aulas
  - Aula 99: Análise de Algoritmos
  - Aula 100: Análise de Algoritmos – Contando Instruções
  - Aula 101: Análise de Algoritmos – Comportamento Assintótico
  - Aula 102: Análise de Algoritmos – Notação Grande-O
  - Aula 103: Análise de Algoritmos – Tipos de Análise Assintótica
  - Aula 104: Análise de Algoritmos – Classes de Problemas
  - Aula 122 – Relações de Recorrência