Análise e Técnicas de Algoritmos
Período 2003.1

Relação de Recorrência

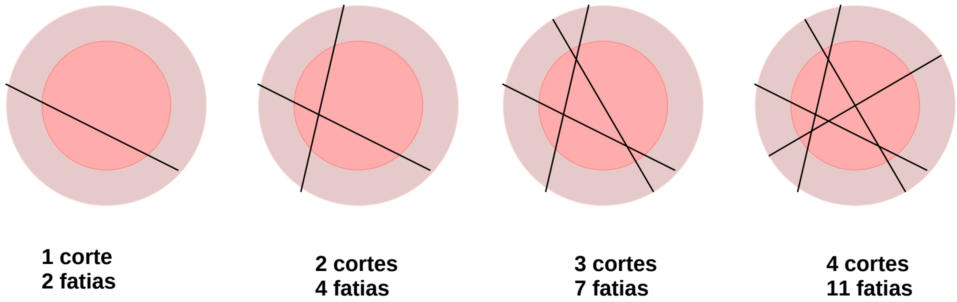
Análise de Algoritmos Recursivos

- A análise de um algoritmo recursivo requer a resolução de uma recorrência.
- Uma recorrência é um algoritmo recursivo que calcula o valor de uma função em um ponto dado.
- Uma recorrência define T(n) em termos de T(n-1), T(n-2), etc.

Exemplo 1:
$T(1) = 1$ e
$T(n) = T(n-1) + 3n + 2$, para $n = 2, 3, 4$, etc.
Para valores pequenos de n, temos os seguintes valores de T(n):

n	1	2	3	4	5	6
$T(n)$	1	9	20	34	51	71

Exemplo 2:
Considere a figura abaixo:



Quantos pedaços obtemos com N cortes na pizza?

É possível observar que o N-ésimo corte cria N novas fatias. Logo, o número total de fatias obtido com N cortes, denotado por $P(N)$, é dado pela seguinte relação de recorrência:

- $P(1) = 2$
- $P(N) = P(N-1) + N$

Derivando Relações de Recorrência

Como proceder para derivar uma relação de recorrência para a análise do tempo de execução de um algoritmo:

- Determinar qual o tamanho n do problema.
- Verificar que valor de n é usado como base da recursão. Em geral é um valor único ($n = 1$, por exemplo), mas pode ser valores múltiplos. Vamos considerar esse valor como n_0.

- Determinar $T(n_0)$. Pode-se usar uma constante c, mas, em muitos, casos um número específico é necessário.
- $T(n)$ é definido como uma soma de várias ocorrências de $T(m)$ (chamadas recursivas), mais a soma de outras instruções efetuadas. Em geral, as chamadas recursivas estão relacionadas com a subproblemas do mesmo tamanho $f(n)$, definindo um termo $a.T(f(n))$ na relação de recorrência.

Logo, a relação de recorrência é definida como:

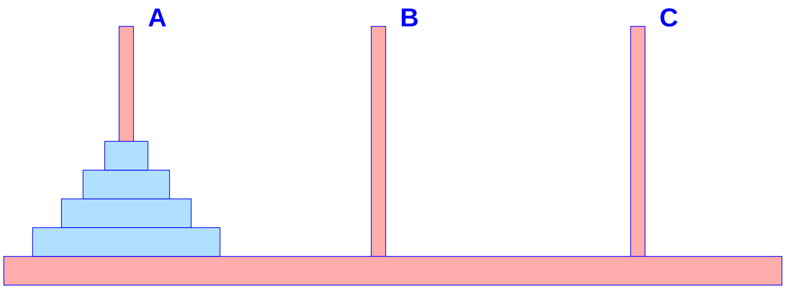
- $T(n) = c$, se $n = n_0$
- $T(n) = a.T(f(n)) + g(n)$, caso contrário

Exemplo 1: O problema do corte das pizzas

No problema do corte das pizzas, a relação de recorrência $f(N)$ é recursivamente definida:

- $f(1) = 2$ (caso base)
- $f(N) = f(N-1) + N$, para $N \geq 2$ (caso recursivo)

Exemplo 2: Torres de Hanoi



- **Objetivo:** Transferir todos os n discos de A para C.
- **Regras:**
  - Mover um disco por vez.
  - Nunca colocar um disco maior em cima de um menor.
- **Solução Recursiva:**
  - Transferir $n-1$ discos de A para B.
  - Mover o maior disco de A para C.
  - Transferir $n-1$ discos de B para C.
- **Algoritmo**

Hanoi(inicial, final, temp, n)
if $n > 1$ **then**
  Hanoi(inicial, temp, final, $n-1$)
Move(inicial, final)
if $n > 1$ **then**
  Hanoi(inicial, temp, final, $n-1$)

- **Relação de Recorrência:**
  - $T(1) = 1$
  - $T(n) = 2.T(n-1) + 1$

Exemplo 3: MergeSort

- **Objetivo:** Ordenar uma lista L de n elementos.
- **Algoritmo**
  mergesort(L, n)
  if $n \leq 1$ **then**
    return L
  else
    return $merge(mergesort(L_1, n/2), mergesort(L_2, n/2)))$
- **Relação de Recorrência:**
  - $T(n) = c$, se $n \leq 1$
  - $T(n) = 2.T(n/2) + d.n$, caso contrário

Resolvendo Relações de Recorrências

- Resolver uma relação de recorrência nem sempre é fácil.
- Resolvendo uma relação de recorrência, determina-se o tempo de execução do algoritmo recursivo correspondente.
- Relação de recorrência: $T(n) = T(n_1) + T(n_2) + ... + T(n_a) + f(n)$.
- Em geral, os a subproblemas têm o mesmo tamanho que é uma fração de n, digamos (n/b).
- A fórmula pode ser generalizada como: $T(n) = a.T(n/b) + f(n)$
- Por exemplo, no caso do **mergesort**, $a = b = 2$ e $f(n) = n$.
- Como resolver uma relação de recorrência:
  - Método do chute e prova por indução.
  - Método da árvore de recursão.
  - Método do desdobramento ou iterativo.
  - Método master.

Método do Chute e Prova por Indução

Exemplo 1:

Seja a seguinte relação de recorrência, apresentada anteriormente:

- $T(1) = 1$
- $T(n) = T(n-1) + 3n + 2$, para $n = 2, 3, 4$, etc.

A relação de recorrência é resolvida em duas partes:

1. **Chute:** $T(n) = 3n^2/2 + 7n/2 - 4$.
2. **Prova:**
   - Caso base: $T(1) = 3.1^2/2 + 7.1/2 - 4 = 1,5 + 3,5 - 4 = 1$.
   - Supondo que $n > 1$ e que é válido para $n-1$, temos que provar para n:

$$\begin{aligned} T(n) &= T(n-1) + 3n + 2 \\ &= 3(n-1)^2/2 + 7(n-1)/2 - 4 + 3n + 2 \\ &= (3n^2 - 6n + 3 + 7n - 7 - 4 + 6n + 4)/2 \\ &= (3n^2 + 7n - 8)/2 \end{aligned}$$

Como a fórmula está correta, prova-se que $T(n) = O(n^2)$.

Exemplo 2:

Seja a seguinte relação de recorrência:

- $T(1) = 1$
- $T(n) = 2T(n/2) + n$, para $n \geq 2$.

A relação de recorrência é resolvida em duas partes:

1. **Chute:** $T(n) = n + nlogn$.
2. **Prova:**
   - Caso base: $T(1) = 1 + log 1 = 1$.
   - Passo indutivo:

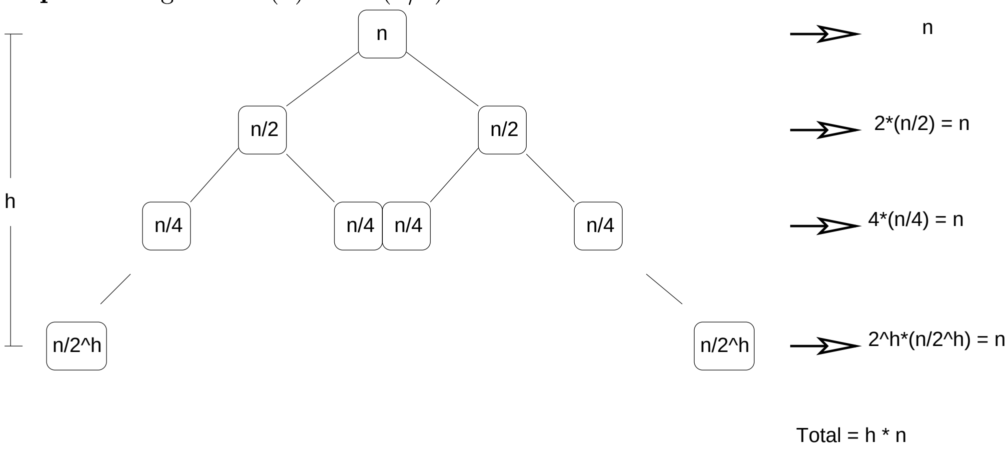
$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2(n/2 + n/2 \log \mathrm{n}/2) + n \\ &= n + n(\log \mathrm{n} - 1) + n \\ &= n + n.\log \mathrm{n} \end{aligned}$$

Como a fórmula está correta, prova-se que $T(n) = O(n.\log \mathrm{n})$.

Método da Árvore de Recursão

- Talvez o método mais intuitivo.
- Consiste em desenhar uma árvore cujos nós representam os tamanhos dos correspondentes problemas.
- Cada nível i contém todos os subproblemas de profundidade i.
- Dois aspectos importantes:
  - A altura da árvore.
  - O número de passos executados de cada nível.
- A solução da recorrência (tempo de execução do algoritmo) é a soma de todos os passos de todos os níveis.
- No caso particular em que o total de passos de cada nível é o mesmo, $l(n)$ por exemplo, a solução é: $T(n) = h.l(n)$, onde h é a altura da árvore.

Exemplo: Mergesort: $T(n) = 2.T(n/2) + n$



A altura da árvore é $h =$ log n. Logo, $O(n.$log n$)$.

Método do Desdobramento

Esse método é o da árvore de recursão, representado de forma algébrica. Consiste em:

- Usar (algumas poucas) substituições repetidamente até encontrar um padrão.
- Escrever uma fórmmula em termos de ne o número de substituições i.
- Escolher i de tal forma que todas as referências a $T()$ sejam referências ao caso base.

- Resolver a fórmula.

Exemplo 1: Solução da recorrência do problema da pizza.
Vamos considerar novamente o exemplo do corte das pizzas.

- $f(1) = 2$
- $f(n) = f(n-1) + n$, para $n \geq 2$.

A relação de recorrência é resolvida fazendo repetidas substituições:
$f(n) = f(n-1) + n$
$f(n) = f(n-2) + (n-1) + n$
$f(n) = f(n-3) + (n-2) + (n-1) + n$
...
$f(n) = f(n-i) + (n-i+1) + ... + (n-1) + n$
O caso base é alcançado quando $i = n-1$. Logo,
$f(n) = 2 + 2 + 3 + ... + (n-2) + (n-1) + n$
$f(n) = n.(n-1)/2 + 1$
Logo $f(n) = O(n^2)$.

Exemplo 2: Solução da recorrência do problema da Torre de Hanoi.

- $T(1) = 1$
- $T(n) = 2.T(n-1) + 1$.

Solução por desdobramento:
$T(n) = 2(2.T(n-2) + 1) + 1$
$= 4.T(n-2) + 2 + 1$
$= 4.(2.T(n-3) + 1) + 2 + 1$
$= 8.T(n-3) + 4 + 2 + 1$
...
$= 2^i.T(n-i) + 2^{i-1} + 2i^{i-2} + ... + 2^1 + 2^0$
O caso base é alcançado quando $i = n-1$. Logo,
$T(n) = 2^{n-1} + 2^{n-2} + 2^{n-3} + ... + 2^1 + 2^0$
Isso é uma **soma geométrica**. Logo, $T(n) = 2^n - 1 = O(2^n)$

Exemplo 3: Mergesort.

- $T(n) = c$
- $T(n) = 2.T(n/2) + d.n$.

Solução por desdobramento:
$T(n) = 2.T(n/2) + d.n$
$= 2.(2.T(n/4) + d.n/2) + d.n$
$= 4.T(n/4) + d.n + d.n$
$= 4.(2.T(n/8) + d.n/4) + d.n + d.n$
$= 8.T(n/8) + d.n + d.n + d.n$

...

$= 2^i.T(n/2^i) + i.d.n$

O caso base é alcançado quando $i = \log n$. Logo,

$T(n) = 2^{\log n}.T(n/2^{\log n}) + d.n.\log n$

$= d.n.\log n + c.n$

Portanto, $T(n) = O(n.\log n)$

Método Master

- Teorema que resolve quase todas as recorrências.
- $T(n)$ é da forma $a.T(n/b) + f(n)$, $a, b > 1$.
- Casos:
    1. se $f(n) \in O(n^{log_b^a - \varepsilon})$, para algum $\varepsilon > 0$, temos que $T(n) = \Theta(n^{log_b^a})$.
    2. se $f(n) \in \Theta(n^{log_b^a})$, temos que $T(n) = \Theta(n^{log_b^a}.\log n)$.
    3. se $f(n) \in \Omega(n^{log_b^a + \varepsilon})$, para algum $\varepsilon > 0$, e se $af(n/b) \leq cf(n)$ para algum $c > 0$, e n suficientemente grande, temos que $T(n) = \Theta(f(n))$.
- A prova do teorema é complexa.
- Esse método é fácil de usar. Entretanto, muitos casos são *casca de banana* e podemos cometer erros.

Exemplo: Mergesort: $T(n) = 2.T(n/2) + n$

- $a = b = 2, f(n) = n$.
- $log_b^a = 1$. Logo caímos no segundo caso.
- $T(n) = \Theta(n.log_b^a.\log n) = \Theta(n.\text{logn})$

Exemplo: $T(n) = 9.T(n/3) + n$

- $a = 9, b = 3, f(n) = n$.
- $log_b^a = 2$. O segundo caso não pode ser aplicado.
- $f(n) = O(n^{log_3^{n} - \varepsilon})$, se $\varepsilon = 1$. Logo caímos no caso 1.
- $T(n) = \Theta(n.log_b^a) = \Theta(n.log_3^9) = \Theta(n^2)$