MASTER THEOREM - GUIDA COMPLETA

Risoluzione ricorrenze divide et impera per esame

SCHEMA GENERALE

Forma standard:

```
T(n) = a·T(n/b) + f(n)
```

Dove:

- **a ≥ 1**: numero di sottoproblemi
- **b > 1**: fattore di riduzione dimensione
- **f(n)**: costo della divisione e combinazione

I TRE CASI DEL MASTER THEOREM

Confronta **f(n)** con **n^(log_b a)**:

CASO 1: f(n) è polinomialmente MINORE

```
Se f(n) = O(n^(log_b a - ε)) per qualche ε > 0

Allora: T(n) = Θ(n^(log_b a))
```

Interpretazione: Il lavoro è concentrato nelle foglie dell'albero di ricorsione

CASO 2: f(n) è UGUALE (a meno di log)

```
Se f(n) = Θ(n^(log_b a) · log^k n) per qualche k ≥ 0

Allora: T(n) = Θ(n^(log_b a) · log^(k+1) n)
```

Interpretazione: Il lavoro è distribuito uniformemente tra i livelli

CASO 3: f(n) è polinomialmente MAGGIORE

```
Se f(n) = Ω(n^(log_b a + ε)) per qualche ε > 0
E  a·f(n/b) ≤ c·f(n) per qualche c < 1 e n sufficientemente grande

Allora: T(n) = Θ(f(n))
```

Interpretazione: Il lavoro è concentrato nella radice

PROCEDURA DI RISOLUZIONE

Step 1: Identifica parametri

- Estrai **a**, **b**, **f(n)** dalla ricorrenza
- Calcola **n^(log_b a)**

Step 2: Confronta f(n) con n^(log_b a)

- Calcola limite: lim (n→∞) f(n) / n^(log_b a)
- Risultato determina il caso

Step 3: Verifica condizioni

- **Caso 1**: Verifica esistenza ε > 0
- **Caso 2**: Identifica k (esponente del log)
- **Caso 3**: Verifica condizione di regolarità

Step 4: Applica soluzione

- Scrivi risultato corrispondente al caso identificato

ESEMPI CASO 1: f(n) < n^(log_b a)

Esempio 1.1 - MergeSort modificato

Ricorrenza: T(n) = 4T(n/2) + n

Soluzione:

```
a = 4, b = 2, f(n) = n
n^(log_b a) = n^(log_2 4) = n^2

f(n) = n = O(n^(2-ε)) con ε = 1
```

```
Caso 1 → T(n) = Θ(n^2)
```

Esempio 1.2 - Ricorrenza con log

Ricorrenza: T(n) = 4T(n/2) + n log n

Soluzione:

```
a = 4, b = 2, f(n) = n log n
n^(log_b a) = n^2

f(n) = n log n = O(n^(2-ε)) con ε = 0.5

Motivo: n log n cresce più lentamente di n^1.5

Caso 1 → T(n) = Θ(n^2)
```

Esempio 1.3 - Molti sottoproblemi

Ricorrenza: T(n) = 8T(n/2) + n^2

Soluzione:

```
a = 8, b = 2, f(n) = n^2
n^(log_b a) = n^(log_2 8) = n^3

f(n) = n^2 = O(n^(3-ε)) con ε = 1

Caso 1 → T(n) = Θ(n^3)
```

ESEMPI CASO 2: f(n) = n^(log_b a) · log^k n

Esempio 2.1 - MergeSort classico

Ricorrenza: T(n) = 2T(n/2) + n

Soluzione:

```
a = 2, b = 2, f(n) = n
n^(log_b a) = n^(log_2 2) = n^1 = n
```

```
f(n) = n = θ(n · log^0 n) con k = 0

Caso 2 → T(n) = θ(n^1 · log^(0+1) n) = θ(n log n)
```

Esempio 2.2 - Ricerca binaria

Ricorrenza: T(n) = T(n/2) + 1

Soluzione:

```
a = 1, b = 2, f(n) = 1
n^(log_b a) = n^(log_2 1) = n^0 = 1

f(n) = 1 = θ(1) = θ(n^0 · log^0 n) con k = 0

Caso 2 → T(n) = θ(n^0 · log^1 n) = θ(log n)
```

Esempio 2.3 - Caso con log n

Ricorrenza: T(n) = 2T(n/2) + n log n

Soluzione:

```
a = 2, b = 2, f(n) = n log n
n^(log_b a) = n

f(n) = n log n = θ(n · log^1 n) con k = 1

Caso 2 → T(n) = θ(n · log^2 n)
```

ESEMPI CASO 3: f(n) > n^(log_b a)

Esempio 3.1 - Lavoro alla radice dominante

Ricorrenza: T(n) = 2T(n/2) + n^2

Soluzione:

```
a = 2, b = 2, f(n) = n^2
n^(log_b a) = n
```

```
f(n) = n^2 = Ω(n^(1+ε)) con ε = 1

Verifica regolarità:
a·f(n/b) = 2·(n/2)^2 = 2·n^2/4 = n^2/2 ≤ c·n^2 con c = 1/2 < 1 ✓

Caso 3 → T(n) = Θ(n^2)
```

Esempio 3.2 - Sottoproblema piccolo

Ricorrenza: T(n) = T(n/5) + n

Soluzione:

```
a = 1, b = 5, f(n) = n
n^(log_b a) = n^(log_5 1) = n^0 = 1

f(n) = n = Ω(n^(0+ε)) con ε = 1

Verifica regolarità:
a·f(n/b) = 1·(n/5) = n/5 ≤ c·n con c = 1/5 < 1 ✓

Caso 3 → T(n) = Θ(n)
```

Esempio 3.3 - Radice pesante con log

Ricorrenza: T(n) = 3T(n/2) + n(n+1)

Soluzione:

```
a = 3, b = 2, f(n) = n^2 + n ≈ n^2
n^(log_b a) = n^(log_2 3) ≈ n^1.585

f(n) = n^2 = Ω(n^(1.585+ε)) con ε = 0.4

Verifica regolarità:
a·f(n/b) = 3·(n/2)^2 = 3n^2/4 ≤ c·n^2 con c = 3/4 < 1 ✓

Caso 3 → T(n) = Θ(n^2)
```

ESEMPI RICORRENZE NON STANDARD

Esempio 4.1 - Riduzione non costante

Ricorrenza: T(n) = T(n/2) + T(√n) + n

Nota: Master Theorem NON si applica direttamente (due sottoproblemi diversi)

Soluzione alternativa (substitution):

```
Ipotesi: T(n) = O(n)

T(n) = T(n/2) + T(√n) + n
     ≤ c(n/2) + c√n + n
     = cn/2 + c√n + n
     ≤ cn  (per c ≥ 2 e n sufficientemente grande)

Quindi T(n) = Θ(n)
```

Esempio 4.2 - Coefficiente variabile

Ricorrenza: T(n) = 2/3 · T(n-1) + 2n

Nota: Non forma T(n) = aT(n/b) + f(n)

Soluzione (substitution):

```
Ipotesi: T(n) = O(n)

T(n) = (2/3)T(n-1) + 2n
     ≤ (2/3)c(n-1) + 2n
     = (2c/3)n - 2c/3 + 2n
     ≤ cn  se c ≥ 6

Quindi T(n) = Θ(n)
```

Esempio 4.3 - Somma su più divisioni

Ricorrenza: T(n) = 3T(n/5) + T(n/6) + n

Nota: Master Theorem non si applica direttamente

Analisi:

```
Parte divide et impera:
- 3T(n/5): domina rispetto a T(n/6)
```

```
- Per semplicità: T(n) ≈ 3T(n/5) + n

Applicando Master Theorem approssimato:
a = 3, b = 5, f(n) = n
n^(log_5 3) ≈ n^0.68

f(n) = n > n^0.68 → Caso 3
T(n) = Θ(n)
```

CASI IN CUI MASTER THEOREM FALLISCE

Caso 1: GAP tra f(n) e n^(log_b a)

Ricorrenza: T(n) = 2T(n/2) + n / log n

```
a = 2, b = 2, f(n) = n/log n
n^(log_b a) = n

f(n) = n/log n < n ma NON polinomialmente minore
(non esiste ε > 0 tale che n/log n = O(n^(1-ε)))

Master Theorem NON si applica
Soluzione: T(n) = Θ(n log log n) (dimostrazione avanzata)
```

Caso 2: Condizione regolarità non soddisfatta

Ricorrenza: T(n) = 2T(n/2) + n log n

```
a = 2, b = 2, f(n) = n log n
n^(log_b a) = n

Sembrerebbe Caso 3: n log n > n
Ma verifica regolarità:
a·f(n/b) = 2·(n/2)·log(n/2)
         = n·log(n/2)
         = n·(log n - log 2)
         = n log n - n log 2

NON esiste c < 1 tale che n log n - n log 2 ≤ c·n log n

Master Theorem NON si applica nel Caso 3 standard
Ma è Caso 2 con k=1: T(n) = Θ(n log^2 n)
```

TABELLA RIEPILOGATIVA CASI COMUNI

Ricorrenza	a	b	n^(log_b a)	f(n)	Caso	Soluzione
T(n) = T(n/2) + 1	1	2	1	1	2 (k=0)	Θ(log n)
T(n) = 2T(n/2) + n	2	2	n	n	2 (k=0)	Θ(n log n)
T(n) = 4T(n/2) + n	4	2	n^2	n	1	Θ(n^2)
T(n) = 2T(n/2) + n^2	2	2	n	n^2	3	Θ(n^2)
T(n) = 3T(n/2) + n	3	2	n^1.585	n	1	Θ(n^1.585)
T(n) = T(n/5) + n	1	5	1	n	3	Θ(n)
T(n) = 8T(n/2) + n^2	8	2	n^3	n^2	1	Θ(n^3)
T(n) = 2T(n/2) + n log n	2	2	n	n log n	2 (k=1)	Θ(n log^2 n)

METODO DI SOSTITUZIONE (quando MT fallisce)

Step 1: Ipotizza forma soluzione

- Basati su intuizione o casi simili
- Esempio: T(n) = O(n) oppure T(n) = O(n log n)

Step 2: Dimostra per induzione

```
Dimostra: T(n) ≤ c·g(n) per n ≥ n_0

Passo base: Verifica per n = n_0

Passo induttivo:
1. Assumi T(k) ≤ c·g(k) per k < n
2. Sostituisci nella ricorrenza
3. Semplifica algebricamente
4. Mostra T(n) ≤ c·g(n) per opportuna scelta di c
```

Esempio completo

Ricorrenza: T(n) = T(n/2) + T(n/3) + n

Ipotesi: T(n) = O(n)

Dimostrazione:

```
Voglio T(n) ≤ cn per c opportuno

T(n) = T(n/2) + T(n/3) + n
     ≤ c(n/2) + c(n/3) + n   [per ipotesi induttiva]
     = cn/2 + cn/3 + n
     = cn(1/2 + 1/3) + n
     = 5cn/6 + n
     ≤ cn   se 5c/6 + 1 ≤ c
            se c/6 ≥ 1
            se c ≥ 6

Quindi T(n) = O(n) con c = 6
```

CHECKLIST ESAME

✓ **Identifica parametri**: estrai a, b, f(n)
✓ **Calcola n^(log_b a)**: usa log properties
✓ **Confronta ordini**: quale domina tra f(n) e n^(log_b a)?
✓ **Caso 1**: f(n) più piccolo → soluzione n^(log_b a)
✓ **Caso 2**: f(n) uguale (con log) → aggiungi log al risultato
✓ **Caso 3**: f(n) più grande → verifica regolarità → soluzione f(n)
✓ **MT fallisce**: usa substitution method

ERRORI COMUNI DA EVITARE

❌ Confondere log_b a con a/b
❌ Non verificare condizione regolarità nel Caso 3
❌ Applicare MT quando ricorrenza non è forma standard
❌ Dimenticare fattore logaritmico nel Caso 2
❌ Non considerare che Caso 1 richiede gap POLINOMIALE
❌ Pensare f(n) = n log n sia Caso 1 per T(n) = 2T(n/2) + n log n
(è Caso 2 con k=1!)

TRUCCHI E SUGGERIMENTI

Trick 1: Calcolo veloce log_b a

```
log_b a = log_2 a / log_2 b

Esempi:
```

```
- log_2 4 = 2
- log_2 8 = 3
- log_3 9 = 2
- log_4 16 = 2
```

Trick 2: Riconoscimento rapido casi

```
Se a = b^k → log_b a = k → confronta f(n) con n^k

Esempio: T(n) = 4T(n/2) + ...
        a = 4 = 2^2 → confronta f(n) con n^2
```

Trick 3: Caso 2 è il più frequente

Molti algoritmi divide et impera bilanciano perfettamente lavoro tra livelli → Caso 2

FORMULARIO RAPIDO

Caso 1: f(n) < n^(log_b a) → **T(n) = Θ(n^(log_b a))**
Caso 2: f(n) = Θ(n^(log_b a) log^k n) → **T(n) = Θ(n^(log_b a) log^(k+1) n)**
Caso 3: f(n) > n^(log_b a) + regolarità → **T(n) = Θ(f(n))**

Logaritmi utili:

- log_2 1 = 0
- log_2 2 = 1
- log_2 4 = 2
- log_2 8 = 3
- log_3 9 = 2
- log_4 16 = 2