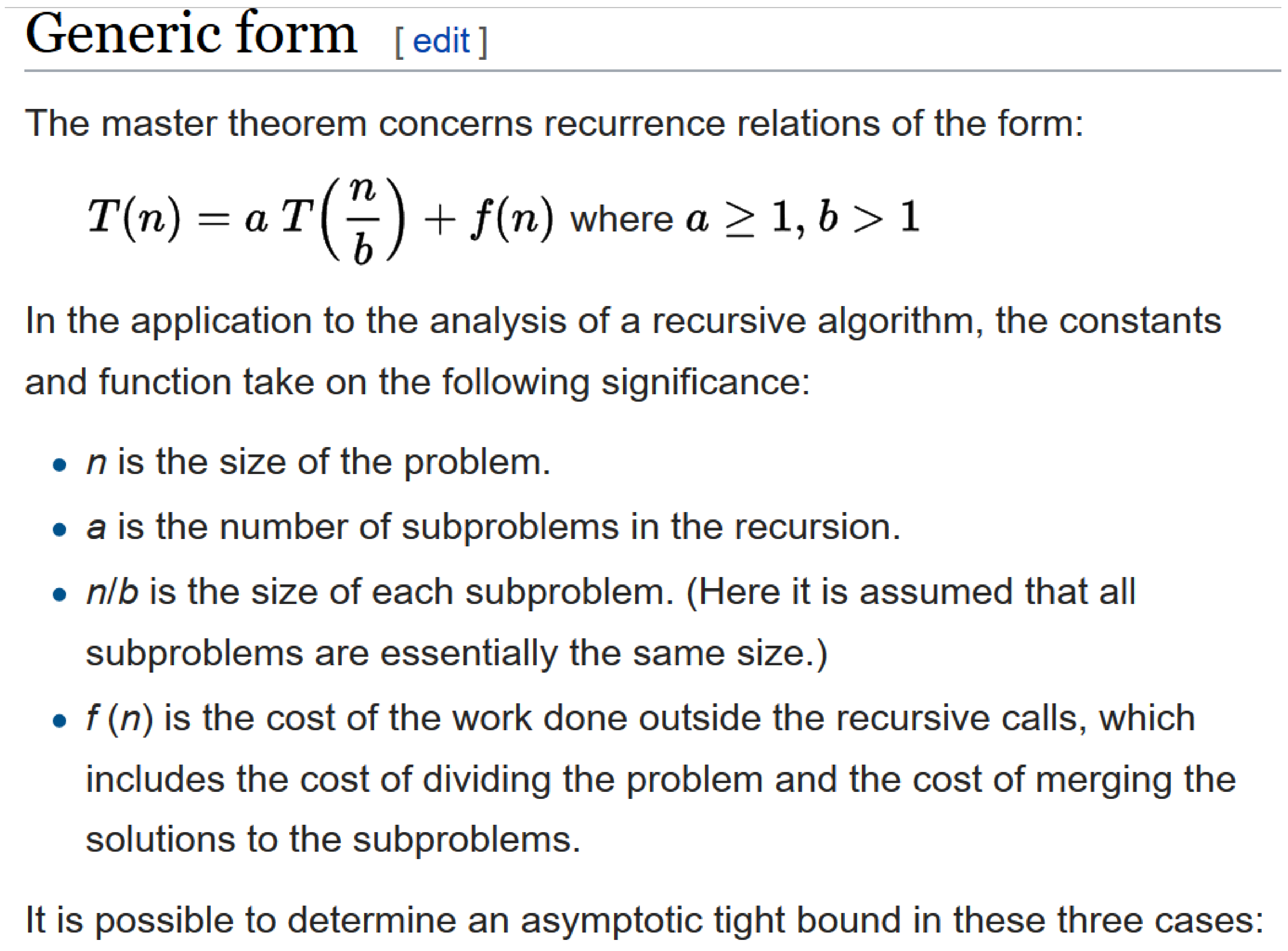
Master Teoremi



Generic form [edit]

The master theorem concerns recurrence relations of the form:

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n) \text{ where } a \geq 1,\, b > 1$$

In the application to the analysis of a recursive algorithm, the constants and function take on the following significance:

- *n* is the size of the problem.
- *a* is the number of subproblems in the recursion.
- *n*/*b* is the size of each subproblem. (Here it is assumed that all subproblems are essentially the same size.)
- *f* (*n*) is the cost of the work done outside the recursive calls, which includes the cost of dividing the problem and the cost of merging the solutions to the subproblems.

It is possible to determine an asymptotic tight bound in these three cases:

Master Teoremi

Case 1 [edit]

Generic form [edit]

If $f(n) = O\left(n^c\right)$ where $c < \log_b a$ (using big O notation)

then:

$$T(n) = \Theta\left(n^{\log_b a}\right)$$

Example [edit]

$$T(n) = 8T\left(\frac{n}{2}\right) + 1000n^2$$

As one can see from the formula above:

$a = 8,\ b = 2,\ f(n) = 1000n^2$, so

$f(n) = O\left(n^c\right)$, where $c = 2$

Next, we see if we satisfy the case 1 condition:

$$\log_b a = \log_2 8 = 3 > c.$$

It follows from the first case of the master theorem that

$$T(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^3\right)$$

(indeed, the exact solution of the recurrence relation is $T(n) = 1001n^3 - 1000n^2$, assuming $T(1) = 1$).

Master Teoremi

Case 2 [edit]

Generic form [edit]

If it is true, for some constant $k \geq 0$, that:

$$f(n) = \Theta\left(n^c \log^k n\right) \text{ where } c = \log_b a$$

then:

$$T(n) = \Theta\left(n^c \log^{k+1} n\right)$$

Example [edit]

$$T(n) = 2T\left(\frac{n}{2}\right) + 10n$$

As we can see in the formula above the variables get the following values:

$$a = 2,\ b = 2,\ c = 1,\ f(n) = 10n$$

$$f(n) = \Theta\left(n^c \log^k n\right) \text{ where } c = 1, k = 0$$

Next, we see if we satisfy the case 2 condition:

$$\log_b a = \log_2 2 = 1, \text{ and therefore, } c = \log_b a$$

So it follows from the second case of the master theorem:

$$T(n) = \Theta\left(n^{\log_b a} \log^{k+1} n\right) = \Theta\left(n^1 \log^1 n\right) = \Theta\left(n \log n\right)$$

Thus the given recurrence relation $T(n)$ was in $\Theta(n \log n)$.

(This result is confirmed by the exact solution of the recurrence relation, which is $T(n) = n + 10n\log_2 n$, assuming $T(1) = 1$.)

Master Teoremi

Case 3 [edit]

Generic form [edit]

If it is true that:

$$f(n) = \Omega\left(n^{c}\right) \text{ where } c > \log_b a$$

and if it is also true that:

$af\left(\frac{n}{b}\right) \leq kf(n)$ for some constant $k < 1$ and sufficiently large n (often called the *regularity condition*)

then:

$$T\left(n\right) = \Theta\left(f(n)\right)$$

Example [edit]

$$T(n) = 2T\left(\frac{n}{2}\right) + n^2$$

As we can see in the formula above the variables get the following values:

$$a = 2,\ b = 2,\ f(n) = n^2$$
$$f(n) = \Omega\left(n^c\right), \text{ where } c = 2$$

Next, we see if we satisfy the case 3 condition:

$$\log_b a = \log_2 2 = 1, \text{ and therefore, yes, } c > \log_b a$$

The regularity condition also holds:

$$2\left(\frac{n^2}{4}\right) \leq kn^2, \text{ choosing } k = 1/2$$

So it follows from the third case of the master theorem:

$$T\left(n\right) = \Theta\left(f(n)\right) = \Theta\left(n^2\right).$$

Thus the given recurrence relation $T(n)$ was in $\Theta(n^2)$, that complies with the $f(n)$ of the original formula.

(This result is confirmed by the exact solution of the recurrence relation, which is $T(n) = 2n^2 - n$, assuming $T(1) = 1$.)

Example: merge sort

1. ***Divide:*** Trivial.
2. ***Conquer:*** Recursively sort 2 subarrays.
3. ***Combine:*** Linear-time merge.

$$T(n) = \underbrace{2}_{\textit{\# subproblems}} T(\underbrace{n/2}_{\textit{subproblem size}}) + \underbrace{O(n)}_{\textit{work dividing and combining}}$$

$$n^{\log_b a} = n^{\log_2 2} = n^1 = \mathrm{n} \Rightarrow \text{CASE } 2 \ (k = 0)$$

$$\Rightarrow \ T(n) = \Theta(n \log n) \ .$$

örnekler

1. $T(n) = 3T(n/2) + n^2 \implies T(n) = \Theta(n^2)$ (Case 3)

2. $T(n) = 4T(n/2) + n^2 \implies T(n) = \Theta(n^2 \log n)$ (Case 2)

3. $T(n) = T(n/2) + 2^n \implies \Theta(2^n)$ (Case 3)

4. $T(n) = 2^n T(n/2) + n^n \implies$ Does not apply (a is not constant)

5. $T(n) = 16T(n/4) + n \implies T(n) = \Theta(n^2)$ (Case 1)

6. $T(n) = 2T(n/2) + n \log n \implies T(n) = n \log^2 n$ (Case 2)

7. $T(n) = 2T(n/2) + n/\log n \implies$ Does not apply (non-polynomial difference between $f(n)$ and $n^{\log_b a}$)

8. $T(n) = 2T(n/4) + n^{0.51} \implies T(n) = \Theta(n^{0.51})$ (Case 3)

9. $T(n) = 0.5T(n/2) + 1/n \implies$ Does not apply ($a < 1$)

10. $T(n) = 16T(n/4) + n! \implies T(n) = \Theta(n!)$ (Case 3)

11. $T(n) = \sqrt{2}T(n/2) + \log n \Longrightarrow T(n) = \Theta(\sqrt{n})$ (Case 1)

12. $T(n) = 3T(n/2) + n \Longrightarrow T(n) = \Theta(n^{\lg 3})$ (Case 1)

13. $T(n) = 3T(n/3) + \sqrt{n} \Longrightarrow T(n) = \Theta(n)$ (Case 1)

14. $T(n) = 4T(n/2) + cn \Longrightarrow T(n) = \Theta(n^2)$ (Case 1)

15. $T(n) = 3T(n/4) + n \log n \Longrightarrow T(n) = \Theta(n \log n)$ (Case 3)

16. $T(n) = 3T(n/3) + n/2 \Longrightarrow T(n) = \Theta(n \log n)$ (Case 2)

17. $T(n) = 6T(n/3) + n^2 \log n \Longrightarrow T(n) = \Theta(n^2 \log n)$ (Case 3)

18. $T(n) = 4T(n/2) + n/\log n \Longrightarrow T(n) = \Theta(n^2)$ (Case 1)

19. $T(n) = 64T(n/8) - n^2 \log n \Longrightarrow$ Does not apply ($f(n)$ is not positive)

20. $T(n) = 7T(n/3) + n^2 \Longrightarrow T(n) = \Theta(n^2)$ (Case 3)

21. $T(n) = 4T(n/2) + \log n \Longrightarrow T(n) = \Theta(n^2)$ (Case 1)

22. $T(n) = T(n/2) + n(2 - \cos n) \Longrightarrow$ Does not apply. We are in Case 3, but the regularity condition is violated. (Consider $n = 2\pi k$, where k is odd and arbitrarily large. For any such choice of n, you can show that $c \geq 3/2$, thereby violating the regularity condition.)