用性能更好的算法取代低效的算法是唯一确保性能得到数量级改进的方法。

时间成本通常用“大O”表示法来写，如O（f（n）），其中n是输入大小的某个重要方面，f（n）是描述算法执行多少个重要操作来处理大小为n的输入的函数。函数f（n）通常被简化为仅显示其增长最快的项，因为对于大的n值，该项支配f（n）的值。

表格

描述已自动生成

预计算 通过在执行到达热点代码之前（在程序的早期、链接时、编译时或设计时）执行计算，从程序的热点部分移除计算。

延迟计算 通过在接近需要计算的地方执行计算，从某些代码路径中移除计算。

批量处理 一起对多个项目进行计算，而不是一次处理一个项目。

缓存 通过保存和重用昂贵的计算结果，而不是重新计算它们，来减少计算。

特化 通过移除未使用的通用性来减少计算。

加大处理单元 通过一次处理更大组的输入，来减少重复操作的成本。

提示 通过提供可能提高性能的提示来减少计算。

优化预期路径 在运行时按预期频率的降序测试输入或事件。

哈希 计算较大数据结构（如可变长度字符串）的紧凑数值摘要（哈希）。哈希可以代表数据结构进行比较，以提高性能。

双重检查 通过执行一个廉价的检查，必要时再执行一个昂贵的检查，来减少计算。

• 警惕那些声称拥有常数时间算法的陌生人。它们可能是 O(n) 的。

 • 多个高效的算法可以以某种方式组合，使得它们整体运行时间为 O(n^2) 或更差。

• 二分查找的时间复杂度为 O(log2 n)，但它不是最快的搜索算法。插值查找的时间复杂度为 O(log log n)，而哈希是常数时间的。

 • 对于少于 4 个条目的小表，所有搜索算法检查的条目数量大约相同。