|  |  |  |
| --- | --- | --- |
| **Notation** | **Intuition** | **As  n \to \infty, eventually...** |
| f(n) \in O(g(n)) | *f* is bounded above by *g* (up to constant factor) asymptotically | |f(n)|  \leq  g(n)\cdot kfor some *k* |
| f(n) \in \Omega(g(n)) | *f* is bounded below by *g* (up to constant factor) asymptotically | |f(n)|  \geq  g(n)\cdot kfor some *k* |
| f(n) \in \Theta(g(n)) | *f* is bounded both above and below by *g* asymptotically | |g(n)|\cdot k_1 \leq |f(n)| \leq |g(n)|\cdot k_2for some *k*1, *k*2 |
| f(n) \in o(g(n)) | *f* is dominated by *g* asymptotically | |f(n)| \le |g(n)|\cdot \varepsilonfor every \varepsilon |
| f(n) \in \omega(g(n)) | *f* dominates *g* asymptotically | |f(n)| \ge |g(n)|\cdot kfor every *k* |
| f(n)\sim g(n)\! | *f* is equal to *g* asymptotically | f(n)/g(n) \to 1 |

A problem is said to have **optimal substructure** if an optimal solution can be constructed efficiently from optimal solutions to its subproblems.

**Recurrences:** T(n) = a \* T(n / b) + f(n)

Then:

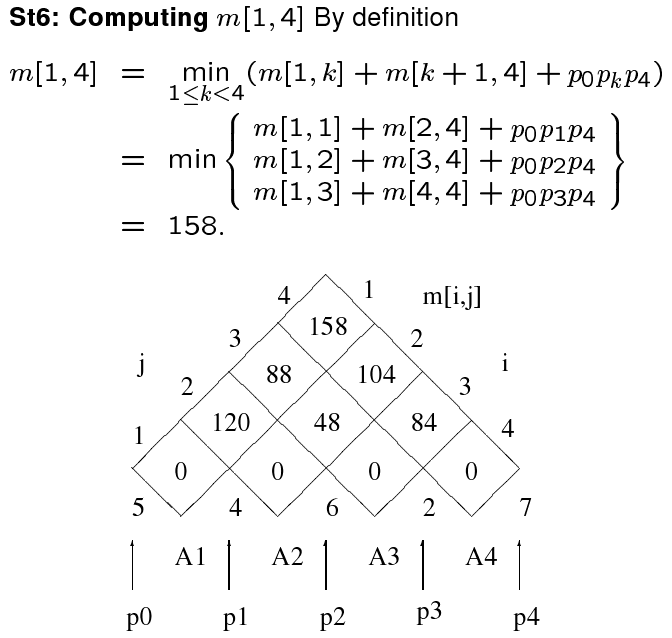
If f(n) = O(nlogb(a-e)) for some e > 0, then T(N) = Θ(nlogb(a))

If f(n) = Θ (nlogb(a)), then T(N) = Θ(nlogb(a)log2(n))

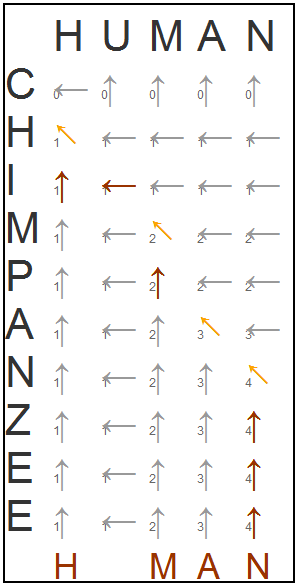
If f(n) = Ω(nlogb(a+e)) for some e > 0, **and** af(n / b) <= cf(n) for some c > 1 for all n > n0 , then T(N) = Θ(f(n))

|  |  |  |
| --- | --- | --- |
| **Category** | **Simple Form** | **Representative** |
| Constant | Constant | 5 |
| Logarithmic | Log(n) aka log2n | Log(n) + 1 |
| Polylog | C \* logi(n) | Log2(n) + 3log(n) - 1 |
| Square Root | Sqrt(n) | Sqrt(n + 3) + log(n) + 1 |
| Linear | N | 5n + 56 |
| NlogN | N \* log(n) | Nlog(n) + 4n - 5 |
| Quadratic | N2 | 3n2 + n + 2log(n) - 4 |
| Cubic | N3 | 5n3 - 5n2 + sqrt(n) + 5log(n) |
| Polynomial | nk | Nk + 1 |
| Factorial | N! | N! + n5 - log(n) |
| Exponential | 2n | 2n + n4 + nlog(n) |

2n > 2log2(n) > nlog(log(n)) > sqrt(n3)log2(n) > log2(nn) > n/log­2(n)



Longest Common Subsequence:



**Graphs:**

**Kruskal's:** Pick smallest weight, then next smallest weight, etc. EASY

**Prim's:** Push starting node into Q. Dequeue next node, put costs of linked nodes into chart. Scan the Q for next smallest weight, and Dequeue that. Keep going. Still easy.