Fundamentals of asymptotic notation

$$o(1) \leq O(1) \subseteq \Theta$$

Limits to prove $f(n) = o(g(n))$

Provide constants to prove $f(n) = \Theta(g(n))$

---

Algorithm Analysis

What does a specific algorithm do?

Analyze the running time of an algorithm

- Counting loops and operations
- Bounding the number of function calls
- Recurrence relations
- BFS
Recurrence relation:

\[ T(n) = \begin{cases} & T(n/2) + n \quad \text{merge sort} \\ & \text{master theorem} \\ & \text{Substitution} \\ & \text{recursion tree} \end{cases} \]

- Amortized analysis
- Prim's algorithm \( O(m \log n) \) amortized using the heap
  potential functions

Prove correctness:
- Termination
  - Usually simple
  - Bounded number of steps
    - e.g., looking at every element of an array
    - Look at any one element at most \( \log n \) times
- Correctness
  - Contradiction
- Contradiction
  - assume simplest case

- induction
  - base case
  - inductive step
  - termination

- Stay ahead arguments (greedy)
  - similar to induction
  - on termination
  - argue your set size is the same length as optimal

- loop invariants
  - form of induction
  - show that an "invariant" holds at every step of a loop

Greedy Algorithms
Greedy Algorithms
make progress with local choices to obtain
global optimum
problems: interval scheduling
   MST - Kruskal, Prim
   union-find, priority queues (e.g., heap)
huffman coding, Dijkstra's algorithm

Graph Exploration
definitions
   vertices, edges, adjacency list
   undirected, directed
proofs: contradiction
BFS/DFS as building blocks
connected components, top sort
   bipartiteness, strongly connected components

Divide and Conquer
recursively break problem into smaller
subproblems, solve, and combine the solutions
   e.g., mergesort, quicksort
   techniques: induction
techniques: induction

recurrence relations

\[ T(n) = aT\left(\frac{n}{b}\right) + \Theta(n) \]

running time \# subproblems size of subproblems

merge sort \[ T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \]

closed form \[ T(n) = \Theta(n \log n) \]

problems: selection, matrix multiplication, closest pair

Dynamic Programming

recursively break into smaller subproblems

techniques: recurrence relations

provide answer, not running time

problems:

chain matrix multiplication, weighted interval scheduling, sequence alignment, shortest paths
shortest paths
how to handle negative weighted
Floyd-Warshall
all-pairs shortest paths