Midterm Review

Tuesday, July 2, 2019 11:38 AM

Fundamentals
asymptotic notation
$o()$ $O()$ Ω Θ

limits to prove $f(n) = o(g(n))$
provide constants to prove $f(n) = O(g(n))$
Ω
Θ

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
  BFS
- recurrence relations

recurrence relations

$T(n) = 2T(n/2) + n$ mergesort

master theorem

substitution

recursion tree

- amortized analysis

Prim's algorithm $O(m + n \log n)$ amortized using thin heap

potential functions

prove correctness

termination

- usually simple
- bounded number of steps

e.g. looking at every element of array

look at any one element at most lgn 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

techniques
· induction
stay-ahead

make progress with local choices to obtain global optimum

problems: interval scheduling

MST - kruskal, prim

union find, priority queues (e.g. thin 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

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

running time | # subproblems | size of subproblems | local work

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

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 weight edges

Floyd-Warshall,

all-pairs shortest paths