# CSE 373: Final Review

Pete Morcos
University of Washington
5/31/00

http://www.cs.washington.edu/education/courses/cse373/00sp

## Overview

- Basic math
  - logs, exponents, summations
  - inductive proofs
- Asymptotic analysis
  - big-oh, big-theta, big-omega
  - the nightmare of exponential algorithms
  - costs of time and space
- Lists, Stacks, Queues
  - details like header nodes, circular or double linking
  - array or pointer implementations



## Overview

- Trees
  - terms, height tends to be logarithmic (if balanced)
- Binary Search Trees
  - how to Find, Insert; bad worst case behavior
    - All operations might cost O(N)
  - AVL trees for maintaining balance
    - No operation costs more than O(log N)
  - Splay trees for good amortized performance
    - One operation might be O(N), but overall they average to O(log N)
  - Idea of “rotation” to rearrange the tree
  - Lazy deletion



## Overview

- Hash tables
  - Collision strategies: chaining, probing
  - Trade space to gain time
- Heaps (Priority Queues)
  - Array implementation
  - BuildHeap can be done in O(N)
- Binomial Queues
  - Merge operation is fast
  - details of Insert and DeleteMin



## Overview

- B-trees
  - node arrangement (internal vs leaf)
  - details of insert and remove
  - good for very large, disk-based trees
- Selection
  - naive selection (scan), quickselect
  - median is hardest to do



## Overview

- Sorting
  - Insertion sort
  - Selection sort
  - Shellsort – a modification of insertion sort
  - Heapsort
  - Mergesort – divide-and-conquer
  - Quicksort – divide-and-conquer
    - fast because partition is in-place and very simple/efficient
    - issues surrounding pivot selection
  - Bucket sort, Radix sort
  - concept of a “stable sort”
- Recurrence relations

## Overview

- Disjoint Set (Union/Find)
  - union-by-xxx
  - path compression
- Graphs
  - adjacency matrix vs. adjacency list
  - terms for types of connectivity
  - Topological Sort
  - BFS, DFS
  - Dijkstra (weighted shortest path) – a greedy algorithm
  - Prim/Kruskal (minimum spanning tree) – greedy
  - Hamiltonian circuit problem – NP completeness
    - Exhaustive search



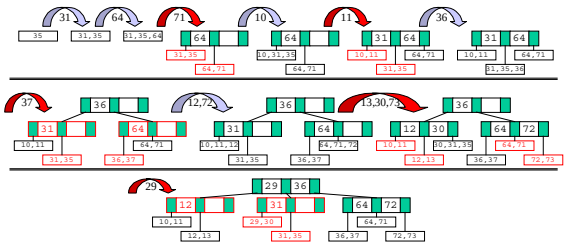
## Overview

- NP-completeness (brief intro)
  - all NPC problems are basically equivalent
  - familiarity with these will help you realize if/when you've run into a hard problem
- Amortization (brief intro)
  - how to use a potential function (if you have one) to compute amortized budgets in general
  - how to amortize the binomial queue operations, specifically
- Algorithmic Techniques
  - Huffman coding – greedy algorithm
  - Closest point – divide-and-conquer



## B-trees

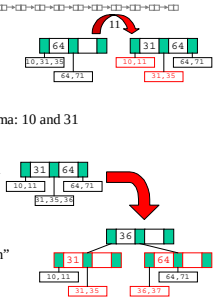
- Let's grow a b-tree one step at a time
  - M=3, so leaves hold 2-3 values, internal nodes have 2-3 children





## B-tree Insertion

- A node overflows:
  - Create new leaf node
    - Divide values evenly (10,11 and 31,35)
  - Tell parent, "You now have 2 children"
    - Parent also needs to know the new minima: 10 and 31
- Parent accepts new child:
  - If room, reshuffle pointers and add child
    - Minima in first example are 10, 31, 64
  - If parent is full, split into two
    - Divide the M+1 children evenly
    - Tell *its* parent, "you now have 2 children"
- If there's no parent, create new root





## B-tree Removal

- Remove item from leaf:
  - If leaf underflows, tell parent
- Parent handles small child:
  - Try to borrow from child's siblings
  - Else, delete child and distribute items to child's siblings
- In example:
  - Can't borrow from [29,30]
    - So remove [31] and add to sibling
  - Intermediate has only 1 child
    - Neither sib can lend a child
    - So remove intermediate, and give its child to a sibling