CMPT307: Tutorial 2

Week 14-1

Xian Qiu

Simon Fraser University

xianq@sfu.ca

Data Structures

- dynamic sets:
  - linked lists
  - hash tables: hashing with chaining and open addressing
  - binary search trees: rotations
  - red-black trees
- binary heap/priority queues: heap-sort, MST, shortest paths
- disjoint sets: connected components, testing cycles, MST
- amortized analysis
  - aggregate analysis
  - accounting method
  - potential function method

Sorting/Order Statistics

- insertion sort: running time is not asymptotically optimal
- merge sort: divide-and-conquer, not in place
- heap sort: binary heap data structure
- quick sort: divide-and-conquer, asymptotically optimal on average
- lower bound of comparison sort
- radix sort: beat $O(n \log n)$ for k constant (using stable sort)
- order statistics: divide-and-conquer

Graph Problems

- elementary graph algorithms
  - BFS: unweighted shortest path, matching
  - DFS: topological sorting, strongly connected components
- MST: greedy (graph cut)
- shortest paths: RELAX, dynamic programming
- maximum bipartite matching: augmenting paths (optimal condition)

Algorithm Design

- divide-and-conquer: sorting, order statistics, maximum subarray, (square) matrix multiplication
- dynamic programming: rod cutting, matrix-chain multiplication, longest common subsequence, optimal binary search tree
- greedy algorithm: activity selection, Huffman code, MST, matroids

Complexity

- polynomial time algorithms
- complexity classes: $\mathcal{P}, \mathcal{NP}, \text{co-}\mathcal{NP}$ and NP-completeness
- polynomial time reductions
- basic NP-complete problems
- $\mathcal{P} = \mathcal{NP}$? open, but most believe not

- $\mathbb{A}$ returns a feasible solution in polynomial time
- no need to prove optimality, feasibility is often easy
- often easy to show that the running time is polynomial
- want approximation ratio as close to lower bound as possible
  important and often hard

About the Final Exam

COURSE	DAY	DATE	TIME	ROOM	TYPE
cmpt 307-d300	Friday	Dec 16	08:30-11:30	SWH10081	closed exam

- final exam procedures
- office hours on Dec 12, starting from 13:00
  - if no students come, it ends at 14:00
  - other time by appointment
- coverage: all lecture contents (chapters/sections not covered by lectures are not required)

Q. Types

- running time analysis
- data structures
- algorithm design
- complexity

Running Time Analysis

- amortized analysis: analyze the amortized cost
- worst-case: may need to use the amortized cost
- average-case
  1. make reasonable assumption on input (often uniform distribution)
  2. define suitable indicator random variable X_i, $P(X_i = 1) =?$
  3. represent the running time T as a function of X_i, say $T = f(X_1, X_2, ...)$
  4. compute $\mathbb{E}[T]$

Data Structures

- ▷ worst-case running time
- ▷ average-case running time (probabilistic analysis)
- ▷ amortized costs
- ▷ (balanced) binary search trees: rotations
- ▷ heaps/priority queues
- ▷ disjoint sets

Algorithm Design

- divide-and-conquer:
    1. describe an algorithm (or write pseudocode)
    2. the correctness is often straightforward, but you must analyze the running time
    3. for recurrences, you should be able to use the three approaches (master theorem will be presented as appendix to you)
- dynamic programming
    1. derive a recursive formula (including boundary conditions)
    2. optimal substructure must be formally proved (need verify feasibility when using cut-and-paste argument)
    3. analyze the running time based on the recursive formula
    4. no need to write pseudocode or to construct optimal solution if not indicated

Algorithm Design

- greedy algorithm (matroids)
  1. describe a greedy strategy and analyze the running time (often straightforward)
  2. prove the correctness (important)
  3. you may also be asked to formulate the problem as a standard matroid maximization problem, so you also need to prove that the system is a matroid
- approximation algorithm
  1. describe an algorithm and analyze the running time
  2. analyze the approximation ratio (important)

- $L \in \mathcal{NP}$ or co-$\mathcal{NP}$? short and checkable certificate
- show that decision problem B is NP-complete
  1. show that $B \in \mathcal{NP}$
  2. you may use any known NP-complete problem A and show that $A \leq_{\mathsf{P}} B$: must check that the reduction is correct and can be done in polynomial time