Algorithms 2018/2019 (CC4010) – December 11th, 2018

Test #2 - Auxiliary Material

Linear Algorithms for the Selection Problem

- **Selection Problem**: determining the k-th smallest item of a set
- **QuickSelect:** randomized algorithm for selection problem with linear expected running time
- **Median of Medians:** (recursive) deterministic algorithm for selection problem with linear worst case time

---

Sorting in Linear Time

- **CountingSort:** compute the frequency of each element; use frequency array to produce the output
- **RadixSort:** use stable sort algorithm to sort each "digit", starting from the lowest significance one
- **BucketSort:** use buckets to store the elements on equally sized intervals/ranges; sort each bucket independently and concatenate the results

---

String Matching

- **String Matching:** find all occurrences of a pattern P in text a T
- **Naive String Matching:** brute force algorithm trying all possible shifts of a pattern
- **Automaton for String Matching:** use DFA with $|P|$ states and $|alphabet|$ transitions in each state
- **Knuth Morris Pratt Algorithm:** use π function (longgest proper prefix which is also a proper suffix) to skip many characters when there is a mismatch
- **Rabin-Karp Algorithm:** use rolling hash function as heuristic to skip some possible shifts
- **Trie:** prefix tree representing set of words
- **Suffix Tree:** stores all suffixes of a given word
- **Suffix Array:** sorted array of all suffixes of a given word
- **LCP Array:** stores longest common prefix between pairs of consecutive suffixes

---

Greedy Algorithms and Dynamic Programming

- **Optimization Problem:** when want to find the best solution according to e certain criteria (ex: max or min)
- **Greedy Algorithm:** follows the problem solving heuristic of making the locally optimal solution
- **Optimal Substructure:** when the optimal solution contains in itself optimal solutions for subproblems
- **Greedy Choice Property:** when the optimal solution is consistent with the choice of the greedy algorithm
- **Dynamic Programming (DP):** algorithmic technique based on storing the solutions of subproblems instead of recomputing them
- **Overlapping Subproblems:** when there are many equal subproblems (the search space is "small")

NP-completeness

- **Decision Problem:** problems in which the answer is boolean: YES or NO
- **P:** set of decision problems that can be solved in polynomial time
- **NP:** set of decision problems for which if the answer is YES thn there is a proof of that that can be checked in polynomial time
- **coNP:** set of decision problems for which if the answer is NO then there is a proof of that that can be checked in polynomial time
- **NP-hard:** set of problems that are as *hard* as *any* NP problem
- **NP-complete:** set of problems which are both NP and NP-hard
- Problem A is **poly-time reducible** to problem B (written as $A \leq_p B$) if we can solve problem A in polynomial time given a black-box algorithm for problem B.