COMP3121/9101
Algorithm Design

Tutorial 8
String Matching, Linear Programming

Before the Tutorial

Before coming to the tutorial, try and answer these discussion questions yourself to get a firm understanding of how you're pacing in the course. No solutions to these discussion points will be officially released, but feel free to discuss your thoughts on the forum. Your tutor will give you some comments on your understanding.

- Briefly explain how the Rabin-Karp and Knuth-Morris-Pratt algorithms perform string matching.
- What is the difference between linear programming and *integer* linear programming? Which one can we solve in polynomial time?
- Read through the problem prompts and think about how you would approach the problems.

Tutorial Problems

Problem 1. In this problem, we study the algorithm of Rabin-Karp. Consider the string $S = abccbbcabc$ over the alphabet $\Sigma = \{a, b, c\}$. We use the Rabin-Karp algorithm to determine whether S contains the substring $S' = bca$.

(a) Since the size of the alphabet is 3, we set $d = 3$ and choose our prime to be $p = 5$. Using these values for d and p, convert S and S' into their equivalent base 3 numbers.

(b) Determine the hash values for each substring of S of length $|S'|$. Explain how the algorithm would determine whether S contains the substring S'.

(c) How many false positives do we obtain? *A false positive match occurs when the hash values match but the strings do not match.*

Problem 2. In this problem, we study the algorithm of Knuth-Morris-Pratt. Again, consider the string $S = abccbbcabc$ over the alphabet $\Sigma = \{a, b, c\}$. We perform the same string matching problem as problem 1 by determining whether S contains the substring $S' = bca$.

(a) Construct a finite automaton for S'.

- It might help to visually represent each state of the automaton as the *length* of the prefix of S' that has already been matched. What do each of the transitions of the finite automaton represent?

(b) What is the transition table for the above automaton?

(c) Using the finite automaton described in the previous parts, explain how the algorithm would determine whether S contains the substring S'.

(d) Slightly modify the above algorithm to count the number of occurrences that substring S' would appear in S.

Problem 3. We define the weighted vertex cover problem as follows:

Weighted Vertex Cover:

Instance: A graph $G = (V, E)$ and a weight function $w : V \to \mathbb{R}^+$.
Question: Find a subset $V' \subseteq V$ such that each edge $e \in E$ has an endpoint in V' and $\sum_{v \in V'} w(v)$ is minimised.

To formulate a problem as a linear program, we need to establish an appropriate characterisation of the variables and constraints.

(a) What could the variables be?

(b) Using the variables above, what is a suitable objective function?

Hint. *What are you optimising?*

(c) The main constraint that we want to model is that each edge (u, v) must have an endpoint in V'. Using the variables from the previous part, construct a linear equation to ensure this constraint is satisfied.

(d) Hence, formulate the *Weighted Vertex Cover* problem as an *integer* linear program.

Note. *You would not be able to solve this integer linear program in polynomial time. You'd need to relax the integrality constraint.*

After the Tutorial

After your allocated tutorial (or after having done the tutorial problems), review the discussion points. Reflect on how your understanding has changed (if at all).

- In your own time, try and attempt some of the practice problems marked [K] for further practice. Attempt the [H] problems once you're comfortable with the [K] problems. All practice problems will contain fully-written solutions.
- How can you turn a max flow problem into a linear program? How could you model the flow capacity constraints with a linear programming constraint? What is the objective function of the maximum flow problem?