Advanced Algorithms and Datastructures

Exam Notes

Author:
Jenny-Margrethe Vej – 250986 – rwj935

Block 4, April-June, 2014

Contents

1 Maximum Flow

Wuuuh, Max Flow!! You Rock! Kick some ass!

1.1 Flow Networks

1.1.1 Flow Networks and Flow

Let $G = (V, E)$ be a flow network with a capacity function c. Let s be the source of the network, and let t be the sink. A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

Capacity Constraints: For all $u, v \in V$, we require $0 \leq f(u, v) \leq c(u, v)$
Flow Conservation: For all $u \in V - \{s, t\}$, we require

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v) \tag{1}$$

When $(u, v) \notin E$, there can be no flow from u to v, and $f(u, v) = 0$. The value $|f|$ of a flow f is defined as

$$|f| = \sum_{v \in V} f(s, u) - \sum_{v \in V} f(v, s) \tag{2}$$

1.1.2 An example of a flow

For example the transporting example from the book. A firm wants to transport goods from 1 place to anther, using a third part driver.

Guessing this particular section is not thaaaaat important for the exam. ;o)

1.1.3 Modelling problems with antiparallel edges

Antiparallel edges is 2 edges going to/from (v, u) - so 2 edges between 2 vertices but with opposite directions. To come around that, we transform our network into an equivalent one containing no antiparallel edges (adding an extra vertex for that, so we can split one of the edges). The resulting network is equivalent to the original one, due to the fact, that you do not add or subtract anything from the

capacity. It is the same:

1.1.4 Networks with multiple sources and sinks

A maximum flow problem may have several sources and sinks, rather than just one of each. To fix that, we just add a supersource and a supersink with infinity capacity from s to each of the multiple sources.

1.2 The Ford-Fulkerson Method

1: FORD-FULKERSON-METHOD(G, s, t)
2: initialise flow f to 0
3: **while** there exists an augmenting path p in the residual network G_f **do**
4: augment flow f along p
5: **end while**
6: **return** f

1.2.1 Residual Network

Suppose that we have a flow network $G = (V, E)$ with source s and sink t. Let f be a flow in G, and consider a pair of vertices $u, v \in V$. We define the residual capacity $c_f(u, v)$ by

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E, \\ f(v,u) & \text{if } (v,u) \in E, \\ 0 & \text{otherwise} \end{cases}$$

2 Fibonacci Heaps

3 NP-Completeness

4 Randomised Algorithms

5 Hashing

OMG LOLZ'n SHIZZLES! You'r gonna nail this shit!

5.1 Direct-address tables

We have a scenario where we want to maintain a dynamic set of some sort. Each element has a key drawn from a universe $U = \{0, 1, ..., m-1\}$ where m is not too large, and no two elements have the same key.

Representing it by a direct-address table, or array $T[0...m-1]$, each slot or position corresponds to a key in U. If there is an element x with the key k, then $T[k]$ contains a pointer to x - otherwise $T[k]$ is empty represented by NIL

Illustration in the book on page 254

1: DIRECT-ADDRESS-SEARCH(T, k)
2: **return** $T[k]$
3: DIRECT-ADDRESS-INSERTT, x
4: **return** $T[key[x]] \leftarrow x$
5: DIRECT-ADDRESS-DELETET, x
6: **return** $T[key[x]] \leftarrow NIL$

5.2 Hash tables

5.3 Hash functions

5.3.1 The division method

5.3.2 The multiplication method

5.3.3 Universal hashing

5.4 Open addressing

5.5 Perfect hashing

6 Exact Exponential Algorithms

7 Approximation Algorithms

8 Computational Geometry

9 Linear Programming and Optimisation