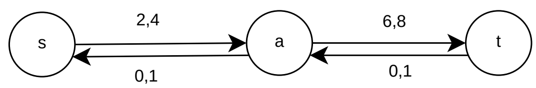
MTAT.03.286: Advanced Methods in Algorithms **University of Tartu**

Homework 3

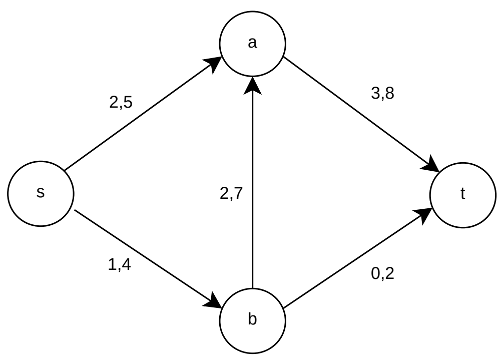
Due date: November 12, 2014

---

1. (a) Is there a legal flow from s to t in the following network with upper and lower bounds? Justify your answer.



   (b) Find a legal flow from s to t in the following network with upper and lower bounds. (You don't have to specificy all the steps in Ford-Flukerson or Dinitz algorithm that you are using, but you have to show the reduction and the resulting flow.)



   (c) Find a maximum flow in the network in part (b).

2. By using the Fast Fourier Transform (FFT) algorithm, evaluate the polynomial $A(x) = x^5 + 2x^3 + x^2 - 1$ at the complex 6th roots of unity. Show at least one level of recursion.

3. Let $A(x) = 2x + 2$ and $B(x) = x^2 - 3x + 1$. In this question, we will compute the polynomial $C(x) = A(x) \cdot B(x)$ by using the FFT algorithm.

   (a) What is the minimum number of points we need to use? Explain.

   (b) Evaluate $A(x)$ at the complex 4th roots of unity. Show at least one level of recursion.

   (c) Evaluate $B(x)$ at the complex 4th roots of unity. Show at least one level of recursion.

   (d) Compute $C(x)$ at the complex 4th roots of unity.

   (e) Find the coefficients of $C(x)$.

4. Let a and b be two n-bit nonnegative integers, and let

$$(a_{n-1}, a_{n-2}, \cdots, a_1, a_0) \qquad \text{and} \qquad (b_{n-1}, b_{n-2}, \cdots, b_1, b_0)$$

   be their respective binary representations. Define integer $c = a \cdot b$. In this question we are interested in algorithm for computing c.

(a) Define polynomials

$$A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0$$

and

$$B(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_1 x + b_0 \ .$$

Show that $a = A(2)$ and $b = B(2)$.

(b) Define polynomial $C(x) = A(x) \cdot B(x)$. (Note that the polynomial $C(x)$ can be computed from $A(x)$ and $B(x)$ using FFT algorithm.) Is $C(2) = c$? Prove or disprove.

(c) What is the degree m of $C(x)$?

(d) Observe that the coefficients of $C(x)$ are not neccessarily all 0 and 1. Propose an algorithm that computes the binary representation of c, namely

$$(c_m, c_{m-1}, \cdots, c_1, c_0) \ ,$$

from the coefficients of $C(x)$ using only $O(n)$ arithmetic operations over integers with $O(\log n)$ bits.