École Polytechnique Fédérale de Lausanne
Prof. Eisenbrand

February 22, 2018

Discrete Optimization (Spring 2018)

Assignment 1

Problem 8 can be **submitted** until March 2 12:00 noon, please send the source code in C++ to `igor.malinovic@epfl.ch` . You are allowed to submit your solutions in groups of at most three students.

Problem 1

Provide a certificate (as in Theorem 0.1 in the lecture notes) of the unsolvability of the linear equation

$$\begin{pmatrix} 2 & 1 & 0 \\ 5 & 4 & 1 \\ 7 & 5 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$$

Solution:

Applying Gaussian elimination we get

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ -5/2 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix}, A' = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 3/2 & 1 \\ 0 & 0 & 0 \end{pmatrix}, b' = \begin{pmatrix} 1 \\ -1/2 \\ 1 \end{pmatrix},$$

hence the third row of Q is our certificate: $q = \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}^\top$.

Problem 2

Show the "if" direction of the Farkas' lemma: given $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$, if there exists a $\lambda \in \mathbb{R}^m_{\geqslant 0}$ such that $\lambda^\top A = 0$ and $\lambda^\top b = -1$, then the system $Ax \leq b$ is unfeasible.

Solution:
Suppose that there exists $x^* \in \mathbb{R}^n$ such that $Ax^* \leq b$. Then, since $\lambda \geq 0$, we have:

$$\lambda^\top A x^* \leq \lambda^\top b \implies 0 \leq -1,$$

a contradiction.

Problem 3

Consider the following linear program:

$$\begin{array}{lrcrcl} \max & x & + & y & & \\ \text{s.t.} & 3x & + & 2y & \leq & 6 \\ & x & + & 4y & \leq & 4. \end{array}$$

The solution $(x, y) = (8/5, 3/5)$ satisfies the both constraints and has the objective value $11/5$. Provide a certificate that this is an optimal solution.

Solution:
The desired certificate is $(3/10, 1/10)$. Summing up the two constraints above with multipliers $3/10$ and $1/10$ respectively, one obtains that any feasible solution has to satisfy the inequality

$$x + y \leq 11/5. \tag{1}$$

Observe that (1) is at the same time an upper bound on the objective function value. Furthermore, the solution $(x, y) = (8/5, 3/5)$ satisfies this constraint with equality, thus it is optimal.

Problem 4

Find the binary representation of 235.

Solution:
By applying the algorithm seen in class, we obtain that $235_2 = 11010111$.

Problem 5

Show that the binary representation with leading bit one of a positive natural number is unique.

Solution:
Assume that a number n has two representations, i.e. there are $a_0, \ldots, a_k, b_0, \ldots, b_{k'} \in \{0, 1\}$ such that $a_k = b_{k'} = 1$ and

$$n = \sum_{i=0}^{k} a_i \cdot 2^i = \sum_{j=0}^{k'} b_j \cdot 2^j.$$

We first show that $k = k'$. Otherwise, assume without loss of generality that $k' < k$, and we have

$$\sum_{j=0}^{k'} b_j \cdot 2^j < 2^{k'+1} \leq 2^k \leq \sum_{i=0}^{k} a_i \cdot 2^i,$$

a contradiction. Now, since $k = k'$ there must be an index ℓ such that $a_\ell = 1$ and $b_\ell = 0$. Choose ℓ as the smallest such index. Then we have:

$$\sum_{i=\ell+1}^{k} a_i \cdot 2^i + 2^\ell = \sum_{j=\ell+1}^{k} b_j \cdot 2^j,$$

which gives a contradiction since the right-hand side is divisible by $2^{\ell+1}$ while the left-hand side is not.

Problem 6

Show that there are n-bit numbers $a, b \in \mathbb{N}$ such that the Euclidean algorithm on input a and b performs $\Omega(n)$ arithmetic operations. *Hint: Fibonacci numbers*

Solution:
Since $F_n = F_{n-1} + F_{n-2}$, clearly the Euclidean division between F_n, F_{n-1} gives $q = 1, r = F_{n-2}$ for any $n \geq 2$, hence $GCD(F_n, F_{n-1})$ will call $GCD(F_{n-1}, F_{n-2})$, which will call $GCD(F_{n-2}, F_{n-3})$, etc., until $GCD(1, 0)$ is called. Hence the Euclidean algorithm performs $\Omega(n)$ recursive calls, hence $\Omega(n)$ arithmetic operations. To complete the proof we need to show that F_n can be represented with n bits: this follows from $F_n \leq 2^n$, which can be easily proved by induction.

Problem 7

Suppose we are given three $n \times n$ matrices $A, B, C \in \mathbb{Z}^{n \times n}$ and we want to test whether $A \cdot B = C$ holds. We could multiply A and B and then compare the result with C. This would amount to

running time (number of arithmetic operations) of $O(n^3)$ with the standard matrix-multiplication algorithm.
We now show how to perform an efficient *randomized test*. Suppose that you can draw a vector $v \in \{0,1\}^n$ i.i.d. at random in time $O(n)$. The idea is then to compute the product $B \cdot v$ and then the product $A \cdot (B \cdot v)$ and afterwards $C \cdot v$, all in time $O(n^2)$. Show the following.

a) If $A \cdot B \neq C$, then $P(A \cdot (B \cdot v) = C \cdot v) \leq 1/2$.

b) Let $v_1, \ldots, v_k \in \{0,1\}^n$ be i.i.d. at random and suppose that $A \cdot B \neq C$. The probability of the event: $A \cdot (B \cdot v_i) = C \cdot v_i$ for each $i = 1, \ldots, k$ is bounded by $1/2^k$.

c) Conclude that there is an algorithm that runs in time $O(k \cdot n^2)$ which tests whether $A \cdot B = C$ holds. The probability that the algorithm gives the wrong result is bounded by $1/2^k$.

Solution:

a) If $A \cdot B \neq C$, then there exist $i, j \in [n]$ such that $(A \cdot B)_{ij} \neq C_{ij}$. Without loss of generality one can assume that $j = n$. Observe that for every $\bar{v} \in \{0,1\}^{(n-1)}$, one has either $(A \cdot B)_i \cdot \begin{pmatrix} \bar{v} \\ 1 \end{pmatrix} \neq C_i \cdot \begin{pmatrix} \bar{v} \\ 1 \end{pmatrix}$ or $(A \cdot B)_i \cdot \begin{pmatrix} \bar{v} \\ 0 \end{pmatrix} \neq C_i \cdot \begin{pmatrix} \bar{v} \\ 0 \end{pmatrix}$. Thus, $(A \cdot B) \cdot v = C \cdot v$ for at most 2^{n-1} vectors $v \in \{0,1\}^n$, i.e.,

$$P(A \cdot (B \cdot v) = C \cdot v) \leq \frac{1}{2^n} 2^{n-1} = 1/2$$

for v being uniformly distributed in $\{0,1\}^n$.

b) Since $v_1, \ldots, v_k$ are i.i.d., we get directly from a) that the desired probability is equal to

$$\prod_{i \in [k]} P(A \cdot (B \cdot v_i) = C \cdot v_i) \leq 1/2^k.$$

c) Algorithm: Chose $v \in \{0,1\}^n$ uniformly at random and check whether $A(Bv) = Cv$. Repeat this process k times. If in any iteration the equation is unsatisfied, output "no", otherwise output "yes".

Calculating the two sides of the equation takes $O(n^2)$ arithmetic operations, including the linear number of comparisons. By repeating the process k times we have $O(kn^2)$ operations in total. The algorithm will possibly detect some "no" instances as "yes" (i.e., false positives). This probability is bounded by $1/2^k$ in b).

Problem 8 ($\star$)

Let a and b be two natural numbers with binary representations $a_0, \ldots, a_{l-1}$ and $b_0, \ldots, b_{l-1}$, respectively. Given that $a > b$ design an algorithm which outputs $c = a - b$ in its binary representation with leading bit one. Additionally, we require this algorithm to have the running time of $O(l)$ basic operations. The algorithm shall be implemented in C++.

Solution:

Here we provide a pseudo code. We are going to publish the best submitted C++ code on git.

Input: Two natural numbers a and b in their binary representations
$a_0, \ldots, a_{l-1}$, $b_0, \ldots, b_{l-1}$, where $a > b$.
Output: The binary representation of $c = a - b$ with the leading bit 1.

carry $:= 0$
for $i = 0, \ldots, l-1$
  $c_i :=$ carry $+ a_i + b_i \pmod 2$
  carry $:=$ (carry $\wedge b_i) \vee$ (carry $\wedge \neg a_i) \vee (b_i \wedge \neg a_i)$
$j := l - 1$
while $j > 0$ **and** $c_j = 0$
  $j = j - 1$
return $c_0, \ldots, c_j$

The algorithm performs $O(l)$ basic operations. Note that there cannot be any asymptotically faster algorithm since to correctly compute the sum we need to read all the input, hence we already need $\Omega(l)$ operations.