Cryptography, 2SAT, and Linear Programming

Shuvom Sadhuka
Outline

1. Extended Euclidean Algorithm
   a. Explanation
   b. Example

2. RSA
   a. Explanation
   b. Example

3. 2SAT
   a. 2SAT as a graph
   b. Randomized Algorithm

4. Linear Programming
   a. Explanation
   b. Example 1
   c. Example 2

5. Problems
Extended Euclidean Algorithm

Given $a, b$, the extended Euclidean algorithm calculates $x, y$ such that $ax+by=d$ where $d=\gcd(a,b)$. This is important for RSA.

FINDGCD(a,b):

$a \leftarrow$ larger number

$b \leftarrow$ smaller number

return $\leftarrow (d,x,y)$ s.t. $ax+by = d$ with $d=\gcd(a,b)$

**Base Case:**

if $b=0$ return $(a, 1, 0)$

**Recursive Case:**

$k \leftarrow (a - a \mod b)/b$

$(d,x,y) \leftarrow$ FINDGCD($b, a \mod b$)

Return $(d, y, x - ky)$
Extended Euclidean Algorithm

Given \( a, b \), the extended Euclidean algorithm calculates \( x, y \) such that \( ax + by = d \) where \( d = \text{GCD}(a, b) \). This is important for RSA.

FINDGCD\((a, b)\):

1. \( a \leftarrow \) larger number
2. \( b \leftarrow \) smaller number
3. \( \text{return} \leftarrow (d, x, y) \) s.t. \( ax + by = d \) with \( d = \text{gcd}(a, b) \)

Base Case:

- if \( b = 0 \) return \((a, 1, 0)\)

Recursive Case:

- \( k \leftarrow \lfloor a/b \rfloor \)
- \((d, x, y) \leftarrow\) FINDGCD\((b, a \mod b)\)
- \( \text{return} \leftarrow (d, y, x - ky) \)

1. \( k = \lfloor a/b \rfloor \)
2. \( bx + (a \mod b) y = d \)
3. \( bx + y (a-kb) = d \)
4. \( ay + b (x-ky) = d \)
Extended Euclidean Algorithm: Example

Given $a$, $b$, the extended Euclidean algorithm calculates $x, y$ such that $ax+by=d$ where $d=\text{GCD}(a,b)$. This is important for RSA.

\begin{align*}
\text{FINDGCD}(a,b): \\
a &\leftarrow \text{larger number} \\
b &\leftarrow \text{smaller number} \\
\text{return} &\leftarrow (d,x,y) \text{ s.t. } ax+by = d \text{ with } d=gcd(a,b) \\
\text{Base Case:} &\hspace{1cm} \text{if } b=0 \text{ return } (a, 1, 0) \\
\text{Recursive Case:} &\hspace{1cm} k \leftarrow (a - a \mod b)/b \\
&\hspace{1cm} (d,x,y) \leftarrow \text{FINDGCD}(b, a \mod b) \\
&\hspace{1cm} \text{return } (d, y, x - ky)
\end{align*}

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$k$</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>20</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Extended Euclidean Algorithm: Example

Given $a, b$, the extended Euclidean algorithm calculates $x,y$ such that $ax+by=d$ where $d=\text{GCD}(a,b)$. This is important for RSA.

FINDGCD($a,b$):

$a \leftarrow$ larger number

$b \leftarrow$ smaller number

return $\leftarrow (d,x,y)$ s.t. $ax+by = d$ with $d=\text{gcd}(a,b)$

**Base Case:**

if $b=0$ return $(a, 1, 0)$

**Recursive Case:**

$k \leftarrow \frac{(a - a \mod b)}{b}$

$(d,x,y) \leftarrow \text{FINDGCD}(b, a \mod b)$

return $(d, y, x - ky)$

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$k$</td>
<td>$x$</td>
<td>$y$</td>
</tr>
<tr>
<td>51</td>
<td>20</td>
<td>2</td>
<td>-9</td>
<td>23</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
<td>1</td>
<td>5</td>
<td>-9</td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>1</td>
<td>-4</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>-4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
RSA Motivation

1. Bob finds two large primes $p$, $q$ and computes $n=pq$. (Rabin-Miller)
RSA Motivation

1. Bob finds two large primes $p, q$ and computes $n = pq$. (Rabin-Miller)
2. Bob chooses $e < n$ s.t. $\text{GCD}(e, (p-1)(q-1)) = 1.$
RSA Motivation

1. Bob finds two large primes $p$, $q$ and computes $n=pq$. (Rabin-Miller)
2. Bob chooses $e < n$ s.t. GCD($e$, $(p-1)(q-1)) = 1$.
3. Bob calculates $d$ s.t. $de = 1 \mod (p-1)(q-1)$. (Extended Euclidean)
RSA Motivation

1. Bob finds two large primes $p$, $q$ and computes $n=pq$. (Rabin-Miller)

2. Bob chooses $e < n$ s.t. $\text{GCD}(e, (p-1)(q-1)) = 1$.

3. Bob calculates $d$ s.t. $de = 1 \text{ mod } (p-1)(q-1)$. (Extended Euclidean)

4. The public key is $k_e = (n, e)$ and the private key is $k_d = d$. 

$m \rightarrow m^e \mod n$
1. Bob finds two large primes $p$, $q$ and computes $n=pq$. (Rabin-Miller)

2. Bob chooses $e < n$ s.t. $\text{GCD}(e, (p-1)(q-1)) = 1$.

3. Bob calculates $d$ s.t. $de = 1 \mod (p-1)(q-1)$. (Extended Euclidean)

4. The public key is $k_e = (n, e)$ and the private key is $k_d = d$.

5. Alice encodes $c = m^e \mod n$. Bob decodes $c^d \mod n = m^{ed \mod (p-1)(q-1)} \mod n = m$. (Repeated Squaring)
Let $p=11$, $q=29$. Bob chooses $e=3$. What is $n$? What value of $d$ should be used in the private key? What is the encoding of $m=100$?
RSA

1. Bob finds two large primes $p$, $q$ and computes $n=pq$. (Rabin-Miller)

2. Bob chooses $e < n$ s.t. GCD($e$, $(p-1)(q-1)) = 1$.

3. Bob calculates $d$ s.t. $de = 1 \mod (p-1)(q-1)$. (Extended Euclidean)

4. The public key is $k_e = (n, e)$ and the private key is $k_d = d$.

5. Alice encodes $c = m^e \mod n$. Bob decodes $c^d \mod n = m^{ed} \mod (p-1)(q-1)$ mod $n = m$. (Repeated Squaring)

Let $p=11$, $q=29$. Bob chooses $e=3$. What is $n$? What value of $d$ should be used in the private key? What is the encoding of $m=100$?
Suppose we have logical expression that is the conjunction (AND) of multiple clauses, where each clause is a disjunction (OR) of two literals. For example:

\[(\bar{a} \lor b) \land (b \lor \bar{c}) \land (a \lor c) \land (\bar{a} \lor d)\]
2SAT: Implication Graph

\((\overline{a} \lor b) \land (b \lor \overline{c}) \land (a \lor c) \land (\overline{a} \lor d)\)
2SAT: Randomized Algorithm

\[(\overline{a} \lor b) \land (b \lor \overline{c}) \land (a \lor c) \land (\overline{a} \lor d)\]

At each step, take one unsatisfied clause and flip a coin to flip one literal
Linear Programming

A linear program is a type of optimization problem, where we want to (minimize/maximize) an objective function while obeying some constraints. In a linear program, the constraints and objective function are both linear in the variables of interest.

- Let $c_1 x_1 + c_2 x_2 + \ldots + c_n x_n = c$. Note that this defines an n-dimensional hyperplane.
- Constraints of the form $b_1 x_1 + b_2 x_2 + \ldots + b_n x_n \leq b$ define half-spaces.
- The set of points satisfying all the constraints is given by the intersection of all these half-spaces, which is a convex polyhedron.
- The objective function $a_1 x_1 + a_2 x_2 + \ldots + a_n x_n$ is a moveable hyperplane, we want to find the highest (maximization) or lowest (minimization) value of $a$ such that the hyperplane $a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = a$ still intersects the polyhedron.

Point of interest: a lot of interesting and difficult problems (e.g. certain NP-complete problems) can be phrased as linear programs.
Linear Programming Example

A farmer has 10 acres to plant in wheat and rye. He has to plant at least 7 acres. However, he has only $1200 to spend and each acre of wheat costs $200 to plant and each acre of rye costs $100 to plant. Moreover, the farmer has to get the planting done in 12 hours and it takes an hour to plant an acre of wheat and 2 hours to plant an acre of rye. If the profit is $500 per acre of wheat and $300 per acre of rye how many acres of each should be planted to maximize profits?
Problem 1

Run the Extended Euclidean algorithm for $a=111$ and $b=75$ to find their gcd, $d$, and find integers $x$ and $y$ such that $ax+by=d$. 
Problem 2

a) Prove that RSA is multiplicative: if $C_1$ is the encryption of $M_1$ and $C_2$ is the encryption of $M_2$ then $C_1 C_2$ is the encryption of $M_1 M_2$.

b) Let $(n, e)$ be a public key of RSA. Show that if there exists an efficient algorithm $A$ that decrypts 1 percent of messages, we can build an efficient randomized algorithm that decrypts every message with probability $>0.9$. 
Problem 3

(Fractional Knapsack)

a) We are filling a knapsack with items. The knapsack has a maximum weight capacity of $W$. Each item $i$ is associated with a value $v_i$ and weight $w_i$. We can split any item $i$ into a fractional piece $p_i$ (e.g. $p_i = \frac{1}{2}$) before putting the fractional piece into the bag, and the valuation of this piece will be $v_i p_i$. There are $n$ total items. Describe a linear program to find the maximum value knapsack we can fill.

b) Now suppose the knapsack is shared between $k$ different people who have different opinions about the values of items. For person $j$ ($j = 1, 2, ..., k$), the value of item $i$ is $v_{ij} \geq 0$. For each way of putting items into the knapsack, different people will have different evaluations for the knapsack ($p$ fraction of item $i$ will contribute $p \cdot v_{ij}$ to the evaluation of person $j$). We want to be fair to all the people, so we will define the actual value of the knapsack to be the minimum evaluation out of the $k$ people. Write a linear program that finds the maximum possible value of a knapsack.