EECS 376 Discussion 13

Sec 27: Th 5:30-6:30 DOW 1017

IA: Eric Khiu

Slide deck available at course drive/Discussion/Slides/Eric Khiu

Announcements

- HW 11 (last hw!) is due Tuesday 4/23
- Final Exam review session Wednesday, April 24th, from 7-9pm at BBB 1670
- Final Exam is Wednesday 5/1 from 7-9pm

Agenda

- Fermat's Little Theorem and Euler's Theorem
- RSA
- Cryptography and Complexity
- Zero Knowledge Proofs
- Final Exam Review (if time)

FLT and Euler's Theorem

Fermat's Little Theorem

- For a prime number p and any $a, k \in \mathbb{Z}$:

$$a^{1+k(p-1)} \equiv a \pmod{p}$$

- **Example:** Compute $5^{376185} \bmod 376183$ (Hint: 376183 is prime)

$$376185 = 1 + 1(376183 - 1) + 2$$

Here, we have $p = 376183$, $k = 1$, and $a = 5$. Applying Theorem 2.2.2,

$$\begin{aligned} 5^{376185} &\equiv 5^{1+1(376183-1)+2} \pmod{376183} \\ &\equiv 5^{1+1(376183-1)} \cdot 5^2 \pmod{376183} \\ &\equiv 5 \cdot 5^2 \pmod{376183} \\ &\equiv 125 \pmod{376183} \end{aligned}$$

Remark on FLT

- In the current version of course notes (and some books), the FLT is written as

Theorem 236 (Fermat's Little Theorem)

Let p be a prime number. Let a be any element of $\mathbb{Z}_p^+$, where

$$\mathbb{Z}_p^+ = \{1, 2, \ldots, p-1\}$$

Then $a^{p-1} \equiv 1 \pmod{p}$.

- We can derive our version of FLT easily as follows:

Proof. By FLT, we know $a^{p-1} \equiv 1 \pmod{p}$. Thus, raising 1 to any power k still results in 1:

$$(a^{p-1})^k \equiv 1^k \equiv 1 \pmod{p}$$

Multiply both sides by a,

$$a \cdot (a^{p-1})^k \equiv a \cdot 1 \pmod{p}$$
$$a^{1+k(p-1)} \equiv a \pmod{p}$$

Euler's Theorem

- For any integers a, k, and $n = pq$, where p and q are distinct primes

$$a^{1+k(p-1)(q-1)} \equiv a \pmod{n}$$

- **Example:** Compute $5^{376376} \bmod 35$

$(p-1)(q-1) = 6 \cdot 4 = 24$

$$\begin{aligned} 5^{376376} &\equiv 5^{15682 \cdot 24+1+7} \pmod{35} \\ &\equiv 5^{15682 \cdot 24+1} \cdot 5^7 \pmod{35} \\ &\equiv 5 \cdot 5^7 \pmod{35} \\ &\equiv 5^8 \pmod{35} \end{aligned}$$

From here, we apply repeated squaring:

$$\begin{aligned} 5^1 &\equiv 5 \pmod{35} \\ 5^2 &\equiv 5^2 \equiv 25 \pmod{35} \\ 5^4 &\equiv 25^2 \equiv 625 \equiv 30 \pmod{35} \\ 5^8 &\equiv 30^2 \equiv 900 \equiv 25 \pmod{35} \end{aligned}$$

RSA

Recap: Modular Inverse

- We say a^{-1} is the **modular (multiplicative) inverse** of a in mod n if

$$a^{-1} \cdot a \equiv 1 \pmod{n}$$

- Or equivalently, there exists some integer k such that

$$a^{-1} \cdot a = 1 + kn$$

- a has modular inverse in n iff a and n are coprime, i.e., $\gcd(a, n) = 1$
- If p is prime, then all $x \in \{1, 2, \ldots, p-1\}$ has a modular inverse
- We can find the modular inverse using Extended Euclid Algorithm

Euler's Theorem: For any integers a, k, and $n = pq$, where p and q are *distinct* primes

$$a^{1+k(p-1)(q-1)} \equiv a \pmod{n}$$

RSA Protocol

PUBLIC INFORMATION

$n \quad e \quad c$

Send $c = m^e \bmod n$

Compute $m' \equiv c^d \pmod{n}$

$$\equiv (m^e)^d \pmod{n}$$
$$\equiv m^{1+k(p-1)(q-1)} \pmod{n}$$
$$\equiv m \pmod{n}$$

Choose p, q, compute n

Find (e, d):

$$e \cdot d \equiv 1\big(\bmod\ (p-1)(q-1)\big)$$
$$\Rightarrow e \cdot d = 1 + k(p-1)(q-1)$$

Want to send m

Compute $c = m^e \bmod n$

RSA Encryption Example

- Alice performs several computations
  - Pick $p = 11$ and $q = 13$
  - Compute $n = 11 \cdot 13 = 143$
  - Compute $(p-1)(q-1) = 10 \cdot 12 = 120$
  - Pick $e = 17$, run `ExtendEuclid(17,120)` and get $d = 113$
  - Publicly broadcast n and e
- Bob wants to send $m = 5$
  - Compute $m^e \bmod n = 5^{17} \bmod 143 = 135$ and send to Alice
- Alice computes $m' = c^d = 135^{113} \bmod 143 = 5$

RSA Protocol

A: Choose p, q, compute $n = pq$

A: Find (e, d): $e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$

A: Broadcast n and e

B: Want to send m

B: Send $c = m^e \bmod n$ to A

A: Compute $c^d \equiv m \bmod n$

PUBLIC INFORMATION

n

e

c

RSA Encryption Security

- Suppose you were Eve and you want to find m

Poll: Which of the following information would allow you to decrypt the message? (select all apply)
A. The product $(p-1)(q-1)$
B. p and q individually
C. A k that satisfies $e \cdot d = 1 + k(p-1)(q-1)$

- Ans: All of the above! Key: You just need d
  A. If you have $(p-1)(q-1)$, you can run Extended Euclid Algorithm to find d
  B. If you have p and q, you can compute $(p-1)(q-1)$ → Case A
  C. We can express p and q in terms of k and n (see WS problem 5) → Case B

RSA Protocol

A: Choose p, q, compute $n = pq$

A: Find (e, d): $e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$

A: Broadcast n and e

B: Want to send m

B: Send $c = m^e \bmod n$ to A

A: Compute $c^d \equiv m \bmod n$

PUBLIC INFORMATION
n
e
c

RSA Signature

Euler's Theorem: For any integers a, k, and $n = pq$, where p and q are *distinct* primes

$$a^{1+k(p-1)(q-1)} \equiv a \pmod{n}$$

- Now suppose Alice wants to send a message rather than receiving a message, she wants to have people validate that it came from her

Want to send m

Compute $s = m^d \bmod n$

Send (m, s)

Choose p, q, compute n

Find (e, d):

$$e \cdot d \equiv 1 \left(\bmod\ (p-1)(q-1)\right)$$

$$\Rightarrow e \cdot d = 1 + k(p-1)(q-1)$$

PUBLIC INFORMATION

$n \qquad e$

Verify: $s^e \equiv \left(m^d\right)^e \pmod{n}$

$$\equiv m^{1+k(p-1)(q-1)} \pmod{n}$$

$$\equiv m \pmod{n}$$

RSA Signature Exercise

Prof. Wein is sending exam questions to the EECS 376 course staff. To ensure that the staff can verify the questions have not been altered, she uses an RSA-based signature scheme with $n = 55$ and public key $e = 27$. What would the signed message (m, s) be, if $m = 52$?

Hint: First find the prime factorization of n

$5 \cdot 11 = 55$ is the only prime factorization!

$(p-1)(q-1) = 4 \cdot 10 = 40$

RSA Signature

A: Choose p, q, compute $n = pq$

A: Find (e, d): $e \cdot d \equiv 1 \big(\bmod (p-1)(q-1)\big)$

A: Broadcast n and e

PUBLIC INFORMATION

$n \quad e$

A: Want to send m

A: Compute $s = m^d \bmod n$ and send to B

B: Verify if $s^e \equiv m \pmod n$

```
function ExtendedEuclid(x, y)
    if y = 0 then
        return (x, 1, 0)
    else
        Write x = qy + r for an integer q, where 0 ≤ r < y
        (g, a', b') ← ExtendedEuclid(y, r)
        a ← b'
        b ← a' − b'q
        return (g, a, b)
```

```
function ExtendedEuclid(x, y)
    if y = 0 then
        return (x, 1, 0)
    else
        Write x = qy + r for an integer q, where 0 ≤ r < y
        (g, a', b') ← ExtendedEuclid(y, r)
        a ← b'
        b ← a' − b'q
        return (g, a, b)
```

Step 1: Find modular inverse

- We have $(p-1)(q-1) = 40$, $e = 27$, we want to find d

x	y	q	r	g	$a \leftarrow b'$	$b \leftarrow a' - b'q$
40	27					

Step 1: Find modular inverse

```
function ExtendedEuclid(x, y)
    if y = 0 then
        return (x, 1, 0)
    else
        Write x = qy + r for an integer q, where 0 ≤ r < y
        (g, a', b') ← ExtendedEuclid(y, r)
        a ← b'
        b ← a' − b'q
        return (g, a, b)
```

- We have $(p-1)(q-1) = 40$, $e = 27$, we want to find d

x	y	q	r	g	$a \leftarrow b'$	$b \leftarrow a' - b'q$
40	27	1	13			
27	13	2	1			
13	1	13	0			
1	0	-	-	1	1	0

function ExtendedEuclid(x, y)
  if $y = 0$ **then**
    return $(x, 1, 0)$
  else
    Write $x = qy + r$ for an integer q, where $0 \le r < y$
    $(g, a', b') \leftarrow$ ExtendedEuclid(y, r)
    $a \leftarrow b'$
    $b \leftarrow a' - b'q$
    return (g, a, b)

Step 1: Find modular inverse

- We have $(p-1)(q-1) = 40$, $e = 27$, we want to find d

x	y	q	r	g	$a \leftarrow b'$	$b \leftarrow a' - b'q$
40	27	1	13	1	-2	1-(-2)(1)=3
27	13	2	1	1	1	0-(1)(2)=-2
13	1	13	0	1	0	1-(0)(13)=1
1	0	-	-	1	1	0

Step 2: Modular Exponentiation

- Now we have $m = 52$ and $d = 3$. We want to compute $m^d \bmod n = 52^3 \bmod 55$
  - Hint: $52 \equiv -3 \pmod{55}$
  - $52^3 \bmod 55 = (-3)^3 \bmod 55 = -27 \bmod 55 = 28$

RSA Signature

A: Choose p, q, compute $n = pq$

A: Find (e, d): $e \cdot d \equiv 1(\bmod (p-1)(q-1))$

A: Broadcast n and e

PUBLIC INFORMATION

n e

A: Want to send m

A: Compute $s = m^d \bmod n$ and send to B

B: Verify if $s^e \equiv m \pmod{n}$

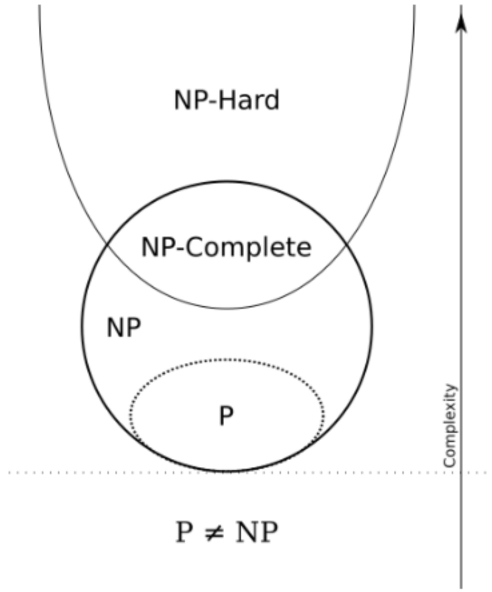
Cryptography and Complexity

Cryptography and NP-Completeness

- RSA and Diffie-Hellman both rely the problems of integer factorization and discrete log, respectively

- These problems are thought to be difficult
  - They're in NP, but are not known to be NP-Hard
  - If P≠NP, there must be languages between P and NP-Complete
  - This class is called NP-Intermediate

- DLOG is *expected* to be NP-Intermediate



Zero Knowledge Proofs

Zero Knowledge Proofs

- Think of a proof as an interaction between a prover and a verifier
  - The goal of the prover is to convince the verifier of some fact
  - In zero knowledge, the prover accomplishes this without giving the verifier any information they don't already know

- Formally, a zero knowledge proof that input x is in language L must meet:
  - **Completeness:** If $x \in L$, then the prover will cause the verifier to accept in any iteration of the game
  - **Soundness:** If $x \notin L$, then the verifier only accepts with small probability (i.e., the verifier for a vast majority of the prover's antics)
  - **Zero knowledge:** The verifier learns nothing besides the fact that $x \in L$
  - **Efficiency:** The prover puts on their whole show in polynomial time

ZKP Example: Sudoku

- In classic Sudoku, the objective is to fill a 9 × 9 grid with digits so that each column, each row, and each of the nine 3 × 3 subgrids that compose the grid contains all of the digits from 1 to 9
- Imagine a card version of sudoku where players place number cards in the grid

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				4
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

ZKP Example: Sudoku

- Suppose Alice (prover) wants to prove that the puzzle has a solution without giving away the solution to Bob (verifier)
- The protocol is as follows:
  1. Alice solves the puzzle behind Bob

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	2	8	4
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

ZKP Example: Sudoku

- Suppose Alice (prover) wants to prove that the puzzle has a solution without giving away the solution to Bob (verifier)
- The protocol is as follows:
  1. Alice solves the puzzle behind Bob
  2. Alice turns the cards she placed facing down

5	3	?	?	7	?	?	?	?
6	?	?	1	9	5	?	?	?
?	9	8	?	?	?	?	6	?
8	?	?	?	6	?	?	?	3
4	?	?	8	?	3	?	?	1
7	?	?	?	2	?	?	?	4
?	6	?	?	?	?	2	8	?
?	?	?	4	1	9	?	?	5
?	?	?	?	8	?	?	7	9

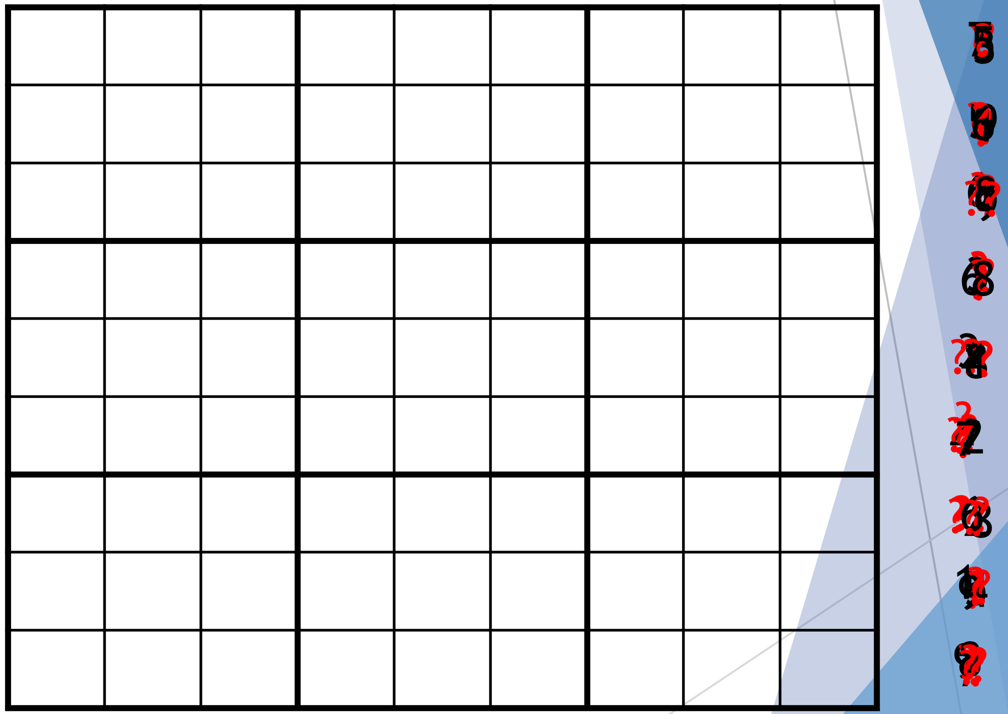
ZKP Example: Sudoku

- Suppose Alice (prover) wants to prove that the puzzle has a solution without giving away the solution to Bob (verifier)
- The protocol is as follows:
  1. Alice solves the puzzle behind Bob
  2. Alice turns the cards she placed facing down
  3. Bob specifies whether he wants to inspect rows, columns, or 3x3 subgrids

5	3	?	?	7	?	?	?	?
6	?	?	1	9	5	?	?	?
?	9	8	?	?	?	?	6	?
8	?	?	?	6	?	?	?	3
4	?	?	8	?	3	?	?	1
7	?	?	?	2	?	?	?	4
?	6	?	?	?	?	2	8	?
?	?	?	4	1	9	?	?	5
?	?	?	?	8	?	?	7	9

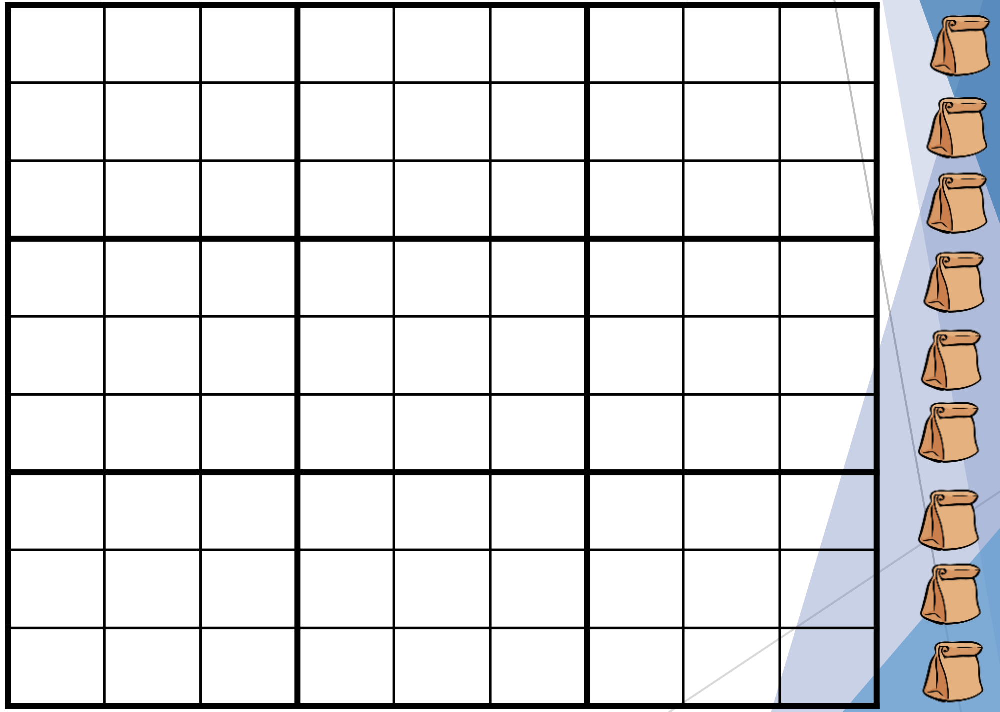
ZKP Example: Sudoku

- Suppose Alice (prover) wants to prove that the puzzle has a solution without giving away the solution to Bob (verifier)
- The protocol is as follows:
  1. Alice solves the puzzle behind Bob
  2. Alice turns the cards she placed facing down
  3. Bob specifies whether he wants to inspect rows, columns, or 3x3 subgrids
     - If Bob chooses <u>rows</u>, he packs the cards in each <u>row</u> into a packet



ZKP Example: Sudoku

- Suppose Alice (prover) wants to prove that the puzzle has a solution without giving away the solution to Bob (verifier)
- The protocol is as follows:
  1. Alice solves the puzzle behind Bob
  2. Alice turns the cards she placed facing down
  3. Bob specifies whether he wants to inspect rows, columns, or 3x3 subgrids
     - If Bob chooses rows, he packs the cards in each row into a packet
     - Alice shuffles the packet



ZKP Example: Sudoku

- Suppose Alice (prover) wants to prove that the puzzle has a solution without giving away the solution to Bob (verifier)
- The protocol is as follows:
  1. Alice solves the puzzle behind Bob
  2. Alice turns the cards she placed facing down
  3. Bob specifies whether he wants to inspect rows, columns, or 3x3 subgrids
     - If Bob chooses <u>rows</u>, he packs the cards in each <u>row</u> into a packet
     - Alice shuffles the packet
     - Bob verifies by checking if each packet contains 1-9

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

ZKP Example: Sudoku

- Suppose Alice (prover) wants to prove that the puzzle has a solution without giving away the solution to Bob (verifier)
- The protocol is as follows:
  1. Alice solves the puzzle behind Bob
  2. Alice turns the cards she placed facing down
  3. Bob specifies whether he wants to inspect rows, columns, or 3x3 subgrids
     - If Bob chooses columns, he packs the cards in each columns into a packet
     - Alice shuffles the packet
     - Bob verifies by checking if each packet contains 1-9

1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9

ZKP Example: Sudoku

Discuss: In step 3.2, why must Alice be the one shuffling the packets?

To ensure that Bob only receives the cards in a random order

- Suppose Alice (prover) wants to prove that the puzzle has a solution without giving away the solution to Bob (verifier)
- The protocol is as follows:
  1. Alice solves the puzzle behind Bob
  2. Alice turns the cards she placed facing down
  3. Bob specifies whether he wants to inspect rows, columns, or 3x3 subgrids
     - If Bob chooses 3x3 subgrids, he packs the cards in each 3x3 subgrids into a packet
     - Alice shuffles the packet
     - Bob verifies by checking if each packet contains 1-9
  4. Repeat step 3 for some number of times. (Note: Bob's choice is random every time)

ZKP Example: Sudoku

1. Alice solves the puzzle behind Bob
2. Alice turns the cards she placed facing down
3. Bob specifies whether he wants to inspect rows, columns, or 3x3 subgrids
   - Bob packs the cards in each row/ column/ subgrid into a packet
   - Alice shuffles the packet
   - Bob verifies if each packet is valid (contains 1-9)
4. Repeat step 3 for some number of times

- **Completeness:** If the puzzle has a solution, Alice can make each packet valid regardless of Bob's choice
- **Soundness:** If the puzzle has no solution, then no matter what Alice does, there must be at least one row/ column/ subgrid that are invalid and Bob has at least 1/3 chance of choosing a challenge that catches this
- **Zero-knowledge:** Bob learns nothing besides the fact that the puzzle has a solution. Specifically, Bob does not know the solution that Alice has. Bob only see some shuffled 1-9s, which he could have generated himself
- **Efficiency:** The verification process is efficient

Final Exam Review

Pick Your Problems

- “Advanced” Turing Reduction
- Verifiers
- Polytime Mapping Reduction
- Search to Decision Reduction
- α-Approximation Proof
- Computing Expected Value

Worksheet Problem 6

Let $\Sigma = \{0, 1\}$ and $L_\infty = \{\langle M \rangle \ : \ M$ accepts infinitely many strings in $\Sigma^*\}$. Is L_∞ decidable or not? If so, clearly describe a Turing machine that decides it; otherwise, prove that it is undecidable.

Worksheet Problem 7

Is the following an efficient verifier for the language $\textsc{No-Clique} = \{(G, k) \mid G \text{ has no clique of size } k\}$? Justify your answer.

V on input $((G, k), c)$:

1. Check that c is a set of k vertices in G, reject otherwise
2. For each pair of vertices (v_1, v_2) in c:
3.   If there is <u>not</u> an edge between v_1 and v_2 in c, *accept*.
4. *reject*.

Worksheet Problem 8b

We want to become a famous chefs. There are n ingredients to work with, and we wish to use m of them $(m \leq n)$ of them to create a new dish. There is an $n \times n$ matrix D that indicates the *discord* between two ingredients. In this case, $D[i, j]$ is an integer value between 0 and 10 (inclusive) where 0 means that the items i and j go together perfectly (there is no discord) and a 10 means they go together very badly. The penalty of a dish is the sum of the discord between each pair of ingredients. Consider the decision problem

$$\textsc{Recipe} = \{D, m, p : \text{ there is a dish with } \geq m \text{ ingredients that has penalty } \leq p\}.$$

(b) Show that $\textsc{Recipe}$ is NP-Hard.

Hint: You may find it helpful to imagine the matrix D as a graph on n vertices. Then consider what kinds of problems care about "all pairs".

Worksheet Problem 9

Recall the language

$$\textsc{Ham-Cycle} = \{\langle G \rangle : G \text{ is an undirected graph with a Hamiltonian cycle}\}.$$

Suppose that there exists a "black box" D that decides $\textsc{Ham-Cycle}$. Describe (with proof) an efficient algorithm that, given an undirected graph G, uses D to output a Hamiltonian cycle of G if one exists, and otherwise outputs "No Hamiltonian cycle exists."

Worksheet Problem 10

An *independent set* of an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ of vertices for which there is no edge between any pair of vertices in S. The *maximum independent set* (MIS) problem is: given a graph, find an independent set of maximum size.

Consider the following algorithm:

Let $S = \emptyset$ and let $G' = G$.

While G' still has at least one vertex:

i. Choose an arbitrary vertex v of G'.

ii. Let $S = S \cup \{v\}$.

iii. Remove v and all its neighbors (including all their incident edges) from G'. (A neighbor of v is any vertex that is connected to v by an edge.)

Output S.

(a) Let $U = V \setminus S$ denote the set of all vertices removed in step 2c, **not including** the vertices selected for S; and let Δ be the maximum degree of all vertices in G. Prove that $|U| \leq |S| \cdot \Delta$.

(b) Using the result from the previous part, conclude that the algorithm obtains a $1/(\Delta+1)$-approximation for MIS.

Worksheet Problem 11

A group of n students, all of whom have distinct heights, line up in a single-file line uniformly at random to get a group picture taken. If a student has any students in front of them who is taller than them, then they will not be seen in the picture. For this reason, every student files one complaint to the photographer for <u>each</u> taller student who is in front of them since each one of these students would individually block the original student from being seen. Compute the expected number of complaints that the photographer will receive.

Thanks for a great semester

- Consider the following classes to learn more about these topics
  - Algorithms: EECS 477, CSE 486
  - Complexity: CSE 574
  - Randomness: CSE 572
  - Cryptography: EECS 475, CSE 575
  - Check the EECS 498/598 list!
- Good luck on the final!