# Pre-lecture brain teaser

$L$ - - - - - -> $L^* \in NP$ or not?

Is NP is closed under the kleene-star operation?

$L \rightarrow L^*$

Problem

Language

Machine

YES!

# Pre-lecture brain teaser

Is $\mathrm{NP}$ is closed under the kleene-star operation? Let $A \in \mathrm{NP}$. Construct NTM $M$ to decide A in nondeterministic polynomial time.

from the kleene-star language!

$w \in A^*$

M [On input $w$]:

1. Non-deterministically divide $w$ into pieces $w = x_1 x_2 \cdots x_k$.
2. For each $x_i$, nondeterministically guess the certificates that show $x_i \in A$.
3. Verify all certificates if possible, then *accept*, Otherwise if verification fails, *reject*.

10 1001 → 101001

A → T  $L(T) = A$  $w \in A^*$

$w = x_1 x_2 \cdots x_k$  $x_i \in A$

Poly-time? of M → Exercise

$w \in A^* \Leftrightarrow x_i \in A \ \forall i$

# ECE-374-B: Lecture 26 - Final Review

Instructor: Abhishek Kumar Umrawal

Apr 30, 2024

University of Illinois at Urbana-Champaign

# Pre-lecture brain teaser

Is $\mathrm{NP}$ is closed under the kleene-star operation?

# Pre-lecture brain teaser

Is $\mathrm{NP}$ is closed under the kleene-star operation? Let $A \in \mathrm{NP}$. Construct NTM $M$ to decide A in nondeterministic polynomial time.

M [On input $w$]:

1. Non-deterministically divide $w$ into pieces $w = x_1 x_2 \cdots x_k$.
2. For each $x_i$, nondeterministically guess the certificates that show $x_i \in A$.
3. Verify all certificates if possible, then *accept*, Otherwise if verification fails, *reject*.

# Announcements

- Midterm 3 grading is in progress and shall finish by this Wednesday.
- Grade estimates will be released as soon as possible after the midterm 3 grades are released.
- Final is next Tuesday. Good luck to who are taking it!
- Please do your ICES evaluations (or, you know, talk to me).

# Final Topics

Topics for the final exam include:

- Everything on Midterm 1:
  - Regular expressions
  - DFAs, NFAs,
  - Fooling Sets and Closure properties
  - CFGs and PDAs
  - CSGs and LBAs
- Turing Machines
- MST Algorithms (Upto Boruvka's Algo.)
  Upto Slide 15

Less emphasis

- Everything on Midterm 2
  - Asymptotic Bounds
  - Recursion, Backtracking
  - Dynamic Programming
  - DFS/BFS
  - DAGs and TopSort
  - Shortest path algorithms
- Everything on Midterm 3
  - Reductions
  - P, NP, NP-hardness
  - Decidability

# Final Topics

In today's lecture let's focus on a few that you guys had trouble on in the midterms (and the most recent stuff whih you'll be tested on).

- Everything on Midterm 1:
  - **Regular expressions**
  - **DFAs, NFAs,**
  - **Fooling Sets and Closure properties**
  - **CFGs** and **PDAs**
  - CSGs and LBAs
- Turing Machines
- MST Algorithms
- Everything on Midterm 2
  - **Asymptotic Bounds**
  - Recursion, Backtracking
  - Dynamic Programming
  - DFS/BFS
  - DAGs and TopSort
  - Shortest path algorithms
- Everything on Midterm 3
  - Reductions
  - **P, NP, NP-hardness**
  - Decidability

# Practice: Asymptotic bounds

Given an asymptotically tight bound for:

$$T(n) = \sum_{i=1}^{n} i^3 \tag{1}$$

$T(n) = \Theta(n^4)$

1. $T(n) = \sum_{i=1}^{n} i^3 = \dfrac{n^2(n+1)^2}{4} = \Theta(n^4)$

$1^3 + 2^3 + 3^3 + \cdots + n^3$

2. $T(n) = T(n-1) + n^3$ (Check!) $T(1) = 1$

$n^3 \cdot n = n^4$

$n \;\text{—}\; n^3$

$\downarrow$

$n-1 \;\text{—}\; (n-1)^3$

$\downarrow$

$n-2$

$\vdots$

$1$

$n$

# Practice: Asymptotic bounds

Given an asymptotically tight bound for:

$$\sum_{i=1}^{n} i^3 \tag{1}$$

**Answer:** $\Theta(n^4)$

Explanation: The closed form for the above sum is $\frac{n^2(n+1)^2}{4}$.

# Practice: Regular expressions

Find the regular expression for the language:

$$\{w \in \{0,1\}^* | w \text{ does not contain } \underline{00} \text{ as a substring}\} \qquad (2)$$

$\epsilon$

0

01

011

01010

## Practice: Regular expressions

Find the regular expression for the language:

$$\{w \in \{0,1\}^* | w \text{ does not contain 00 as a substring}\} \qquad (2)$$

Solution 1: $(\varepsilon + 0)(1 + 10)^*$

Solution 2: $1^*(011^*)^*(\varepsilon + 0)$

## Practice: Fooling Sets

Is the following language regular?

$$\mathsf{L} = \{w | w \text{ does not contain the substring } 00 \text{ nor } 11\ \}$$

## Practice: Fooling Sets

Is the following language regular?

$$\mathsf{L} = \{w | w \text{ does not contain the substring } 00 \text{ nor } 11\}$$

$$R = (01)^* + 0 + 1 + \varepsilon$$

# Practice: Fooling Sets

Is the following language regular?

$$\mathsf{L} = \{w | w \text{ has an equal number of } 0\text{'s and } 1\text{'s } \}$$

$F = \{ 0^i : i \geq 0 \}$   $|F| = \infty$

$0^i \in F$
$0^j \in F$   $i \neq j$   $1^i \in \Sigma^*$   $0^i 1^i \in L$
$0^j 1^i \notin L$

$\Rightarrow$ F is an inf. fooling set for L
$\Rightarrow$ L is non-reg.

$L = \{ 0^n 1^n : n \geq 0 \}$ : Reg ( × ) NO
Non-Reg ( ✓ ) YES

# Practice: Fooling Sets

Is the following language regular?

$$\mathsf{L} = \{w | w \text{ has an equal number of } 0\text{'s and } 1\text{'s }\}$$

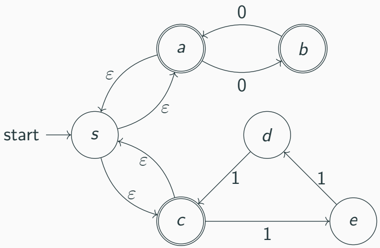
Consider the following set.

$F = \{0^i | i \geq 0\}$

Take $0^i \in F$ and $0^j \in F$ such that $i \neq j$. For $x = 1^i$, observe that $0^i 1^i \in L$ and $0^j 1^i \notin L$. This implies that $F$ is an infinite fooling set for $L$. Hence, $L$ is not regular.

# Practice: NFAs and DFAs

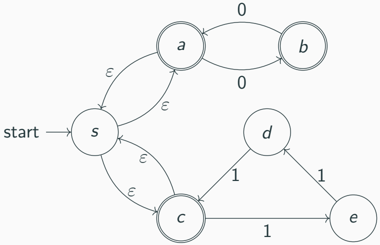
Let M be the following NFA:




Which of the following statements about $M$ are true?

# Practice: NFAs and DFAs

Let M be the following NFA:

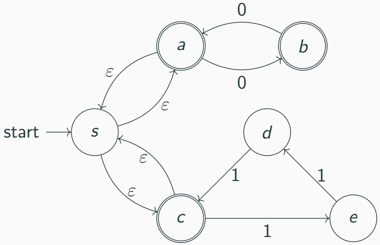



Which of the following statements about $M$ are true?

1. M accepts the empty string
   $\varepsilon$ - True

# Practice: NFAs and DFAs

Let M be the following NFA:

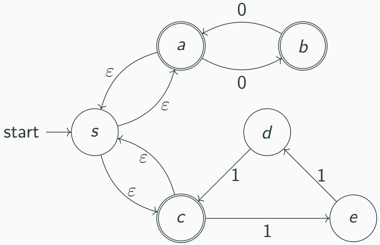



Which of the following statements about $M$ are true?

1. M accepts the empty string $\varepsilon$ - True
2. $\delta(s, 010) = \{s, a, c\}$ - False. $\delta(s, 010) = \emptyset$

# Practice: NFAs and DFAs

Let M be the following NFA:

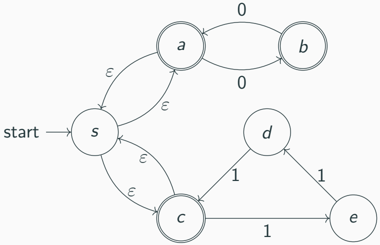



Which of the following statements about $M$ are true?

1. M accepts the empty string $\varepsilon$ - True
2. $\delta(s, 010) = \{s, a, c\}$ - False. $\delta(s, 010) = \emptyset$
3. $\varepsilon - \text{reach}(a) = \{s, a, c\}$ - True

# Practice: NFAs and DFAs

Let M be the following NFA:

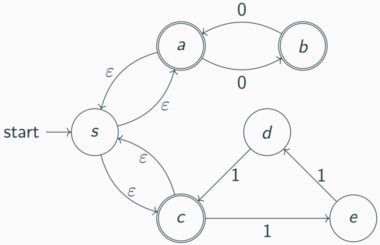



Which of the following statements about $M$ are true?

1. M accepts the empty string $\varepsilon$ - True
2. $\delta(s, 010) = \{s, a, c\}$ - False. $\delta(s, 010) = \emptyset$
3. $\varepsilon - \mathrm{reach}(a) = \{s, a, c\}$ - True
4. $M$ rejects the string 11100111000 - True - three zeros at end, end up in state **b**

# Practice: NFAs and DFAs

Let M be the following NFA:




Which of the following statements about $M$ are true?

1. M accepts the empty string $\varepsilon$ - True
2. $\delta(s, 010) = \{s, a, c\}$ - False. $\delta(s, 010) = \emptyset$
3. $\varepsilon - \mathrm{reach}(a) = \{s, a, c\}$ - True
4. $M$ rejects the string 11100111000 - True - three zeros at end, end up in state **b**
5. $L(M) = (00)^* + (111)^*$ - False - $L(M) = (00 + 111)^*$

# Practice: Closure

Which of the following is true for **every** language $L \subseteq \{0,1\}^*$

1. $L^*$ is non-empty - True - $L^*$ always contains the empty string $\varepsilon$
2. $L^*$ is regular - False - See previous example. Let $L = \{0^{n^2}1|\}$ always contains the empty string $\varepsilon$
3. If $L$ is NP-Hard, then L is not regular - True - All regular languages are in P, becuase DFA's are linear time algorithms
4. If $L$ is not regular, then $L$ is undecidable - False - The language $L = \{0^n1^n | n \geq 0\}$ is not regular but still decidable

# Context-Free Languages

Given $\Sigma = 0, 1$, the language $L = \{0^n 1^n | n \geq 0\}$ is represented by which grammar?

(a)

$$S \rightarrow 0T1|1$$
$$T \rightarrow T0|\varepsilon$$

(b)

$$S \rightarrow 0S1$$

(c)

$$S \rightarrow 0S1|0S|S1|\varepsilon$$

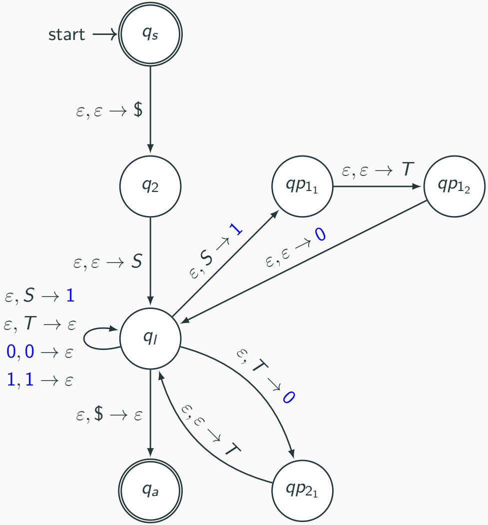
(d)

$$S \rightarrow AB1$$
$$A \rightarrow 0$$
$$B \rightarrow S|\varepsilon$$

(e) None of the above

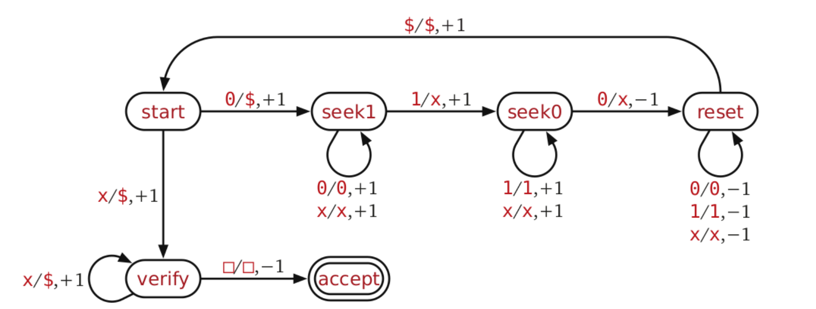
# Push-down Auto-mata

What is the context-free grammar of the following push-down automata:



# Turing machines

You have the following Turing machine diagram that accepts a particular language whose alphabet $\Sigma = \{0, 1\}$. Please describe the language.




$L = \{0^n 1^n 0^n | n \geq 0\}$

# Linear Time Selection

Recall the linear time selection logarithm that uses the medians of medians. We use the same algorithm, but instead of lists of size 5, webreak the array into lists of size 7 and do the median-of-medians as normal. The running time for my new algorithm is:

(a) $O(\log(n))$
(b) $O(n)$
(c) $O(n\log(n))$
(d) $O(n^2)$
(e) None of the above

# Linear Time Selection

Recall the linear time selection logarithm that uses the medians of medians. We use the same algorithm, but instead of lists of size 5, webreak the array into lists of size 7 and do the median-of-medians as normal. The running time for my new algorithm is:

(a) $O(\log(n))$
(b) $O(n)$
(c) $O(n\log(n))$
(d) $O(n^2)$
(e) None of the above

Why did we choose lists of size 5? Will lists of size 3 work?

(Hint) Write a recurrence to analyze the algorithm's running time if we choose a list of size $k$.

## Graph Exploration

We looked at the BasicSearch algorithm:

```
Explore(G, u):
    Visited[1 .. n] ← FALSE
    // ToExplore, S: Lists
    Add u to ToExplore and to S
    Visited[u] ← TRUE
    while (ToExplore is non-empty) do
        Remove node x from ToExplore
        for each edge xy in Adj(x) do
            if (Visited[y] = FALSE)
                Visited[y] ← TRUE
                Add y to ToExplore
                Add y to S
    Output S
```

We said that if ToExplore was a:

- Stack, the algorithm is equivalent to **DFS**
- Queue, the algorithm is equivalent to **BFS**

What if the algorithm was written recursively (instead of the while loop, you recursively call explore). What would the algorithm be equivalent to?

# Minimum Spanning Trees

Let G = (V,E) be a connected, undirected graph with edge weights w, such that the weights are distinct, i.e., no two edges have the same weight. Which of the following is necessarily true about a minimum spanning tree of G?

(a) If $T_1$ and $T_2$ are MSTs of G then $T_1 = T_2$ , i.e., the MST is unique.

(b) There are MSTs $T_1$ and $T_2$ such that $T_1 \neq T_2$ i.e, MST is not unique.

(c) There is an edge $e$ that is **unsafe** that belongs to a MST.

(d) There is a **safe** edge that does not belong to a MST of G.

# Reduction: 3SAT to Clique

Consider the two problems:

3SAT : NP-C

CLIQUE is also NP-C

CLIQUE is in NP

$3SAT \le_p$ CLIQUE

**Problem: 3SAT**

**Instance:** Given a CNF formula $\varphi$ with $n$ variables, and $k$ clauses
**Question:** Is there a truth assignment to the variables such that $\varphi$ evaluates to true

**Problem: Clique**

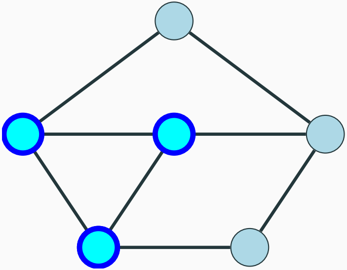
**Instance:** A graph G and an integer $k$.
**Question:** Does G has a clique of size $\geq k$?

Reduce **3SAT** to **CLIQUE**

# Reduction: 3SAT to Clique

Given a graph $G$, a set of vertices $V'$ is:

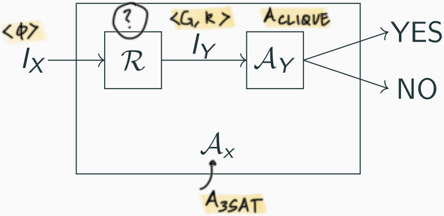
clique: every pair of vertices in $V'$ is connected by an edge of $G$.



$\langle \phi \rangle$ : 3SAT $\longrightarrow$ $\langle G, k \rangle$ : CLIQUE

# Reduction: 3SAT to Clique

Bust out the reduction diagram:



## Reduction: 3SAT to Clique

Some thoughts:

- Clique is a fully connected graph and very similar to the independent set problem
- We want to have a clique with all the satisfying literals
  - Can't have literal and its negation in same clique
  - Only need one satisfying literal per clique

# Reduction: 3SAT to Clique

Hence the reduction creates a undirected graph $G$:

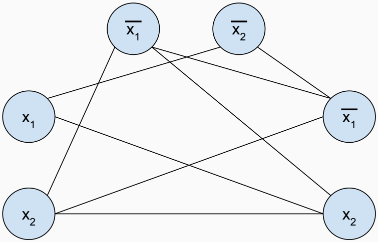
- Nodes in G are organized in $k$ groups of nodes. Each triple corresponds to one clause.
- The edges of G connect all but:
  - nodes in the same triple
  - nodes with contradictory labels ($x_1$ and $\overline{x_1}$)

## Reduction: 3SAT to Clique

Hence the reduction creates a undirected graph $G$:

- Nodes in G are organized in $k$ groups of nodes. Each triple corresponds to one clause.
- The edges of G connect all but:
  - nodes in the same triple
  - nodes with contradictory labels ($x_1$ and $\overline{x_1}$)

$\varphi = (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2)$

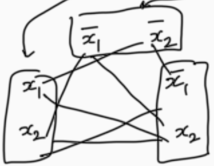


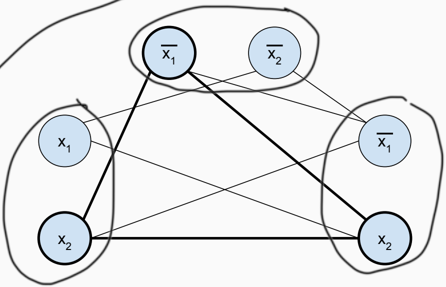
# Reduction: 3SAT to Clique

Hence the reduction creates a undirected graph $G$:

- Nodes in G are organized in $k$ groups of nodes. Each triple corresponds to one clause.
- The edges of G connect all but:
  - nodes in the same triple
  - nodes with contradictory labels ($x_1$ and $\overline{x_1}$)

$\varphi = (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2)$ k = 3 clauses



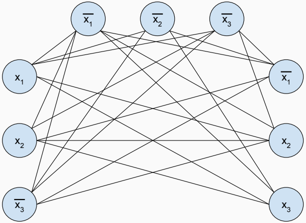



# Reduction: 3SAT to Clique

Hence the reduction creates a undirected graph $G$:

- Nodes in G are organized in $k$ groups of nodes. Each triple corresponds to one clause.
- The edges of G connect all but:
    - nodes in the same triple
    - nodes with contradictory labels ($x_1$ and $\overline{x_1}$)

$\varphi = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

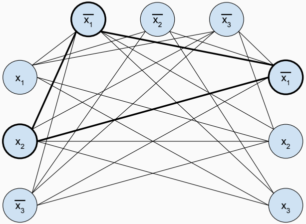


# Reduction: 3SAT to Clique

Hence the reduction creates a undirected graph $G$:

- Nodes in G are organized in $k$ groups of nodes. Each triple corresponds to one clause.
- The edges of G connect all but:
  - nodes in the same triple
  - nodes with contradictory labels ($x_1$ and $\overline{x_1}$)

$\varphi = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$



# 3SAT to Independent Set Reduction

Very similar to 3SAT to independent set reduction:

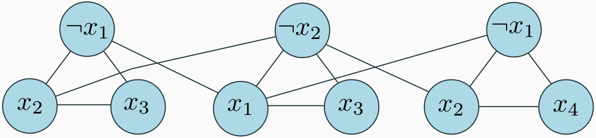



**Figure 1:** Graph for $\varphi = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$