# Computational Complexity

Lecture 1
in which we talk about
Time Complexity, P, NP and coNP

# Evolution of Computation

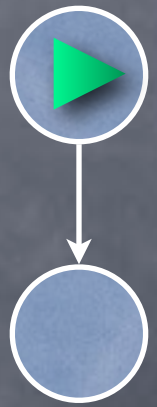
# Evolution of Computation

- The program (Turing Machine) starts in an initial configuration (tape-contents, control-state, head-position)

# Evolution of Computation

- The program (Turing Machine) starts in an initial configuration (tape-contents, control-state, head-position)
  - input explicitly encoded in the initial configuration

# Evolution of Computation



- The program (Turing Machine) starts in an initial configuration (tape-contents, control-state, head-position)
  - input explicitly encoded in the initial configuration
- At every step the configuration evolves

# Evolution of Computation

- The program (Turing Machine) starts in an initial configuration (tape-contents, control-state, head-position)
    - input explicitly encoded in the initial configuration
- At every step the configuration evolves

# Evolution of Computation

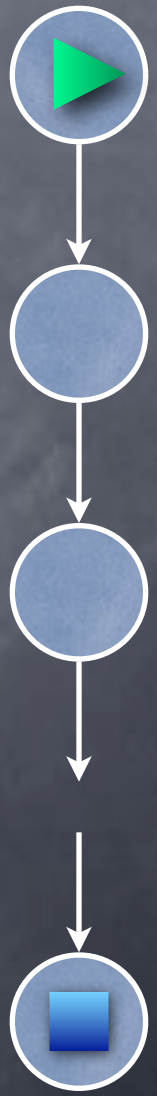
- The program (Turing Machine) starts in an initial configuration (tape-contents, control-state, head-position)
  - input explicitly encoded in the initial configuration
- At every step the configuration evolves

# Evolution of Computation

- The program (Turing Machine) starts in an initial configuration (tape-contents, control-state, head-position)
  - input explicitly encoded in the initial configuration
- At every step the configuration evolves
- Until computation terminates: final configuration

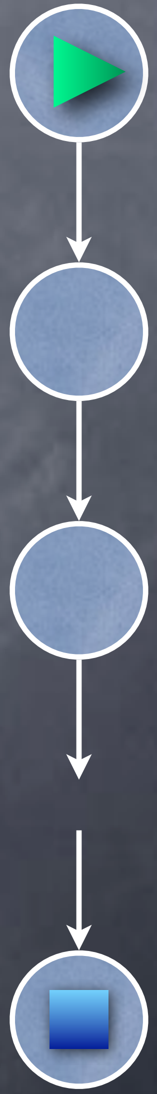
# Evolution of Computation

- The program (Turing Machine) starts in an initial configuration (tape-contents, control-state, head-position)
  - input explicitly encoded in the initial configuration
- At every step the configuration evolves
- Until computation terminates: final configuration

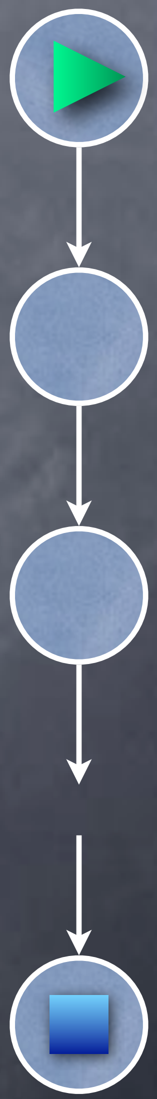


# Evolution of Computation

- The program (Turing Machine) starts in an initial configuration (tape-contents, control-state, head-position)
  - input explicitly encoded in the initial configuration
- At every step the configuration evolves
- Until computation terminates: final configuration
  - output explicitly encoded in the final configuration (say, in the control-state)

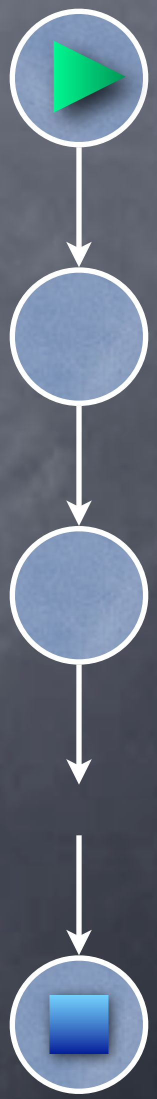


# Time Complexity



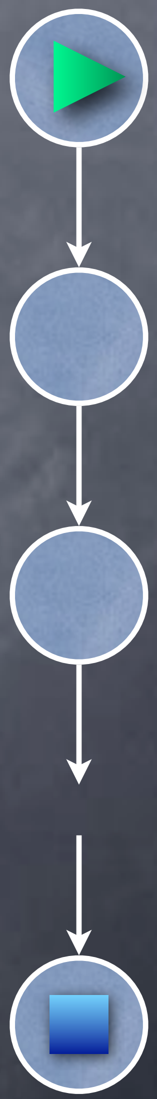
# Time Complexity

- Deterministic TM computation model



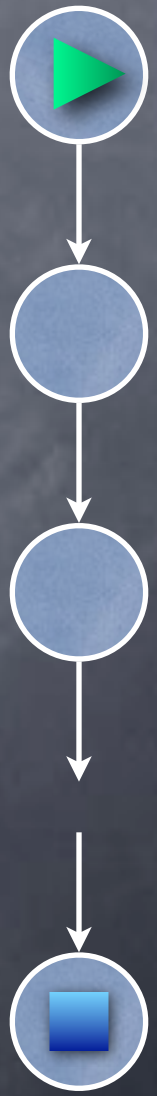
# Time Complexity

- Deterministic TM computation model
- Program (deterministic TM) succinctly specifies the "next configuration" function



# Time Complexity

- Deterministic TM computation model
- Program (deterministic TM) succinctly specifies the "next configuration" function
- Time Complexity of language L (worst case): if there is a TM that decides L (correct on all instances), and for any input instance of size n, it takes at most T(n) steps then L in class DTIME(T)



# Time Complexity

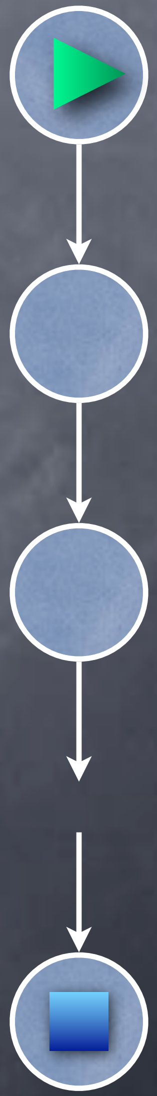
- Deterministic TM computation model
- Program (deterministic TM) succinctly specifies the "next configuration" function
- Time Complexity of language L (worst case): if there is a TM that decides L (correct on all instances), and for any input instance of size n, it takes at most T(n) steps then L in class DTIME(T)

worst-case complexity

# Time Complexity

- Deterministic TM computation model
- Program (deterministic TM) succinctly specifies the "next configuration" function
- Time Complexity of language L (worst case): if there is a TM that decides L (correct on all instances), and for any input instance of size n, it takes at most T(n) steps then L in class DTIME(T)
  - (Note: complexity T is a function of n)

worst-case complexity



# P for Polynomial Time

# P for Polynomial Time

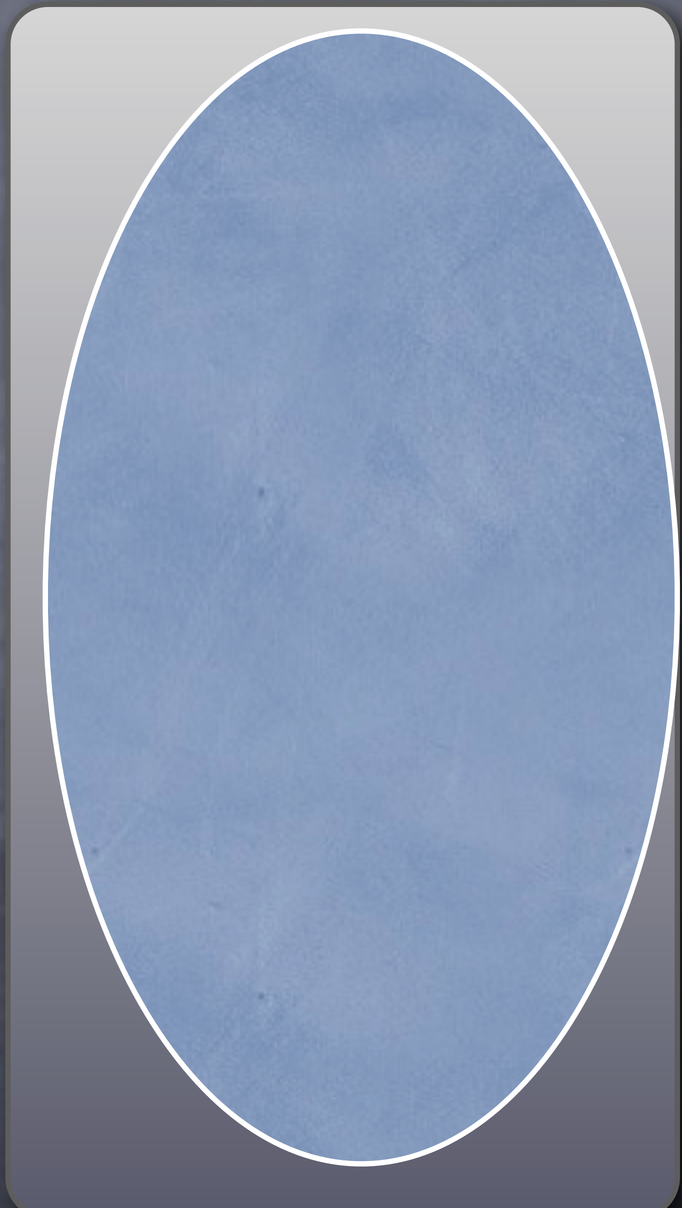
- If a problem is in DTIME(T) and $T(n)=O(n^c)$ for some c, then the problem is in P

# P for Polynomial Time

- If a problem is in DTIME(T) and $T(n)=O(n^c)$ for some c, then the problem is in P
  - $P = \bigcup_{a,b,c>0} \mathrm{DTIME}(a.n^c+b)$

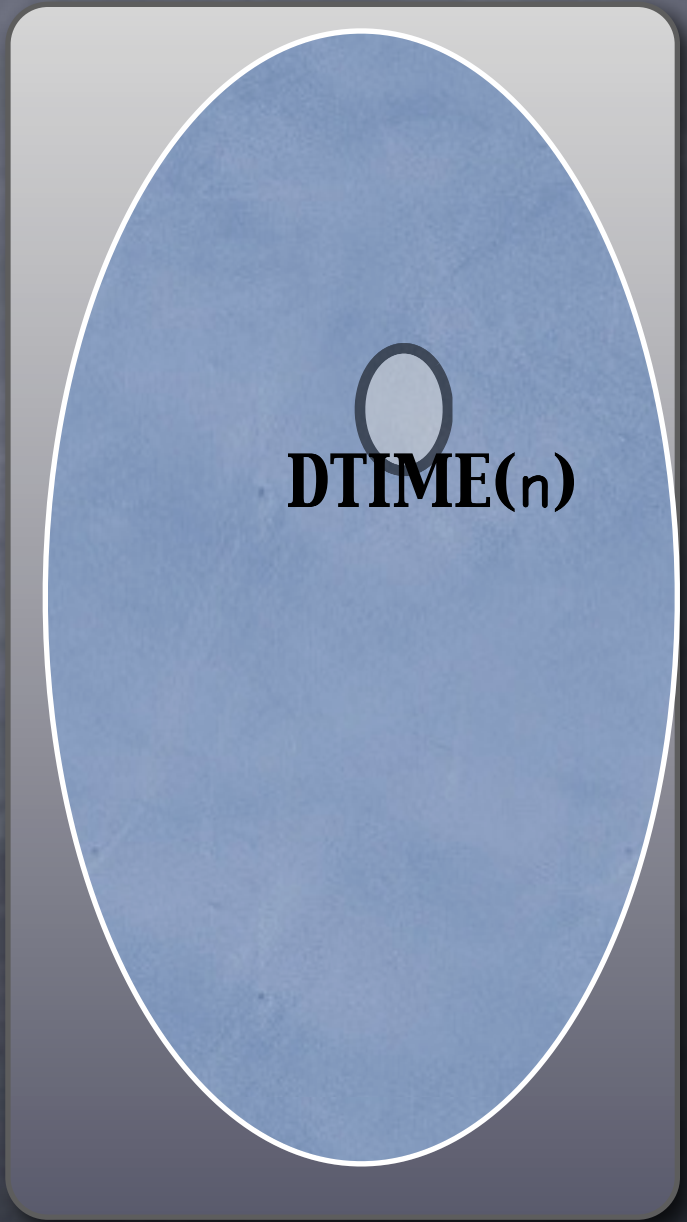
# P for Polynomial Time

- If a problem is in DTIME(T) and $T(n)=O(n^c)$ for some c, then the problem is in P
  - $P = \bigcup_{a,b,c > 0} \text{DTIME}(a.n^c+b)$



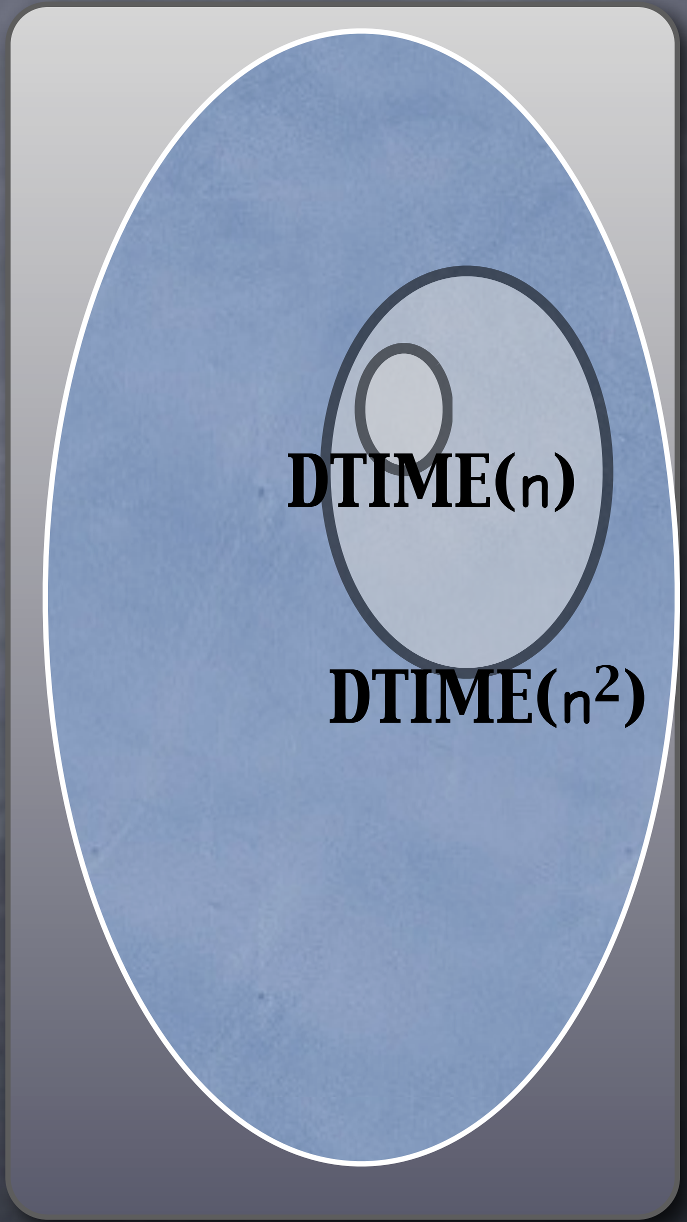
# P for Polynomial Time

- If a problem is in DTIME(T) and $T(n)=O(n^c)$ for some c, then the problem is in P
  - $P = \bigcup_{a,b,c>0} \mathrm{DTIME}(a.n^c+b)$



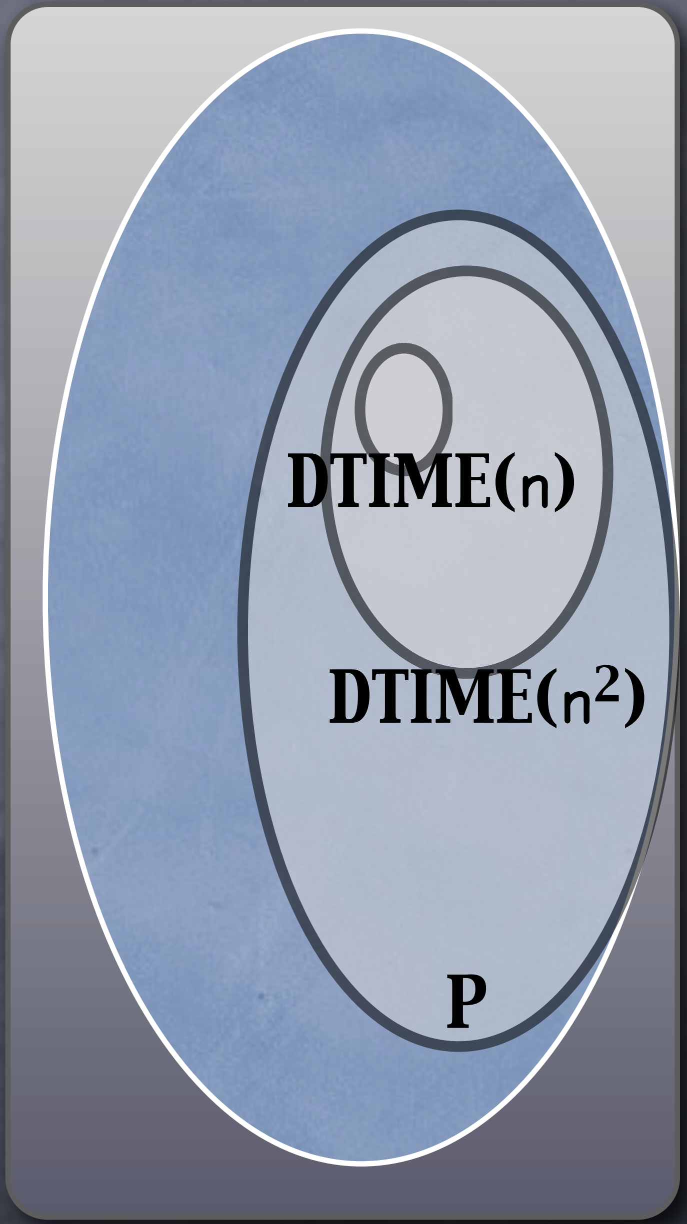
# P for Polynomial Time

- If a problem is in DTIME(T) and $T(n)=O(n^c)$ for some c, then the problem is in P
  - $P = \bigcup_{a,b,c>0} \mathrm{DTIME}(a.n^c+b)$



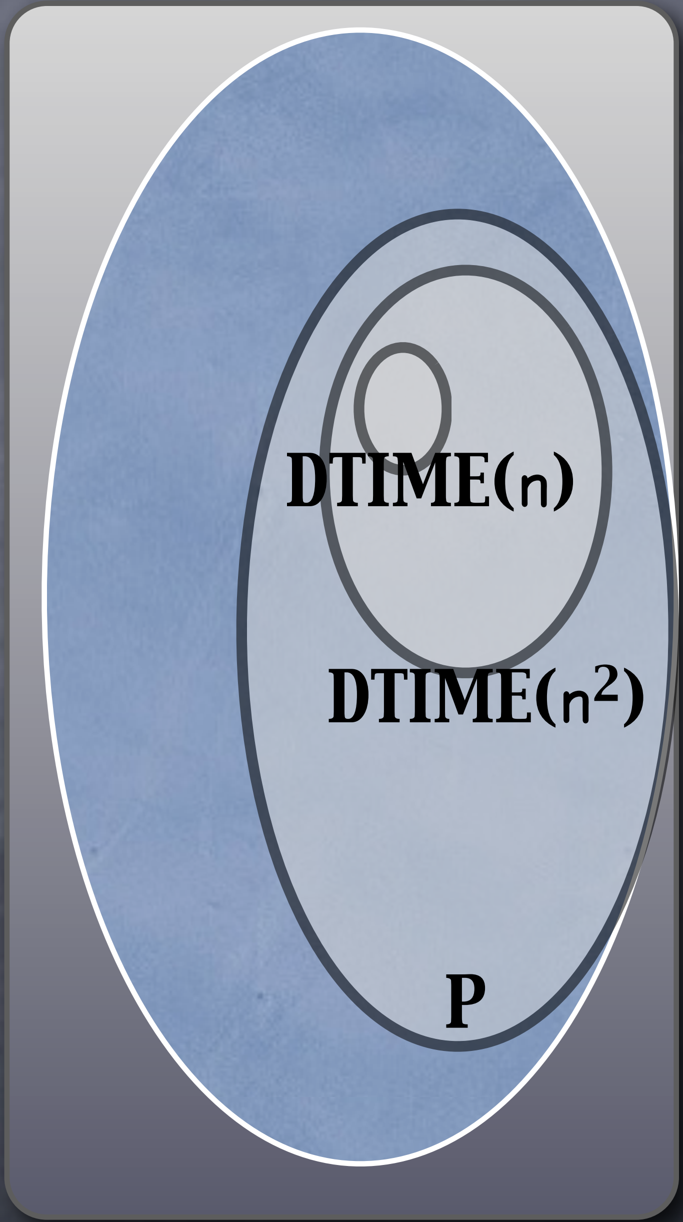
# P for Polynomial Time

- If a problem is in DTIME(T) and $T(n)=O(n^c)$ for some c, then the problem is in P
  - $P = \bigcup_{a,b,c > 0} \mathrm{DTIME}(a.n^c+b)$



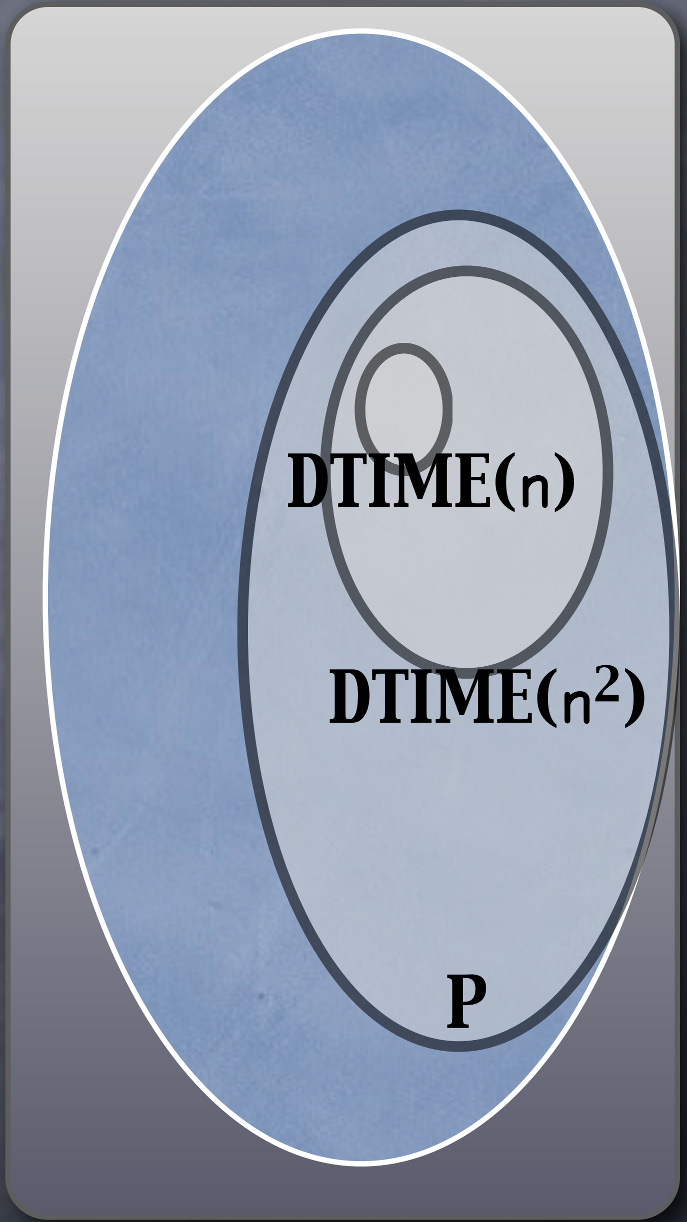
# P for Polynomial Time

- If a problem is in DTIME(T) and $T(n)=O(n^c)$ for some c, then the problem is in P
    - $P = \bigcup_{a,b,c>0} \mathrm{DTIME}(a.n^c+b)$
- DTIME(T) depends on the specifics of the TM model (no. of tapes, alphabet size)



# P for Polynomial Time

- If a problem is in DTIME(T) and $T(n)=O(n^c)$ for some c, then the problem is in P
    - $P = \bigcup_{a,b,c>0} \mathrm{DTIME}(a.n^c+b)$
- DTIME(T) depends on the specifics of the TM model (no. of tapes, alphabet size)
- But P is robust: Models can simulate each other with only "polynomial slow down"



# Non-deterministic Computation

# Non-deterministic Computation

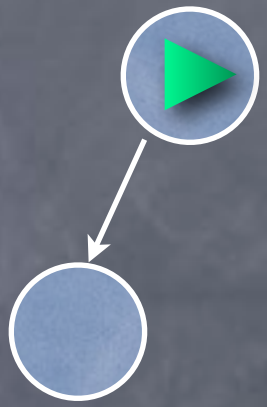
- Not “realistic” as a computation model, but has realistic interpretations (coming up)

# Non-deterministic Computation

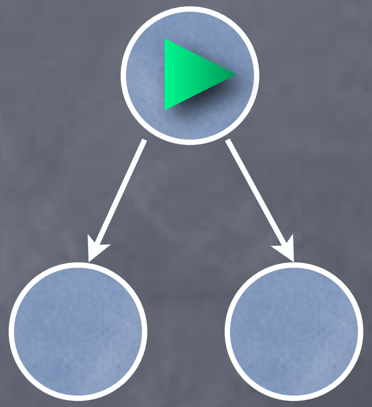
- Not "realistic" as a computation model, but has realistic interpretations (coming up)

# Non-deterministic Computation

- Not "realistic" as a computation model, but has realistic interpretations (coming up)



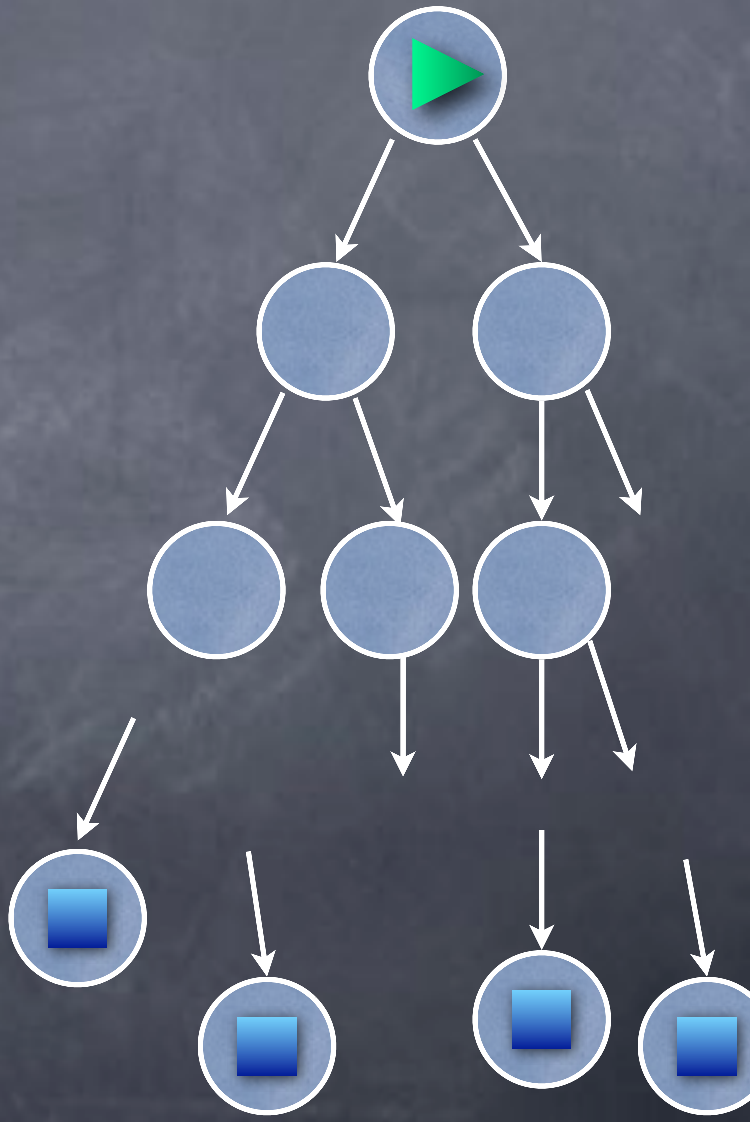
# Non-deterministic Computation



- Not "realistic" as a computation model, but has realistic interpretations (coming up)

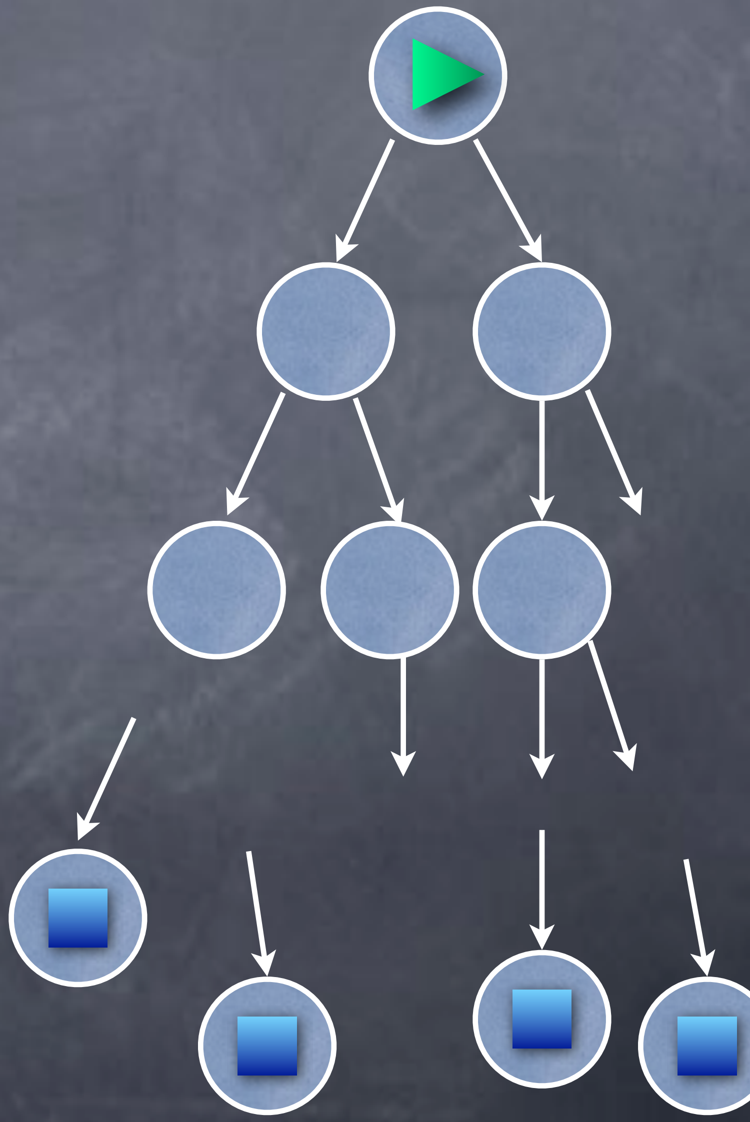
# Non-deterministic Computation

- Not "realistic" as a computation model, but has realistic interpretations (coming up)



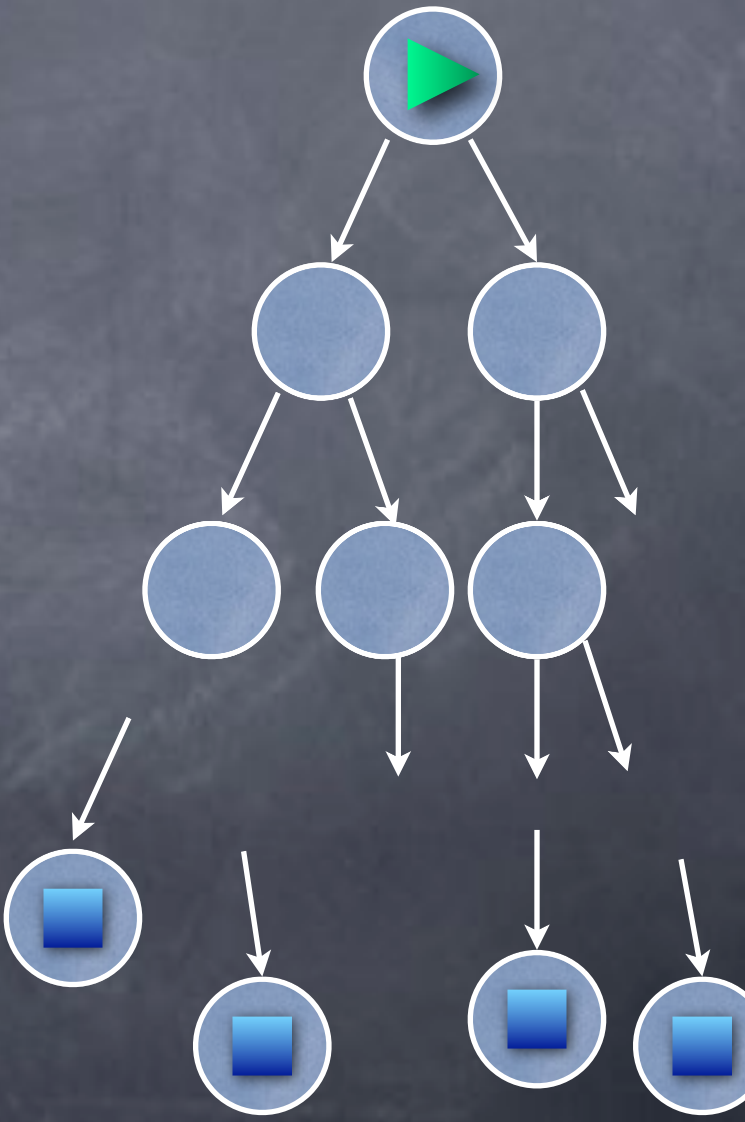
# Non-deterministic Computation

- Not "realistic" as a computation model, but has realistic interpretations (coming up)
- An NTM is said to accept an input if <u>any</u> of the threads of execution accepts it



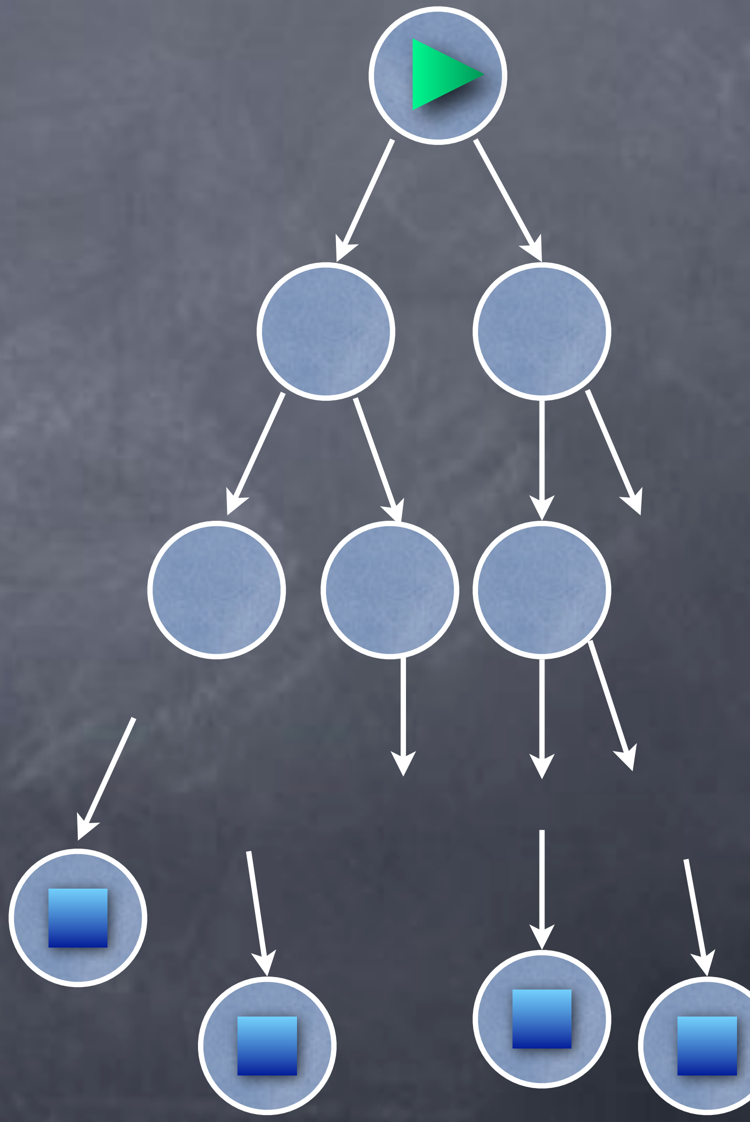
# Non-deterministic Computation

- Not "realistic" as a computation model, but has realistic interpretations (coming up)
- An NTM is said to accept an input if <u>any</u> of the threads of execution accepts it
- Time: longest execution thread



# Non-deterministic Computation

- Not "realistic" as a computation model, but has realistic interpretations (coming up)
- An NTM is said to accept an input if any of the threads of execution accepts it
- Time: longest execution thread
- L $\in$ NTIME(T): an NTM decides L in time at most T



# NTIME(T): alt view

# NTIME(T): alt view

- L is in NTIME(T) iff it can be defined in the following way:

# NTIME(T): alt view

- L is in NTIME(T) iff it can be defined in the following way:
  - $L = \{ x \mid \exists w \text{ s.t. } (x,w) \in L' \}$

# NTIME(T): alt view

- L is in NTIME(T) iff it can be defined in the following way:
  - $L = \{ x \mid \exists\ w \text{ s.t. } (x,w) \in L' \}$
  - Where $L'$ is in $DTIME(T(|x|))$ (with an extra read-once input tape for w)

# NTIME(T): alt view

- L is in NTIME(T) iff it can be defined in the following way:
  - $L = \{ x \mid \exists w \text{ s.t. } (x,w) \in L' \}$ non-std notation
  - Where L' is in DTIME(T(|x|)) (with an extra read-once input tape for w)

# NTIME(T): alt view

- L is in NTIME(T) iff it can be defined in the following way:
  - $L = \{ x \mid \exists\, w \text{ s.t. } (x,w) \in L' \}$ non-std notation
  - Where L' is in DTIME(T(|x|)) (with an extra read-once input tape for w)
- i.e., in time T, deterministic TM for L' can verify a certificate of membership for L

# NTIME(T): alt view

- L is in NTIME(T) iff it can be defined in the following way:
  - $L = \{ x \mid \exists\, w \text{ s.t. } (x,w) \in L' \}$ non-std notation
  - Where L' is in DTIME(T(|x|)) (with an extra read-once input tape for w)
- i.e., in time T, deterministic TM for L' can verify a certificate of membership for L
  - Finding a certificate (or even finding if there exists a certificate) may take longer

# $L \in \mathrm{NTIME}(T)$: Equivalent views

# $L \in NTIME(T)$: Equivalent views

- Non-deterministic M

# $L \in \mathrm{NTIME}(T)$: Equivalent views

- Non-deterministic M
- input: x

# $L \in$ NTIME(T): Equivalent views

- Non-deterministic M
- input: x
- makes non-det choices

# $L \in \mathrm{NTIME}(T)$: Equivalent views

- Non-deterministic M
- input: x
- makes non-det choices
- $x \in L$ iff some thread of M accepts

# $L \in \mathrm{NTIME}(T)$: Equivalent views

- Non-deterministic M
- input: x
- makes non-det choices
- $x \in L$ iff some thread of M accepts
- in at most $T(|x|)$ steps

# $L \in NTIME(T)$: Equivalent views

- Non-deterministic M
- input: x
- makes non-det choices
- $x \in L$ iff some thread of M accepts
- in at most $T(|x|)$ steps

- Deterministic M′

# $L \in NTIME(T)$: Equivalent views

- Non-deterministic M
- input: x
- makes non-det choices
- $x \in L$ iff some thread of M accepts
- in at most $T(|x|)$ steps

- Deterministic M'
- input: x and cert. w

# $L \in$ NTIME(T): Equivalent views

- Non-deterministic M
- input: x
- makes non-det choices
- $x \in L$ iff some thread of M accepts
- in at most $T(|x|)$ steps

- Deterministic M'
- input: x and cert. w
- reads bits from the cert.

# $L \in \mathrm{NTIME}(T)$: Equivalent views

- Non-deterministic M
- input: x
- makes non-det choices
- $x \in L$ iff some thread of M accepts
- in at most $T(|x|)$ steps

- Deterministic M′
- input: x and cert. w
- reads bits from the cert.
- $x \in L$ iff for some cert. w, M′ accepts

# $L \in NTIME(T)$: Equivalent views

- Non-deterministic M
- input: x
- makes non-det choices
- $x \in L$ iff some thread of M accepts
- in at most $T(|x|)$ steps

- Deterministic M′
- input: x and cert. w
- reads bits from the cert.
- $x \in L$ iff for some cert. w, M′ accepts
- in at most $T(|x|)$ steps

# $L \in \mathrm{NTIME}(T)$: Equivalent views

- Non-deterministic M
- input: x
- makes non-det choices
- $x \in L$ iff some thread of M accepts
- in at most $T(|x|)$ steps

- Deterministic M′
- input: x and cert. w
- reads bits from the cert.
- $x \in L$ iff for some cert. w, M′ accepts
- in at most $T(|x|)$ steps

# NP

# NP

- $NP = \bigcup_{a,b,c > 0} NTIME(a.n^c+b)$

# NP

- NP = $\bigcup_{a,b,c > 0}$ NTIME($a.n^c+b$)
    - L is in NP if there's an NTM that decides L in polynomial time (some fixed polynomial)

# NP

- $NP = \bigcup_{a,b,c > 0} NTIME(a.n^c+b)$
  - L is in NP if there's an NTM that decides L in polynomial time (some fixed polynomial)
- L is in NP if there's a TM that verifies certificates for membership in L, in polynomial time

# NP

- $NP = \bigcup_{a,b,c > 0} NTIME(a.n^c+b)$
  - L is in NP if there's an NTM that decides L in polynomial time (some fixed polynomial)
- L is in NP if there's a TM that verifies certificates for membership in L, in polynomial time
  - Recall: polynomial in size of x, not of (x,w)

# NP

- $NP = \bigcup_{a,b,c > 0} NTIME(a.n^c+b)$
  - L is in NP if there's an NTM that decides L in polynomial time (some fixed polynomial)

- L is in NP if there's a TM that verifies certificates for membership in L, in polynomial time
  - Recall: polynomial in size of x, not of (x,w)
  - Or, $L = \{x \mid \exists w, |w| = O(poly(|x|)) \text{ s.t. } (x,w) \in L'\}$, and L' in P

# NP

- $NP = \cup_{a,b,c > 0} NTIME(a.n^c+b)$
  - L is in NP if there's an NTM that decides L in polynomial time (some fixed polynomial)
- L is in NP if there's a TM that verifies certificates for membership in L, in polynomial time
  - Recall: polynomial in size of x, not of (x,w)
  - Or, $L = \{x \mid \exists w, |w| = O(poly(|x|)) \text{ s.t. } (x,w) \in L'\}$, and L' in P
- Note: Completeness and soundness

# Some Problems in NP

# Some Problems in NP

- Graph properties: has a clique of size n/2, has a "Hamiltonian cycle", graph has an "Eulerian tour", two graphs are isomorphic

# Some Problems in NP

- Graph properties: has a clique of size n/2, has a "Hamiltonian cycle", graph has an "Eulerian tour", two graphs are isomorphic
- Numerical properties: is a composite number, is a prime number (not obvious)

# Some Problems in NP

- Graph properties: has a clique of size n/2, has a "Hamiltonian cycle", graph has an "Eulerian tour", two graphs are isomorphic
- Numerical properties: is a composite number, is a prime number (not obvious)
- Constraint satisfaction: equation has solution, Linear Program (LP) is feasible, Integer LP is feasible, has a short Traveling Salesperson tour

# Some Problems in NP

- Graph properties: has a clique of size n/2, has a "Hamiltonian cycle", graph has an "Eulerian tour", two graphs are isomorphic
- Numerical properties: is a composite number, is a prime number (not obvious)
- Constraint satisfaction: equation has solution, Linear Program (LP) is feasible, Integer LP is feasible, has a short Traveling Salesperson tour
- All problems in P (empty certificate)

# Search using Decision

# Search using Decision

- Suppose given "oracles" for deciding all NP languages, can we easily find certificates?

# Search using Decision

- Suppose given "oracles" for deciding all NP languages, can we easily find certificates?
  - Yes! So, if decision easy (decision-oracles realizable), then search is easy too!

# Search using Decision

- Suppose given "oracles" for deciding all NP languages, can we easily find certificates?
  - Yes! So, if decision easy (decision-oracles realizable), then search is easy too!
- Say, given x, need to find w s.t. $(x,w) \in L'$ (if such w exists)

# Search using Decision

- Suppose given "oracles" for deciding all NP languages, can we easily find certificates?
  - Yes! So, if decision easy (decision-oracles realizable), then search is easy too!
- Say, given x, need to find w s.t. $(x,w) \in L'$ (if such w exists)
  - consider $L_1$ in NP: $(x,y) \in L_1$ iff $\exists z$ s.t. $(x,yz) \in L'$. (i.e., can y be a prefix of a certificate for x).

# Search using Decision

- Suppose given "oracles" for deciding all NP languages, can we easily find certificates?
  - Yes! So, if decision easy (decision-oracles realizable), then search is easy too!
- Say, given x, need to find w s.t. $(x,w) \in L'$ (if such w exists)
  - consider $L_1$ in NP: $(x,y) \in L_1$ iff $\exists z$ s.t. $(x,yz) \in L'$. (i.e., can y be a prefix of a certificate for x).
  - Query $L_1$-oracle with (x,0) and (x,1). If $\exists w$, one of the two must be positive: say $(x,0) \in L_1$; then first bit of w be 0.

# Search using Decision

- Suppose given "oracles" for deciding all NP languages, can we easily find certificates?
  - Yes! So, if decision easy (decision-oracles realizable), then search is easy too!
- Say, given x, need to find w s.t. $(x,w) \in L'$ (if such w exists)
  - consider $L_1$ in NP: $(x,y) \in L_1$ iff $\exists z$ s.t. $(x,yz) \in L'$. (i.e., can y be a prefix of a certificate for x).
  - Query $L_1$-oracle with (x,0) and (x,1). If $\exists w$, one of the two must be positive: say $(x,0) \in L_1$; then first bit of w be 0.
  - For next bit query $L_1$-oracle with (x,00) and (x,01)

# What if NP = P

# What if NP = P

- "Can find as efficiently as can verify" (broadly speaking)

# What if NP = P

- "Can find as efficiently as can verify" (broadly speaking)
- Mathematics: Proofs are easy to verify efficiently (if written in full). So we can generate them too efficiently?! Prove/discover theorems mechanically!

# What if NP = P

- "Can find as efficiently as can verify" (broadly speaking)
- Mathematics: Proofs are easy to verify efficiently (if written in full). So we can generate them too efficiently?! Prove/discover theorems mechanically!
- Cryptography: If someone's private key (well, key generation info) given, can verify that it corresponds to a public key. So we can find the private key efficiently?! No public-key crypto!

# What if NP = P

- "Can find as efficiently as can verify" (broadly speaking)
- Mathematics: Proofs are easy to verify efficiently (if written in full). So we can generate them too efficiently?! Prove/discover theorems mechanically!
- Cryptography: If someone's private key (well, key generation info) given, can verify that it corresponds to a public key. So we can find the private key efficiently?! No public-key crypto!
- Solve all sorts of optimization problems efficiently!

# EXP and NEXP

# EXP and NEXP

- EXP is DTIME($2^{poly(n)}$):

# EXP and NEXP

- EXP is DTIME($2^{poly(n)}$):
  - EXP = $\bigcup_{a,b,c > 0}$ DTIME($2^{an^c+b}$)

# EXP and NEXP

- EXP is DTIME($2^{poly(n)}$):
  - EXP = $\cup_{a,b,c>0}$ DTIME($2^{an^c+b}$)
- NEXP is NTIME($2^{poly(n)}$):

# EXP and NEXP

- EXP is DTIME($2^{poly(n)}$):
  - EXP = $\cup_{a,b,c>0}$ DTIME($2^{an^c+b}$)
- NEXP is NTIME($2^{poly(n)}$):
  - NEXP = $\cup_{a,b,c>0}$ NTIME($2^{an^c+b}$)

# EXP and NEXP

- EXP is DTIME($2^{poly(n)}$):
  - EXP = $\cup_{a,b,c > 0}$ DTIME($2^{an^c+b}$)
- NEXP is NTIME($2^{poly(n)}$):
  - NEXP = $\cup_{a,b,c > 0}$ NTIME($2^{an^c+b}$)
  - NEXP = all L of the form:

# EXP and NEXP

- EXP is DTIME($2^{poly(n)}$):
  - EXP = $\bigcup_{a,b,c > 0}$ DTIME($2^{an^c+b}$)
- NEXP is NTIME($2^{poly(n)}$):
  - NEXP = $\bigcup_{a,b,c > 0}$ NTIME($2^{an^c+b}$)
  - NEXP = all L of the form:
    - L = $\{x \mid \exists w, |w| = O(2^{poly(|x|)})$ s.t. $(x,w) \in L'\}$, and L′ in EXP?

# EXP and NEXP

- EXP is DTIME($2^{poly(n)}$):
  - EXP = $\cup_{a,b,c > 0}$ DTIME($2^{an^c+b}$)
- NEXP is NTIME($2^{poly(n)}$):
  - NEXP = $\cup_{a,b,c > 0}$ NTIME($2^{an^c+b}$)
  - NEXP = all L of the form:
    - L = {x | $\exists$w, |w| = O($2^{poly(|x|)}$) s.t. (x,w) $\in$ L'}, and L' in EXP?
      - No! L' in DTIME($2^{poly(|x|)}$)

# EXP and NEXP

- EXP is $DTIME(2^{poly(n)})$:
  - $EXP = \cup_{a,b,c > 0} DTIME(2^{an^c+b})$
- NEXP is $NTIME(2^{poly(n)})$:
  - $NEXP = \cup_{a,b,c > 0} NTIME(2^{an^c+b})$
  - NEXP = all L of the form:
    - $L = \{x \mid \exists w, |w| = O(2^{poly(|x|)}) \text{ s.t. } (x,w) \in L'\}$, and L' in EXP?
      - No! L' in $DTIME(2^{poly(|x|)})$
        - i.e., L' in P

# co-Class

# co-Class

- co-X = { L | $L^c$ is in X } (where $L^c$ = { x | x$\notin$L } )

# co-Class

- co-X = { L | $L^c$ is in X } (where $L^c$ = { x | x$\notin$L } )
- co-DTIME(T) = DTIME(T)

# co-Class

- co-X = { L | $L^c$ is in X } (where $L^c$ = { x | $x \notin L$ } )
- co-DTIME(T) = DTIME(T)
    - $L^c$ in DTIME(T) iff L in DTIME(T)

# co-Class

- co-X = { L | $L^c$ is in X } (where $L^c$ = { x | $x \notin L$ } )
- co-DTIME(T) = DTIME(T)
  - $L^c$ in DTIME(T) iff L in DTIME(T)
  - $M_{L^c} \leftrightarrow M_L$: flip accept/reject states

# co-Class

- co-X = { L | $L^c$ is in X } (where $L^c$ = { x | $x \notin L$ } )
- co-DTIME(T) = DTIME(T)
  - $L^c$ in DTIME(T) iff L in DTIME(T)
  - $M_{L^c} \leftrightarrow M_L$: flip accept/reject states
- co-NTIME(T): all L s.t. $L^c$ is in NTIME(T)

# co-Class

- co-X = { L | $L^c$ is in X } (where $L^c$ = { x | $x \notin L$ } )
- co-DTIME(T) = DTIME(T)
  - $L^c$ in DTIME(T) iff L in DTIME(T)
  - $M_{L^c} \leftrightarrow M_L$: flip accept/reject states
- co-NTIME(T): all L s.t. $L^c$ is in NTIME(T)
  - $M_{L^c} \leftrightarrow M_L$?

# co-Class

- co-X = { L | $L^c$ is in X } (where $L^c$ = { x | x$\notin$L } )
- co-DTIME(T) = DTIME(T)
  - $L^c$ in DTIME(T) iff L in DTIME(T)
  - $M_{L^c} \leftrightarrow M_L$: flip accept/reject states
- co-NTIME(T): all L s.t. $L^c$ is in NTIME(T)
  - $M_{L^c} \leftrightarrow M_L$?
    - flip accept/reject states <u>and</u> flip "there exists" and "for all" in the acceptance criterion (NTM $\leftrightarrow$ "co-NTM")

# co-Class

- co-X = { L | $L^c$ is in X } (where $L^c = \{ x \mid x \notin L \}$ )
- co-DTIME(T) = DTIME(T)
  - $L^c$ in DTIME(T) iff L in DTIME(T)
  - $M_{L^c} \leftrightarrow M_L$: flip accept/reject states
- co-NTIME(T): all L s.t. $L^c$ is in NTIME(T)
  - $M_{L^c} \leftrightarrow M_L$?
    - flip accept/reject states and flip "there exists" and "for all" in the acceptance criterion (NTM $\leftrightarrow$ "co-NTM")
  - $L^c = \{ x \mid \nexists w \text{ s.t. } (x,w) \in L' \} = \{ x \mid \forall w\ (x,w) \in L'^c \}$

# co-Class

- co-X = { L | $L^c$ is in X } (where $L^c = \{ x \mid x \notin L \}$ )
- co-DTIME(T) = DTIME(T)
  - $L^c$ in DTIME(T) iff L in DTIME(T)
  - $M_{L^c} \leftrightarrow M_L$: flip accept/reject states
- co-NTIME(T): all L s.t. $L^c$ is in NTIME(T)
  - $M_{L^c} \leftrightarrow M_L$?

no counter-example

  - flip accept/reject states and flip "there exists" and "for all" in the acceptance criterion (NTM $\leftrightarrow$ "co-NTM")
  - $L^c = \{ x \mid \nexists w \text{ s.t. } (x,w) \in L' \} = \{ x \mid \forall w\ (x,w) \in L'^c \}$

# P, NP and co-NP

# P, NP and co-NP

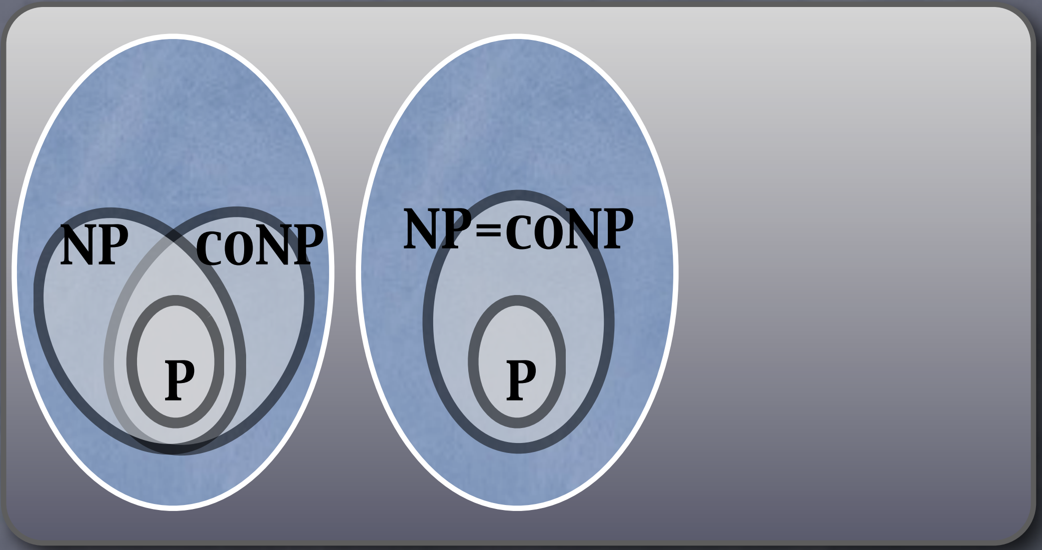
- Different possibilities

# P, NP and co-NP

- Different possibilities

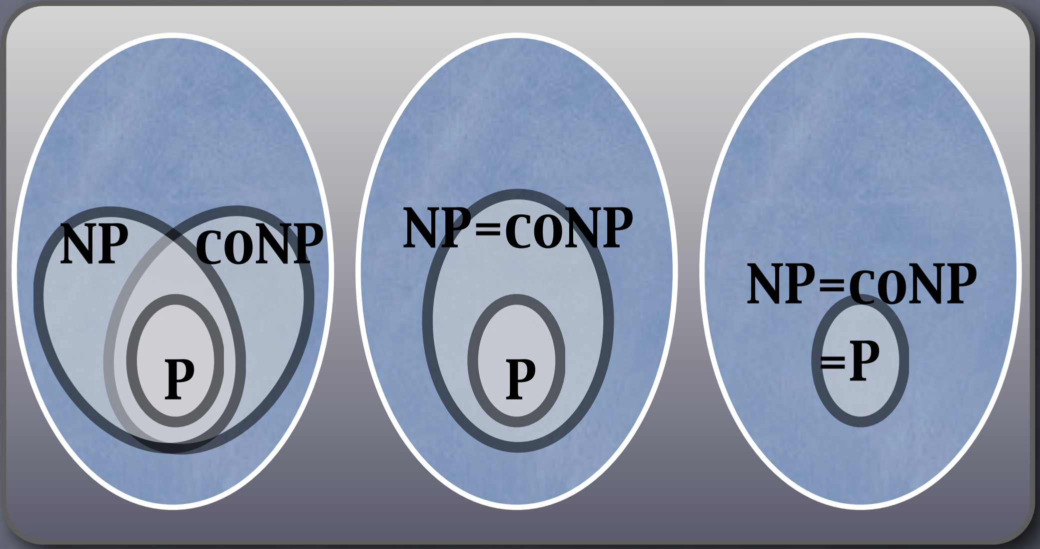
# P, NP and co-NP

- Different possibilities



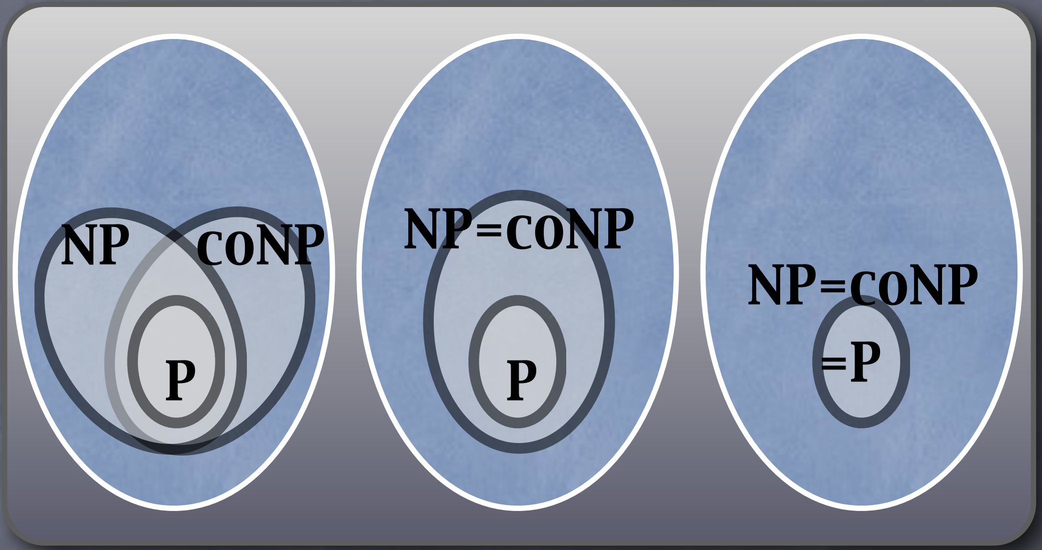
# P, NP and co-NP

- Different possibilities



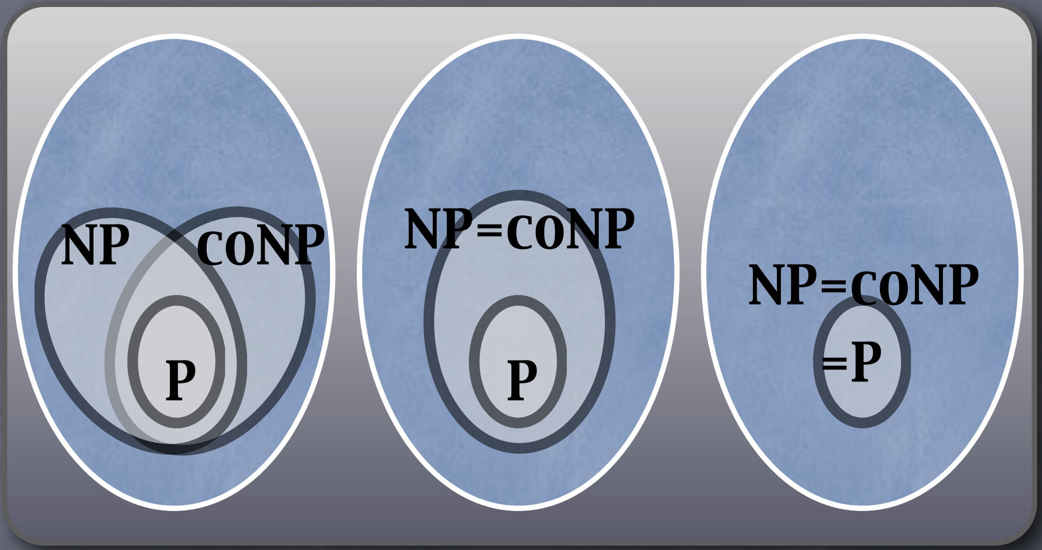
# P, NP and co-NP

- Different possibilities
- If P=NP, then

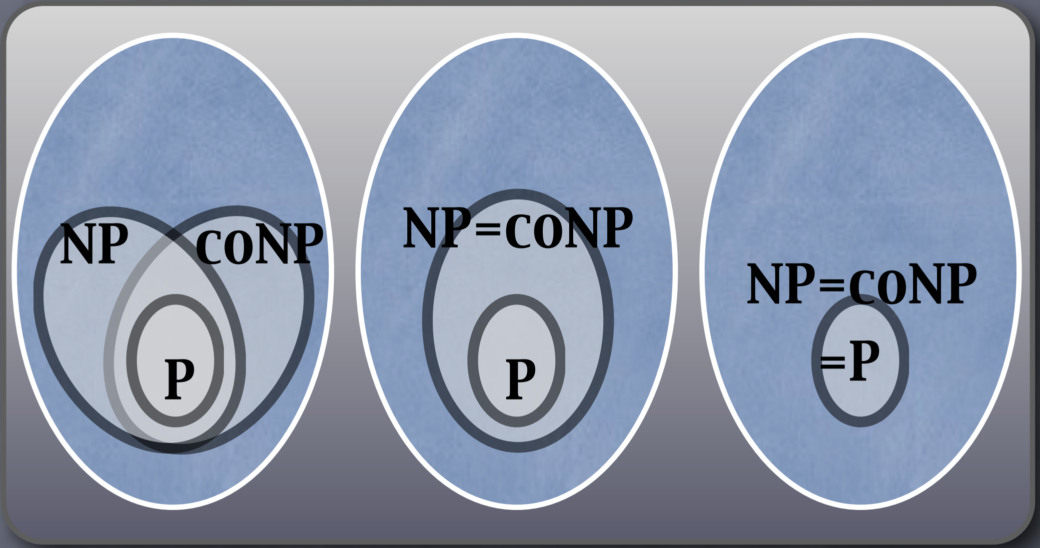


# P, NP and co-NP

- Different possibilities
- If P=NP, then
  - coNP = coP = P = NP

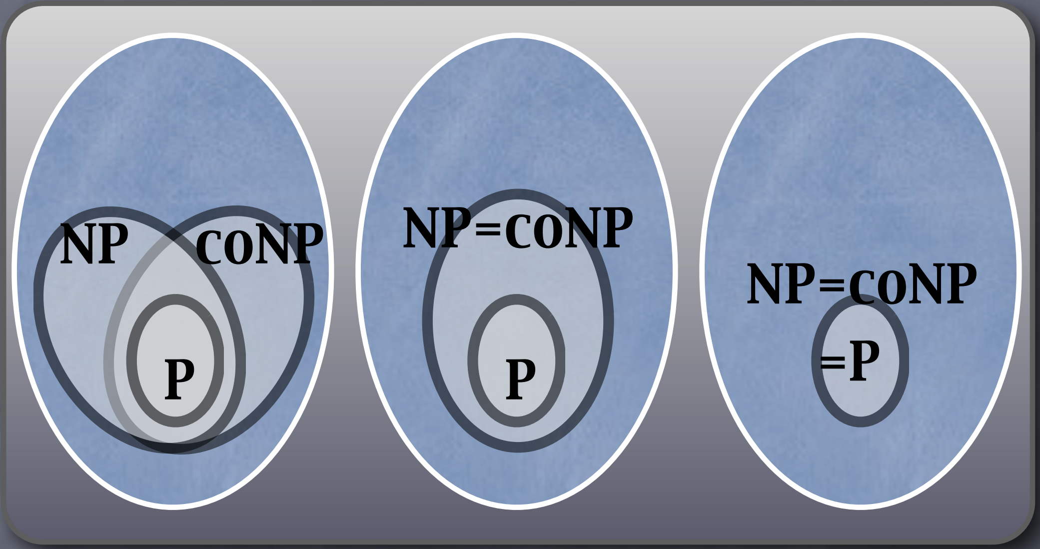


# P, NP and co-NP



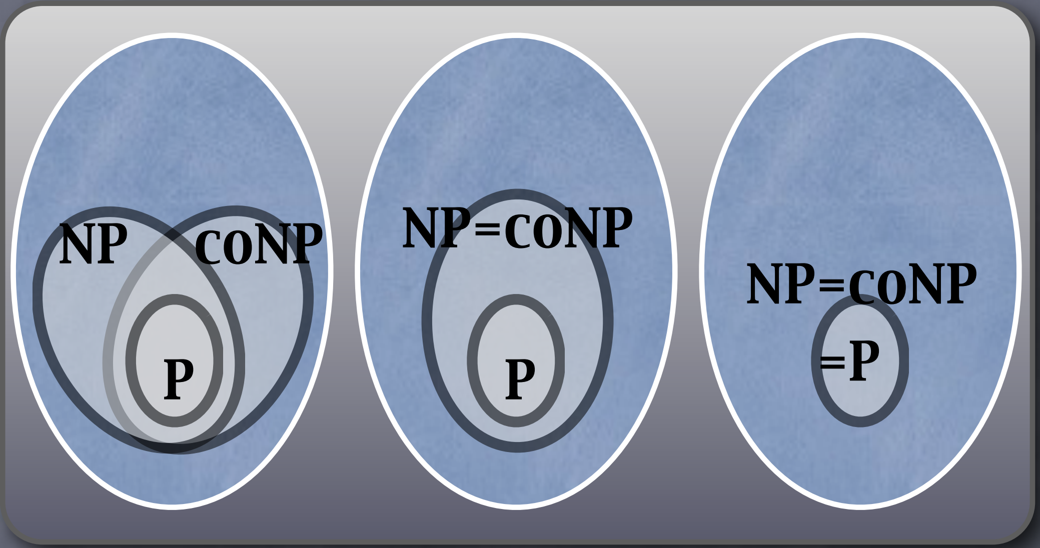

- Different possibilities
- If P=NP, then
  - coNP = coP = P = NP
  - Also, EXP = NEXP [Exercise]

# P, NP and co-NP



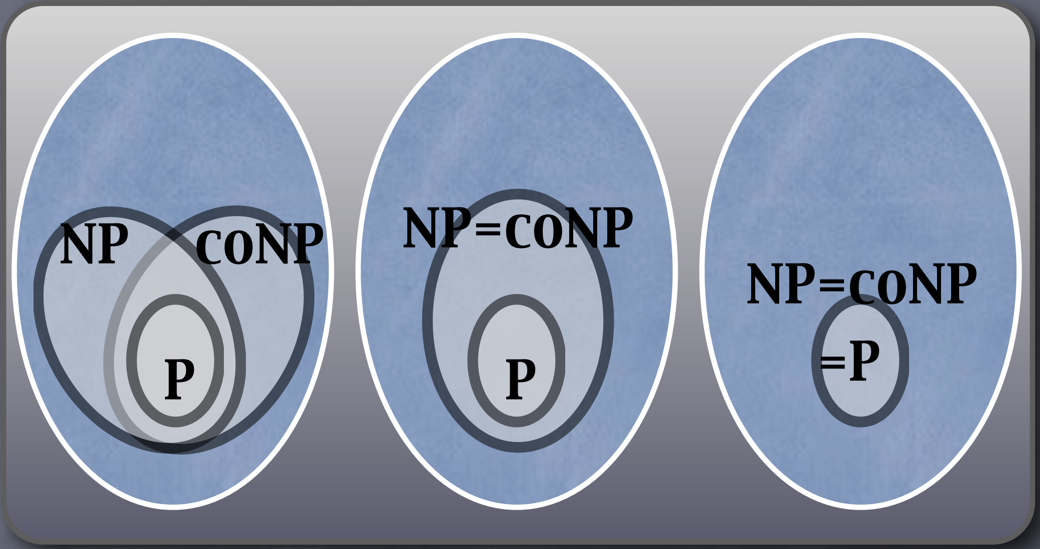

- Different possibilities
- If P=NP, then
  - coNP = coP = P = NP
  - Also, EXP = NEXP [Exercise]
    - padding to scale up both classes

# P, NP and co-NP




- Different possibilities
- If P=NP, then
  - coNP = coP = P = NP
  - Also, EXP = NEXP [Exercise]
    - padding to scale up both classes
    - x → (x,pad), so that Exp(|x|) = Poly(|x,pad|)

# P, NP and co-NP




- Different possibilities
- If P=NP, then
  - coNP = coP = P = NP
  - Also, EXP = NEXP [Exercise]
    - padding to scale up both classes
    - x → (x,pad), so that Exp(|x|) = Poly(|x,pad|)
- If P=NP, then the complexity landscape would get greatly simplified than believed (more later)

# Today

# Today

- DTIME

# Today

- DTIME
  - P, EXP

# Today

- DTIME
  - P, EXP
- NTIME

# Today

- DTIME
  - P, EXP
- NTIME
  - Two views: non-determinism and certificate

# Today

- DTIME
  - P, EXP
- NTIME
  - Two views: non-determinism and certificate
  - NP, NEXP

# Today

- DTIME
  - P, EXP
- NTIME
  - Two views: non-determinism and certificate
  - NP, NEXP
- co-NTIME

# Today

- DTIME
  - P, EXP
- NTIME
  - Two views: non-determinism and certificate
  - NP, NEXP
- co-NTIME
  - Two views: co-NTM and "no counter-example"

# Next ~~Class~~ Lecture

# Next ~~Class~~ Lecture

- NP completeness

# Next ~~Class~~ Lecture

- NP completeness
  - As hard as it gets inside NP

# Next ~~Class~~ Lecture

- NP completeness
  - As hard as it gets inside NP
    - a la reductions (of course)