Polynomial time Verification

P = set of decision problems solvable in polynomial time

Hamiltonian Cycle $\in$ P? we don't know!

Polynomial-time verification algorithm for decision problem X:

input to solution for X is x $\longrightarrow$ input for verification algorithm for X is $x, \underbrace{y}_{\text{certificate/evidence that } X(x) \text{ is YES}}$

X(x) = YES $\longrightarrow$ there is a y s.t. size(y) is polynomial in size(X) and X-VERIFY(x,y) = YES

X(x) = NO $\longrightarrow$ all y make X-VERIFY(x,y) = NO

X(x) = YES if and only if there is a y s.t. size(y) poly in size(X) and X-VERIFY(x,y) = YES

Polynomial-time Verification for HC

certificate/evidence that $X(x)$ is YES

Polynomial-time verification algorithm for decision problem X.

input to solution for X is x → input for verification algorithm for X is x, y

$X(x) = \text{YES}$ if and only if there is a y s.t. size(y) poly in size(X)
and X-VERIFY(x,y) = YES

sequence of vertices in G

HC-VERIFY(G, p)

poly check that each v in G appears at least one in p
poly check that each v in G appears at most once in p
poly for each $(v_i, v_j) \in p$, check that edge (v_i, v_j) exists in G
and edge from last to first exists in G
if all YES, return YES
else return NO

if G has Hamiltonian cycle $\overbrace{c}^{\text{length}=n}$, then HC-VERIFY(G, c) = YES

if G has no Hamiltonian cycle, then HC-VERIFY(G, p) = NO for all p

P = set of decision problems solvable in polynomial time

NP = set of decision problems with polynomial-time verification algorithms

$HC \in NP$ $\qquad$ $HC \overset{?}{\in} P$ don't know (brute force checks all $n!$ permutations of vertices)

P = set of decision problems solvable in polynomial time

NP = set of decision problems with polynomial-time verification algorithms

$HC \in NP$ $HC \overset{?}{\in} P$ don't know (brute force checks all $n!$ permutations of vertices)

Travelling Salesperson: given fully connected, undirected, weighted G, and bound k, determine if G has a tour of total weight $\le k$

$TSP \overset{?}{\in} P$ no one knows $TSP \in NP$: TSP-VERIFY(G, k, p)

P = set of decision problems solvable in polynomial time

NP = set of decision problems with polynomial-time verification algorithms

$HC \in NP$ $\qquad\qquad$ $HC \overset{?}{\in} P$ don't know (brute force checks all $n!$ permutations of vertices)

Travelling Salesperson: given fully connected, undirected, weighted G, and bound k, determine if G has a tour of total weight $\le k$

$TSP \overset{?}{\in} P$ no one knows $\qquad\qquad$ $TSP \in NP$:

TSP-VERIFY(G, k, p)

```
if p is a tour                          poly
    t ← Σ_{(u,v)∈p} w(u,v)              poly
    if t ≤ k return YES                 poly
return NO
```

if G has tour c of total weight $\le k$, TSP-VERIFY(G,k,c) = YES

if G has no tour of total weight $\le k$, TSP-VERIFY(G,k,p) = NO for all p

NP Problems

uses variables, and, or, not, ( )　　　　assignment of T/F to vars to make φ T

SAT: given $\underbrace{\text{Boolean formula}}$ φ, determine if φ has a $\underbrace{\text{satisfying assignment}}$

has satisfying assignment: $\underset{(T \wedge T)}{(x \wedge \sim y)} \vee \underset{(F \wedge F)}{(\sim x \wedge y)}$　$x=T$ $y=F$

$\underset{(T \vee F \vee F)}{(x \vee y \vee z)} \wedge \underset{(F \vee (T \wedge T))}{(\sim x \vee (\sim y \wedge \sim z))}$　$x=T$ $y=F$ $z=F$

no satisfying assignment: $x \wedge \sim x$

$(x \vee y) \wedge (\sim y \vee z) \wedge (x \vee \sim z) \wedge (\sim x \vee y) \wedge (\sim x \vee z)$

brute force: check all 2^{n} possible assignments (n = # of variables)　　SAT $\overset{?}{\in}$ P　no one knows

NP Problems

uses variables, and, or, not, ( )   assignment of T/F to vars to make φ T

SAT: given $\underbrace{\text{Boolean formula}}$ φ, determine if φ has a $\underbrace{\text{satisfying assignment}}$

SAT $\in$ NP: SAT-VERIFY(φ, A) ← assignment of T/F to each variable

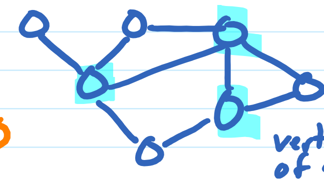
substitute T/F into φ according to A   poly
result ← value of resulting expression   poly
if result = T then return YES
else return NO

size = # variables in φ = polynomial in size of φ

If φ has a satisfying assignment A, then SAT-VERIFY(φ, A) = YES

If φ has no satisfying assignment, then SAT-VERIFY(φ, A) = NO for all A subset of vertices C s.t. all edges have ≥ 1 endpoint in C

VERTEX-COVER: Given undirected G and k, is there a vertex cover C with $|C| \leq k$?

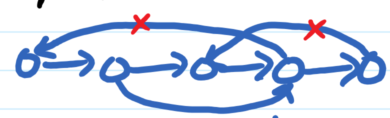


vertex cover of size 3, none of size 2

VC ∈ NP: VC-VERIFY(G, k, C)

```
poly   if C ⊆ V and |C| ≤ k
poly       for each edge (u,v)
poly           if u ∉ S and v ∉ C return NO
           return YES
       return NO
```

FEEDBACK-ARC-SET: Given directed G and k, is there a subset of edges E' s.t. $|E'| \leq k$ and $G' = G$ with edges in E' removed is acyclic



can remove 2 edges to make acyclic; removing only 1 always leaves a cycle

FAS ∈ NP: FAS-VERIFY(G, k, E')

```
poly        if E' ⊆ E and |E'| ≤ k
poly (DFS)      G' ← G with edges in G' removed
poly            return ~HAS-CYCLE(G')
            return NO
```

P=NP?

$P = NP$? ← no one knows

$P \subseteq NP$ is $NP \subseteq P$? does every problem with a poly-time verification alg have a poly-time solution

NP-complete : X is NP-complete if it is among the hardest problems in NP

Polynomial time Verification

P = set of decision problems solvable in polynomial time

Hamiltonian Cycle $\in$ P? we don't know!

Polynomial-time verification algorithm for decision problem X:

input to solution for X is x $\rightarrow$ input for verification algorithm for X is $x, \underbrace{y}$

certificate/evidence that X(x) is YES

X(x) = YES $\longrightarrow$ there is a y s.t. size(y) is polynomial in size(X) and X-VERIFY(x,y) = YES

X(x) = NO $\longrightarrow$ all y make X-VERIFY(x,y) = NO

X(x) = YES if and only if there is a y s.t. size(y) poly in size(X) and X-VERIFY(x,y) = YES certificate/evidence that $X(x)$ is YES

Polynomial-time verification algorithm for decision problem X.

input to solution for X is x → input for verification algorithm for X is x, y

$X(x) = $ YES if and only if there is a y s.t. size(y) poly in size(X)
and X-VERIFY(x,y) = YES

sequence of vertices in G

HC-VERIFY(G, p)

poly check that each v in G appears at least once in p
poly check that each v in G appears at most once in p
poly for each $(v_i, v_j) \in p$, check that edge (v_i, v_j) exists in G

if all YES, return YES
else return NO

length = n

if G has Hamiltonian cycle c, then HC-VERIFY(G, c) = YES

if G has no Hamiltonian cycle, then HC-VERIFY(G, p) = NO for all p

P = set of decision problems solvable in polynomial time

NP = set of decision problems with polynomial-time verification algorithms

HC $\in$ NP

HC $\overset{?}{\in}$ P don't know (brute force checks all $n!$ permutations of vertices)

Travelling Salesperson: given fully connected, undirected, weighted G, and bound k, determine if G has a tour of total weight $\le k$

TSP $\overset{?}{\in}$ P no one knows

TSP $\in$ NP: TSP-VERIFY(G, k, p)

if p is a tour — poly

$t \leftarrow \sum_{(u,v) \in p} w(u,v)$ — poly

if $t \le k$ return YES — poly

return NO

if G has tour c of total weight $\le k$, TSP-VERIFY(G,k,c) = YES

if G has no tour of total weight $\le k$, TSP-VERIFY(G,k,p) = NO for all p

uses variables, and, or, not, ( ) — assignment of T/F to vars to make φ T

SAT: given $\overbrace{\text{Boolean formula } \varphi}^{\text{uses variables, and, or, not, ( )}}$, determine if φ has a $\overbrace{\text{satisfying assignment}}^{\text{assignment of T/F to vars to make } \varphi \text{ T}}$

$(T \wedge T) \vee (F \wedge F)$

has satisfying assignment: $(x \wedge \sim y) \vee (\sim x \wedge y)$ $x=T$ $y=F$

$(T \vee F \vee F) \wedge (F \vee (T \wedge T))$

$(x \vee y \vee z) \wedge (\sim x \vee (\sim y \wedge \sim z))$ $x=T$ $y=F$ $z=F$

no satisfying assignment: $x \wedge \sim x$

$(x \vee y) \wedge (\sim y \vee z) \wedge (x \vee \sim z) \wedge (\sim x \vee y) \wedge (\sim x \vee \sim z)$

brute force: check all 2^{n} possible assignments (n = # of variables)

SAT $\overset{?}{\in}$ P no one knows

uses variables, and, or, not, ( )

assignment of T/F to vars to make φ T

SAT: given $\underbrace{\text{Boolean formula}}$ φ, determine if φ has a $\underbrace{\text{satisfying assignment}}$

SAT $\in$ NP: SAT-VERIFY(φ, A) ← assignment of T/F to each variable

substitute T/F into φ according to A — poly
result ← value of resulting expression — poly
if result = T then return YES
else return NO

size = # variables in φ = polynomial in size of φ

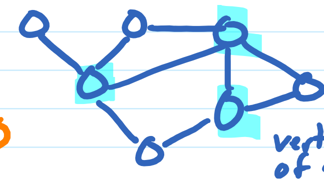
If φ has a satisfying assignment A, then SAT-VERIFY(φ, A) = YES

If φ has no satisfying assignment, then SAT-VERIFY(φ, A) = NO for all A subset of vertices C s.t. all edges have ≥ 1 endpoint in C

VERTEX-COVER: Given undirected G and k, is there a $\underbrace{\text{vertex cover}}$ C with $|C| \leq k$?

VC $\in$ NP:

```
       VC-VERIFY(G, k, C)
poly     if C ⊆ V and |C| ≤ k
poly        for each edge (u,v)
poly           if u ∉ S and v ∉ C return NO
            return YES
         return NO
```

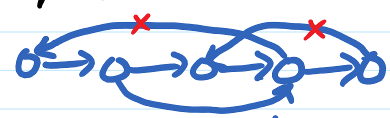


vertex cover of size 3, none of size 2

FEEDBACK-ARC-SET: Given directed G and k, is there a subset of edges E' s.t. $|E'| \leq k$ and $G' = G$ with edges in E' removed is acyclic

FAS $\in$ NP:

```
            FAS-VERIFY(G, k, E')
poly          if E' ⊆ E and |E'| ≤ k
poly (DFS)       G' ← G with edges in G' removed
poly             return ~HAS-CYCLE(G')
              return NO
```



can remove 2 edges to make acyclic; removing only 1 always leaves a cycle

P=NP?

$P \subseteq NP$ is $NP \subseteq P$? does every problem with a poly-time verification alg have a poly-time solution

NP-complete: X is NP-complete if it is among the hardest problems in NP