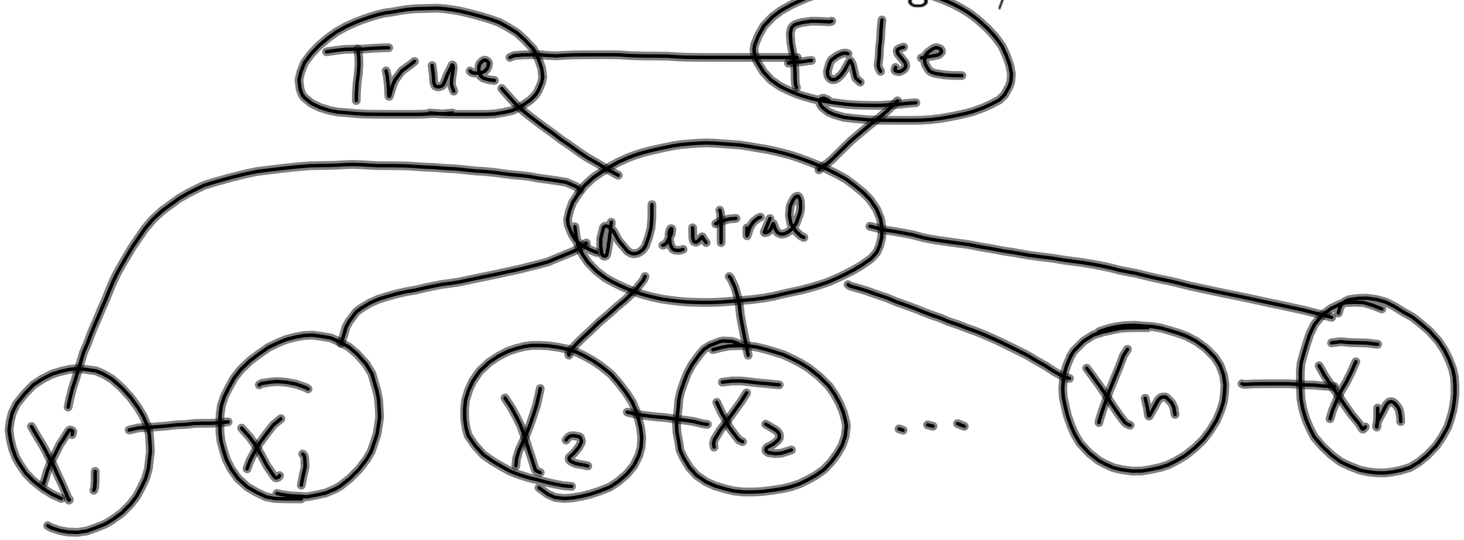
What did we see last time?

- 3-COLOR is in $\mathcal{NP}$.
- We can use a solution to 3-COLOR to assign T/F values:

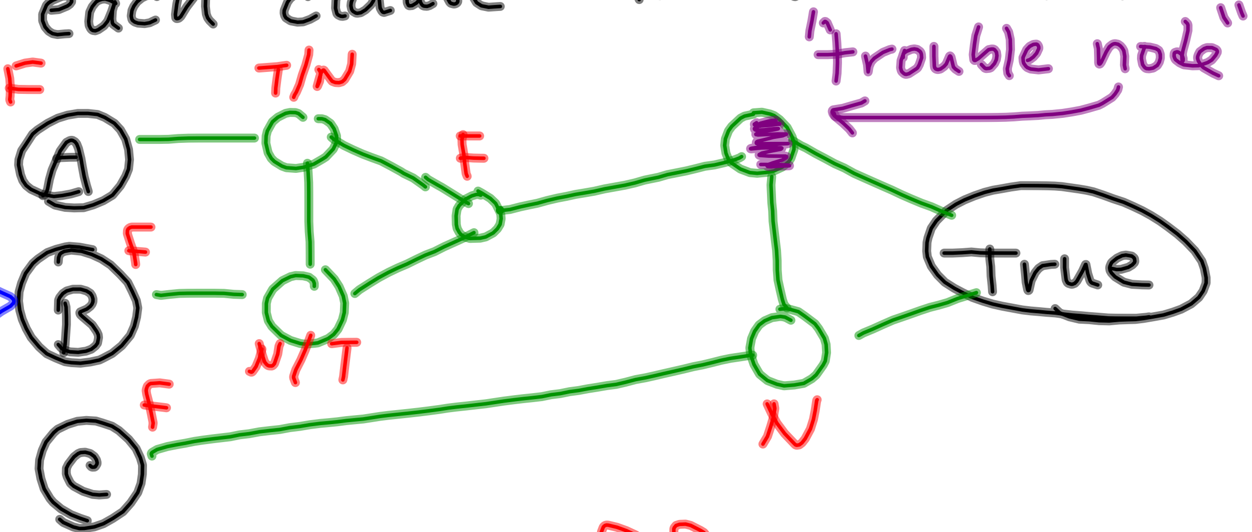


Amend the 3-Color usage to get a satisfying truth value assignment on n variables

$x_1 \vee \bar{x}_2 \vee \bar{x}_3$

for each clause: let vars A, B, C

same vertex x_2 as in last slide



What if A, B, C all F?

○: new

Putting it together...

Could this produce *false negatives*?
That is, could there be an instance of 3-SAT that has a satisfying assignment, but with the corresponding 3-COLOR (that we create) causing a "false"?

No: color 1st slide w/ TVA
each "clause widget"
is colorable
(as at least one var → true)

Putting it together...

Could that create *false positives*?
That is, if 3-Color returns true, do we really know that the corresponding 3-Sat instance has a satisfying assignment?

Ea clause not $F \vee F \vee F$
and no var is neutral

So we have a working TVA

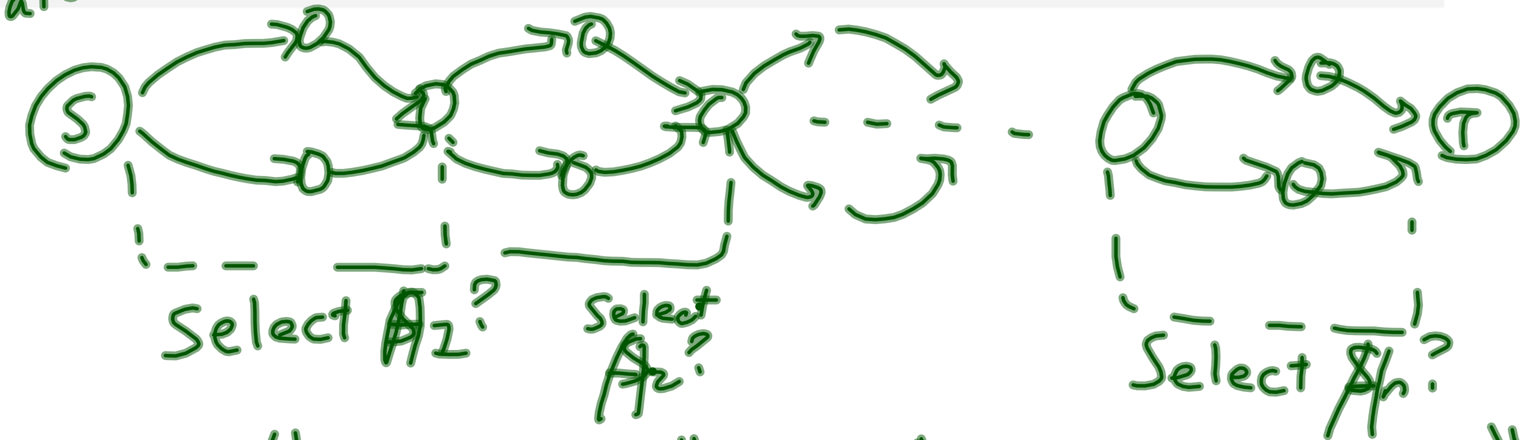
Prove that MIN-COST FAST PATH is $\mathcal{NP}$-complete

MIN-COST FAST PATH is in $\mathcal{NP}$:

- Certificate: path
- Verifier:
  1. If path not path $s \rightsquigarrow t$, reject
  2. add total time, cost on path
     if either > bound, reject
  3. accept

Prove that MIN-COST FAST PATH is $\mathcal{NP}$-complete

scratch



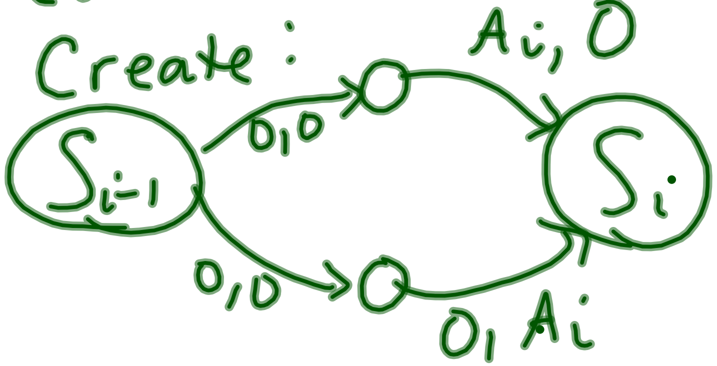
→ "add to T" ⇒ "add to $\Sigma - T$"

Subset Sum(A, T)

Create node S_0

for each i $1..n$

Create:



Designate S_n as dest ("t")

// Set $T' = T$ // "T'" is T for MCFP

Set $C = \sum_i A_i - T$

Prove that CLUSTERING is $\mathcal{NP}$-complete

CLUSTERING is in $\mathcal{NP}$:

- Certificate: Partitioning (K element sets, membership)
- Verifier:

1. If any elt not in any set, reject
2. If any elt in 2+ sets, reject
3. for each set
   if any pair has S.P. > T, reject
   shortest path
4. accept

$T=1$?

Prove that CLUSTERING is $\mathcal{NP}$-complete

// If $(u,v) \notin E$, dist$(u,v) > T$

// else dist$(u,v) \le T$

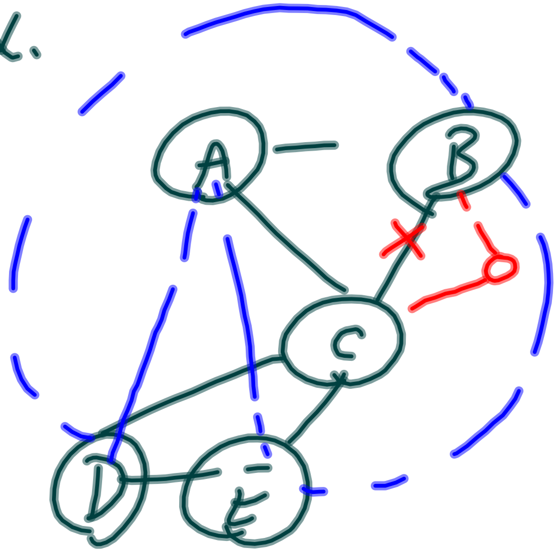
3-COLOR(G)
{
$G' =$ G.V plus ALL poss. edges ; G' is a complete Graph

def $C_e = \begin{cases} 2 & \text{if } e \in G \\ 1 & \text{otherwise} \end{cases}$

Cluster$(G', k=3, T=1)$

}

ex.

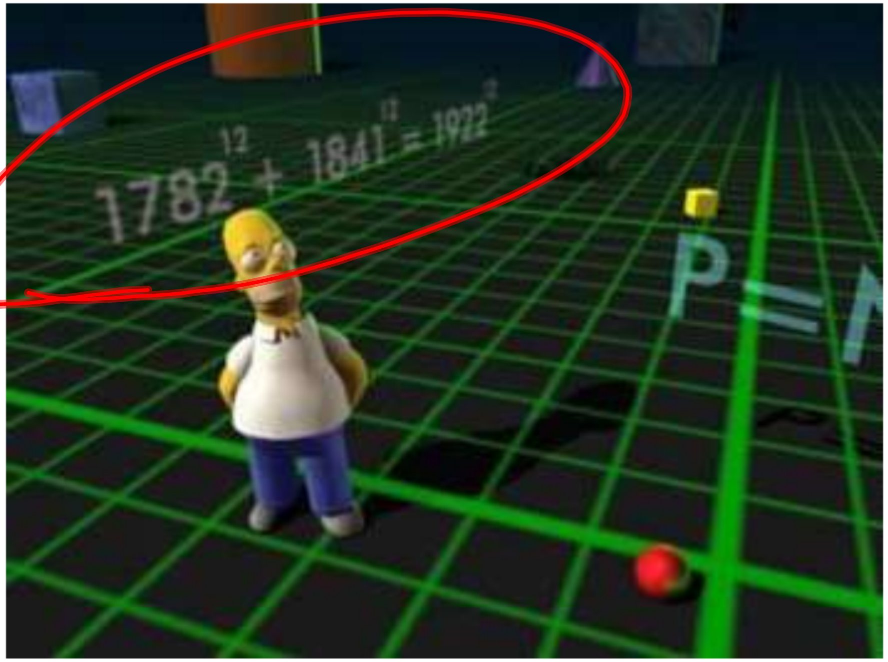


- - - - : ~~cost~~ dist 1

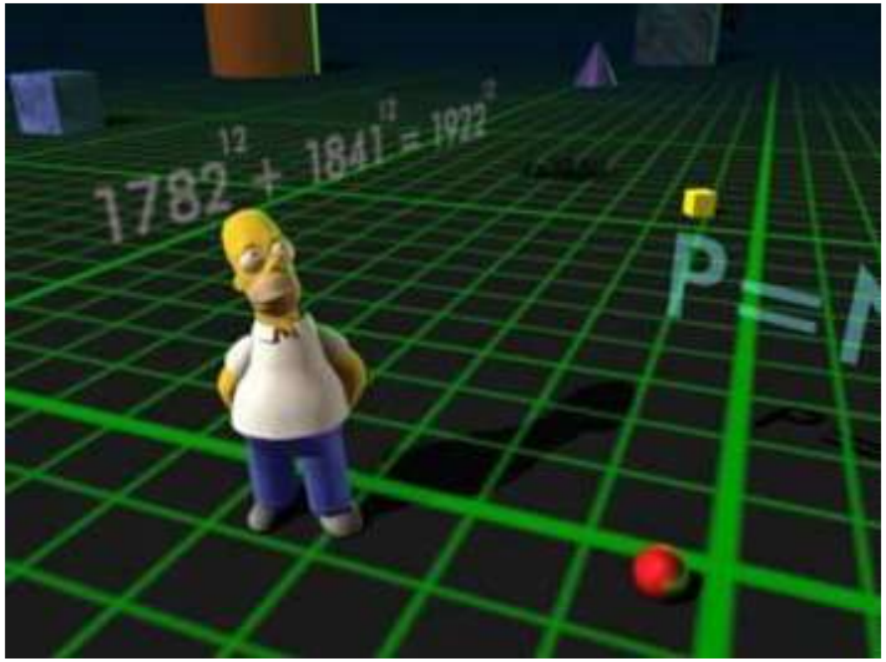
- - - - - : dist 2

$T = 1$

Type	Examples
Packing	Ind Set Str. Ind Set
Covering	Vert. Cover Set Cover
Partitioning	3-COLOR
Sequencing/Permutation	Ham. Path/Cycle TSP


$1782^{12} + 1841^{12} = 1922^{12}$
P=N

Integers
$A, B, C, N;$
$N \geq 3$
$A^N + B^N \neq C^N$



- $1782^{12} + 1841^{12} = 1922^{12}$

even + odd $\neq$ even


P=NP
$e^{\pi i} = -1$

$\mathcal{P} \subseteq \mathcal{NP}$

Vertex Cover

- We don't expect to find a correct and efficient algorithm
- How efficient can we get and still be correct?

- There are $\binom{n}{k}$ subsets
- Each could be a certificate of size k
- Each could be verified in $O(nk)$.
- This gives us a running time of $O(nk\binom{n}{k}) = O(kn^{k+1})$.
- $n = 1000, k = 10, \approx 10^{24}$ seconds to run.
  - Is this a reasonable running time?

We could do better...

```
VertexCover(G=(V,E), k)
```

if $|E| = 0$ **then**
  return true
if $|E| > k \cdot |V|$ **then**
  return false
$e = (u, v) \leftarrow$ arbitrary edge from E
return VertexCover($G - \{u\}, k - 1$) OR VertexCover($G - \{v\}, k - 1$)

$O(2^k kn)$

Vertex Cover in a tree

Greedy Algorithm:

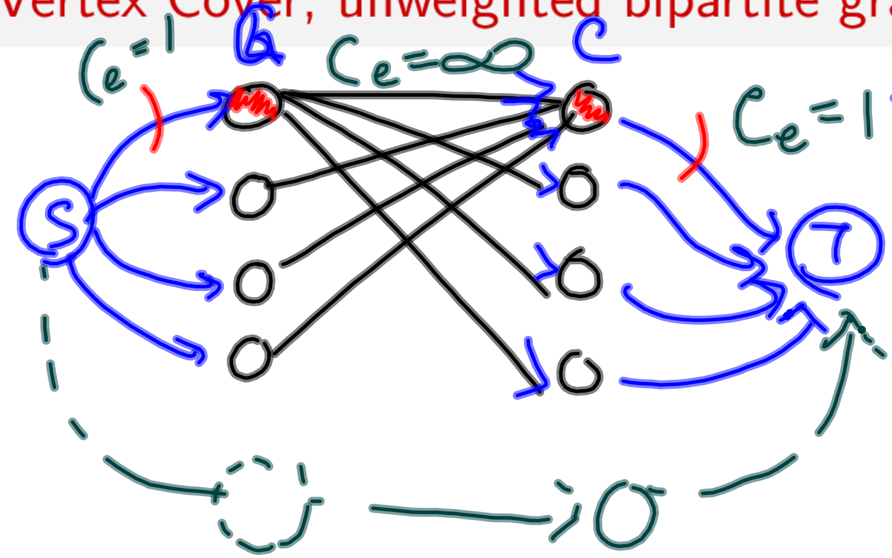
Ignore all leaf
take all parent of leaf
remove all edges incident to chosen
repeat / recurse on smaller subtrees

Vertex Cover in a weighted tree

- Dynamic Programming
- Simple outline?

Complement of hw1, Q3

Vertex Cover, unweighted bipartite graph



Set all e as directed $G \to C$

Simple Greedy Algorithm

Approximate-Vertex-Cover(G=(V,E), k)

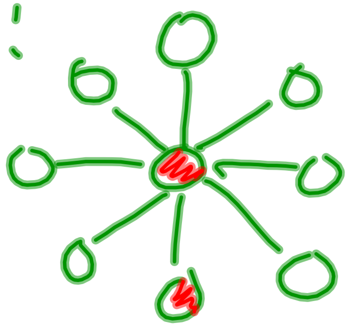
$C \leftarrow \emptyset$
$E' = G.E$
while $E' \neq \emptyset$ **do**
  $e = (u, v) \leftarrow$ arbitrary edge from E
  $C = C \cup \{u, v\}$
  Remove from E' every edge incident on u or v
return C

- Does this get a valid Vertex Cover?

Yes; after iter, E' is uncovered edges. None uncovered at end

Simple Greedy Algorithm

Q:

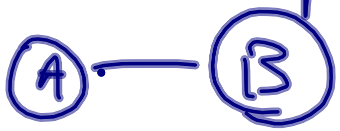


```
Approximate-Vertex-Cover(G=(V,E), k)
```

$C \leftarrow \emptyset$ // $A \leftarrow \emptyset$
$E' = G.E$
while $E' \neq \emptyset$ **do**
  $e = (u, v) \leftarrow$ arbitrary edge from E
  $C = C \cup \{u, v\}$ // $A = A \cup \{e\}$
  Remove from E' every edge incident on u or v
return C

- Does this get the minimum Vertex Cover?

No. Example:



How bad can it be?

- Let C be the cover returned
- Let C^* be optimal cover
- Let A be the set of edges chosen by algorithm

for each edge chosen (by us)
at least one endpt in OPT

$$\longrightarrow |A| \le |C^*|$$

$$|C| = 2|A| \le 2|C^*|$$

So our cover $\le 2 \cdot \mathrm{OPT}$

2-approximation