There is specific order to elements: (topo) order, string characters, intervals order, trees.

No natural order, asked for a partition of vertices reduce to mincut.

Asks for a subset of edges, reduce to max-flow or matching.

Reduction: Somatic re-formulate to a graph problem, use problems that we solved on graphs.

Try Inductive Proof to reduction. More creative, reduce to a problem that you have seen.

Practice HW, Course Material:

- Yes/no problem
- Max-flow augmenting path
- Four Alg design problems (20 minutes) per problem

Think before writing down a solution.
5 part (2) fastest ALG for these problems.

Computing longest common subseq of a pair of strings of length n,

\[ x_1 \ldots x_n \]
\[ y_1 \ldots y_n \]

\[ x = \hat{a}b\hat{a}c\hat{a} \]
\[ y = \hat{a}c\hat{a}b\hat{a} \]

\[ \text{common subseq:} \]
\[ i_1 \prec i_2 \prec i_3 \]
\[ j_1 \prec j_2 \prec j_3 \]

\[ \text{OPT}(i,j) = \text{longest common subseq of } x_i \ldots x_n \]
\[ \text{OPT}(i,j) = \begin{cases} 
\text{OPT}(i-1,j) & \text{if } x_i \text{ is matched to } y_j \\
\text{OPT}(i-1,j-1) & \text{OPT}(i-1,j-1) + 1 & \text{OPT}(i,j-1) & \text{OPT}(i,j-1) & \text{OPT}(i-1,j-1) \\
\text{OPT}(i,j-1) & \text{OPT}(i-1,j-1) & \text{OPT}(i,j-1) & \text{OPT}(i-1,j-1) & \text{OPT}(i,j-1) \\
\end{cases} \]

\[ \text{OPT}(i,j) = \max \{ \text{OPT}(i-1,j), \text{OPT}(i,j-1), \min [x_i = y_j] + \text{OPT}(i-1,j-1) \} \]

Problem 5 sample final 1.

\[ B = b_1 \ldots b_n \]

if some one occurs more than \( n/2 \) times.
Call Same \((b_i, b_j)\) returns true if \(b_i = b_j\)

# call \(O(n \log n)\): output yes if exists dominant

\[
\begin{aligned}
& b_i \cdots b_{n/2} \downarrow b_{n/2+1} \cdots b_n \\
\text{(n divided by 2)}
\end{aligned}
\]

if \(x\) dominant in \(b_i \cdots b_n \Rightarrow \) \(x\) is dominant in at least one of \([b_i \cdots b_{n/2}]\) or \([b_{n/2+1} \cdots b_n]\)

ALG: Recursively find dom in \((b_i \cdots b_{n/2}) = b_i\)

dom in \((b_{n/2+1} \cdots b_n) = b_j\)

If no dominant in \(b_i \cdots b_{n/2}\) and \(b_{n/2+1} \cdots b_n\)

then output no.

Count if \(b_i\) shows up \(> n/2 + 1\) in \((b_i \cdots b_n)\)

then output \(b_i\).

Call if \(b_{n/2+1}\) shows up \(> n/2 + 1\) in \((b_i \cdots b_n)\)

the output \(b_n\).

If none output no.

Run time: \(T(n) = 2T(n/2) + 2n \Rightarrow \) some call \(O(n \log n)\).

\[
T(n) = T(n-4) + Cn
\]

\[
= T(n-8) + C(n-4) + Cn
\]

\[
= T(n-12) + C(n-8) + C(n-4) + C
\]
\[
\begin{align*}
&= c \left( n^4 (n-4) + (n-8) \right) + \Theta(1) \\
&= 4c \left( \frac{n^4}{4} + \frac{1}{4} - \frac{1}{4} + \frac{2}{4} - \frac{1}{4} - 1 \right) \\
&= \frac{n^4 \cdot \left( \frac{3}{4} + 1 \right)}{2} \\
&= \Theta(n^2).
\end{align*}
\]

\[
T(n) = 2T(n-4),
\]

\[
T(n) = 2T(n-8).
\]

\[
T(n) = 2T(n-4) \\
\text{for odd } n.
\]

\[
= 2^k T(n-4; k)
\]

After \( k = \frac{n}{4} \)

\[
2^{\frac{n}{4}} T(n) = 2^{\frac{n}{4}} \neq \Theta(2^{\frac{n}{3}})
\]

\[
10 \ 2^2 = 100 \ n^2 \\
2^{\frac{n}{4}} 2^{\frac{n}{3}}
\]

\[
\left(2^{\frac{n}{4}}\right)^2 = 2^{\frac{2n}{4}} \\
\frac{2^{\frac{2n}{3}}}{2^{\frac{2n}{3}}}
\]

\[
f(n) = \Theta(g(n))
\]

\[
\text{if } \frac{f(n)}{g(n)} \to k \ \text{ as } \ n \to \infty
\]
\[
\frac{2^{n/3}}{2^{n/4}} \to 0 \quad \text{as } n \to \infty
\]

\[
2^n = \Theta \left( 2^{n+100} = 2^{100 \cdot 2^n} \right) \tag{Constant}
\]

HW 6 - P1.

\[I_{2n} = \text{Indep sets that have } 2n\]

\[I_{2n-1} = \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots 2n-\]

\[I_{2n} = I_{2n-1}
\]

\[\text{OPT}(2n,0) = \# \text{indep sets that } 2n, 2n-1 \text{ out}
\]

\[\text{OPT}(2n,1) = \# \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots 2n \text{ is in}
\]

\[\# \text{indep of this graph} \]

\[2n, 2n-1 \text{ out} \quad \text{OPT}(2n,0) \]

\[2n \text{ is in } \quad \text{OPT}(2n,1)
\]

\[2n-1 \text{ is in } \quad \text{OPT}(2n,1)
\]

\[\text{Both cannot be in}
\]

\[\implies \text{ output } \text{OPT}(2n,0) + 2 \cdot \text{OPT}(2n,1)
\]

Reduction to subproblem
\[
\begin{align*}
\text{OPT}(2n,0) & : 2n, 2n-1 \text{ out. So remove last color and you get an independent set of } \text{graph with } 2n-2 \text{ vertices.} \\
& = \text{OPT}(2n-2,0) + 2 \text{OPT}(2n-2,1). \\
\text{OPT}(2n,1) & : 2n \text{ is in } \Rightarrow 2n-1 \text{ out } \\
& \quad 2n-2 \text{ out } \\
& \quad 2n-3 \text{ in } = \text{OPT}(2n-2,1) \text{ (Claim)} \\
& \quad \Rightarrow \text{out: } 2n-3, 2n-2 \text{ out } \\
& \quad \text{OPT}(2n-2,0). \\
\text{OPT}(2n,1) & = \text{OPT}(2n-2,0) + \text{OPT}(2n-2,1) \\
\end{align*}
\]

Another way:
\[
\begin{align*}
\text{OPT}(2n,0) & : 2n, 2n-1 \text{ out } \\
& \Rightarrow 2n \text{ in } \\
& \quad 2 \rightarrow 2n-1 \text{ in } \\
\text{output: } \text{OPT}(2n,0) + \text{OPT}(2n,1) + \text{OPT}(2n,2) \\
\text{OPT}(2n,0) & = \text{OPT}(2n-2,0) + \text{OPT}(2n-2,1) + \text{OPT}(2n-2,2) \\
\text{OPT}(2n,1) & : 2n \lor 2n-1 \times 2n-2 \times \\
& \quad 2n-3 \Rightarrow \text{in OPT}(2n-2,2) \\
& \quad \text{out: } \text{OPT}(2n-2,0) \\
\text{OPT}(2n,2) & : 2n \times 2n-1 \lor 2n-3 \times \\
& \quad 2n-2 \Rightarrow \text{in OPT}(2n-2,1) \\
& \quad \text{out: } \text{OPT}(2n-2,0) \\
\end{align*}
\]

Problem 7:
Number Partition problem

Given non-negative \(y_1, \ldots, y_n\) Is it possible to partition
into two groups so that sum in each group is the same.
a) Show \( \text{Num} \) \( \frac{\text{par}}{\text{in NP}} \).

\[ S \subseteq S \]

\[ \text{sum can be calculated in } n \cdot \max \{y_i\} \]

Input \( y_1, \ldots, y_n \)

- My bits: \( y_i \rightarrow b_1 \) \( b_{y_i} \) bits
- \( n \cdot \max \{y_i\} \)

- Running time is \( \text{poly} \) in input size if \( \text{poly}(n, \max \{y_i\}) \)
- \( \text{exp} \) if \( \text{poly}(n, y_i) \)

\[ a + b \]

\[ a \leftarrow a \cdot k \]

\[ b \leftarrow b \cdot c \]

b) Show that: Subset Sum \( \leq_p \) Number partition:

Subset Sum: Given \( x_1, \ldots, x_n \) \( \text{ints} \) can be \( \text{pos} \)

or \( \text{neg} \). We want to see if \( \exists \) subset that sum up to \( 1 \).

\[ \text{Algorithm (Alg)} \]

\[ y_1, \ldots, y_n \]

\[ \text{Number Partition} \]

\[ 0 \]

\[ \text{Yes} \]

\[ \text{No} \]

\[ S = x_1, \ldots, x_n \]

\[ T \text{ adds up to } S-1 \]

\[ y = S - 1 + 10S \]

\[ Z = 1 + 10S \]
\[ \text{Prob. } A \leq_p B \]

\[ A \in P \Rightarrow A \leq_p B \text{ for all } B \in \text{EXP} \]

True: \( A \in P \Rightarrow \text{poly time ALG for } A \).

don't use help.