asked n log n running time try Greedy ALG
approx Greedy ALG.
There is specific order to elements (topology order, string character, intervals order, Trees.)
D.P

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

Asked for a subset of edges or routing water or electricity
reduce to Max-flow or matching.

Reduction: Sometimes re-formulate to a graph problems
use problems that we solved on graphs

---

try Induction Proof: DP, Greedy, Divide and Conquer

try 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.

$x_1 \text{ --- } x_n$

$y_1 \text{ --- } y_n$

Common subseq:

$i_1 < i_2 - < i_k$

$j_1 < j_2 - < j_k$ s.t.

$x_{i_1} = y_{j_1}$

$x_{i_2} = y_{j_2}$

$\vdots$

$x_{i_k} = y_{j_k}$

x = a b a c a

y = a c a b a

$OPT(i,j)$ = longest common subseq of $x_1 \dots x_i$, $y_1 \dots y_j$

$$OPT(i,j) = \begin{cases} x_i = y_j & OPT(i-1,j-1)+1 \\ x_i \neq x_j & \begin{cases} OPT(i,j-1) & (x_i \text{ is matched to } y_j, \\ OPT(i-1,j) & y_j \text{ is matched to } x_i, \\ OPT(i-1,j-1) & \text{none is matched in lng common subseq} \end{cases} \end{cases}$$

$$OPT(i,j) = \max\{OPT(i,j-1), OPT(i-1,j), \mathbb{I}[x_i = y_j] + OPT(i-1,j-1)\}$$

Problem 5 sample final 1.

$B = b_1 \text{ --- } b_n$

if ~~some~~ one occurs more than $n/2$ times.

Call Same (b_i, b_j) returns true if $b_i = b_j$

call $O(n \lg n)$. output yes if exists dominant

no o.w.

$\boxed{b_1 \ldots b_{n/2} \mid b_{n/2+1} \ \ b_n}$ (n divisible by 2)

if x dominant in $b_1 - b_n \Rightarrow x$ is dominant in at least one of $[b_1 - b_{n/2}]$ or $[b_{n/2+1} - b_n]$

ALG: Recursively find domin of $(b_1 - b_{n/2}) = b_i$

dom of $(b_{n/2+1} \rightarrow b_n) = b_j$

If no dominant in $b_1 - b_{n/2}$ and $b_{n/2+1} - b_n)$ then output no.

Count if b_i shows up $> n/2 + 1$ in $(b_1 - b_n)$ then output b_i

Cou if b_j shows up $> n/2 + 1$ in $(b_1 - b_n)$ the output b_n.

If none output no.

Run time: $T(n) = 2T(n/2) + \underset{\text{Sam cal}}{2n} \Rightarrow$ # Same call $\leq O(n \lg n)$.

$T(n) = T(n-4) + cn$

$= T(n-8) + c(n-4) + cn$

$= T(n-12) + c(n-8) + c(n-4) + c$

$= c\ (n + (n-4) + (n-8) + \cdots 0)$

$= 4 \cdot c \left(\underbrace{\tfrac{n}{4} + (\tfrac{n}{4} - 1) + (\tfrac{n}{4} - 2) + \cdots 1}_{\frac{\frac{n}{4} \cdot (\frac{n}{4} + 1)}{2}} \right)$

$= \Theta(n^2).$

$T(n) = 2T(n-4).$ $\quad T(n-4) = 2T(n-8)$

$= 4\ T(n-8)$ $\quad \boxed{T(n) = 2T(n-4)}$

$= 8\ T(n-12)$ $\quad$ for any n.

$= 2^k\ T(n - 4 \cdot k)$

after $k = n/4$

$2^{n/4}\ T(0) = 2^{n/4} \neq \Theta(2^{n/3})$

$10\ n^2 = 100\ n^2$

$2^{n/4} \qquad 2^{n/3}$

$(2^{n/4})^2 = 2^{2n/4}$ $\qquad \frac{2^{2n/3}}{2^{n/3}}$

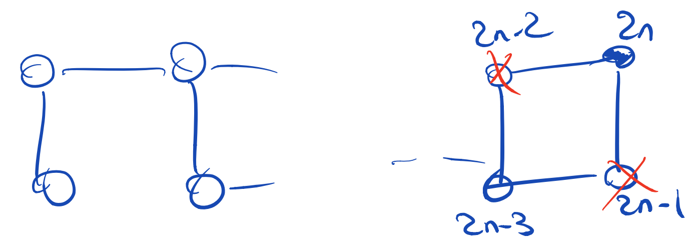
$f(n) = \Theta(g(n))$

if $\frac{f(n)}{g(n)} \to k$ as $n \to \infty$

$$\frac{2^{n/4}}{2^{n/3}} \xrightarrow[n\to\infty]{} 0$$

$$2^n = \Theta\left(2^{n+100} = \underbrace{2^{100}}_{\text{constant}} \cdot 2^n\right)$$

HW6 - P1.



indep sets

Claim I_{2n} = Indep sets that have $2n$

I_{2n-1} = 〃 〃 〃 〃 $2n-1$

$|I_{2n}| = |I_{2n-1}|$.

OPT(2n,0) = # indep sets that $2n$, $2n-1$ out

OPT(2n,1) = # 〃 〃 〃 $2n$ is in.

indep of this graph

- → $2n$, $2n-1$ out OPT(2n,0)
- → $2n$ is in OPT(2n,1)
- → $2n-1$ is in OPT(2n,1)
- → ~~Both can be in~~

⇒ output OPT(2n,0) + 2 OPT(2n,1).

Reduction to subproblems

$OPT(2n, 0)$: $2n$, $2n-1$ out. So remove last column and and you get an dadp set of graph with $2n-2$ vertices.

$= OPT(2n-2, 0) + 2\, OPT(2n-2, 1)$.

$OPT(2n, 1)$: $2n$ is in $\Rightarrow$ $2n-1$ out, $2n-2$ out

$2n-3 \rightarrow$ in $= OPT(2n-2, 1)$ (claim)

$2n-3 \rightarrow$ out $\Rightarrow$ $2n-3, 2n-2$ out, $OPT(2n-2, 0)$.

$$OPT(2n, 1) = OPT(2n-2, 0) + OPT(2n-2, 1)$$

Another way

$OPT(2n,$ $0 \rightarrow 2n, 2n-1$ out; $1 \rightarrow 2n$ in; $2 \rightarrow 2n-1$ in $)$

output $OPT(2n, 0) + OPT(2n, 1) + OPT(2n, 2)$

$$OPT(2n, 0) = OPT(2n-2, 0) + OPT(2n-2, 1) + OPT(2n-2, 2)$$

$OPT(2n, 1)$ $2n$ ✓ $2n-1$ ✗ $2n-2$ ✗

$2n-3 \rightarrow$ in $OPT(2n-2, 2)$ +

$2n-3 \rightarrow$ out $OPT(2n-2, 0)$

$OPT(2n, 2)$: $2n$ ✗ $2n-1$ ✓ $2n-3$ ✗

$2n-2 \rightarrow$ in $OPT(2n-2, 1)$ +

$2n-2 \rightarrow$ out $OPT(2n-2, 0)$

Problem 7:

Number Partition problem.

$\sum_i y_i = 2k$

Q: find a subset of sum $= k$.

Given non-neg int $y_1 \ldots y_n$ Is it possible to partition into two groups so that sum in each group is the same.

a) Show Num Par in NP.

$S, \bar{S}$

sum can be calculated in $n \cdot \max \lg y_i$

Input $y_1 \text{---} y_n$

How many bits: $y_i \longrightarrow \lg y_i$ bits

$n \cdot \max \lg y_i$

$\Rightarrow$ Running time is poly in input size if $\text{poly}(n, \max \lg y_i)$

exp if $\text{poly}(n, y_i)$

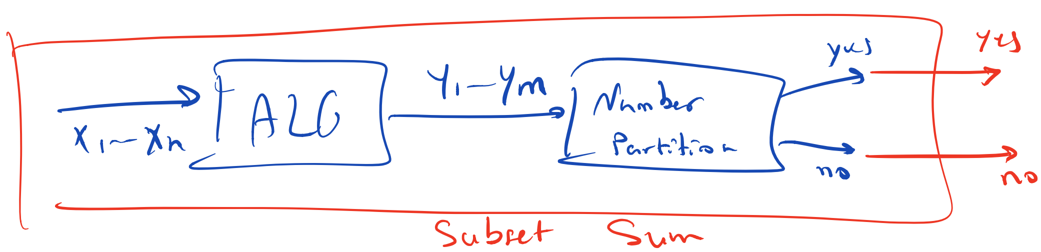
$a + b$

$a_1 \; a \text{---} \; a_k$

$b_1 \text{---} \; b_k$

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

Subset Sum: Given $x_1 \text{---} x_n$ integers can be pos or neg. We want to see if $\exists$ subset that sum up to 1.



$S = \underbrace{x_1 \text{---}}_{\text{subse}} x_n$

T adds up to 1 $\qquad \bar{T}$ adds up to $S-1$

$y = S - 1 + 10S \qquad z = 1 + 10S$

Pr— $A \le_p B$

help

$A \in P \Rightarrow \underbrace{A \le_p B}$ for all $B \in \cancel{NP}$ EXP

True. $A \in P \Rightarrow$ poly time ALG for A.

don't use help.