DP: Longest Palindromic Subsequence (LPS)

Prob

Given a string s, find the length of the ~~LPS~~ longest subsequence that is a palindrome.

I/P :- A string s of length n where $1 \leq n \leq 1000$
string contains lower case letters only.

O/P :- Integer representing the length of the LPS.

Ex :-
I/P :- S = "bbbab"
O/P :- 4 "bbbb"

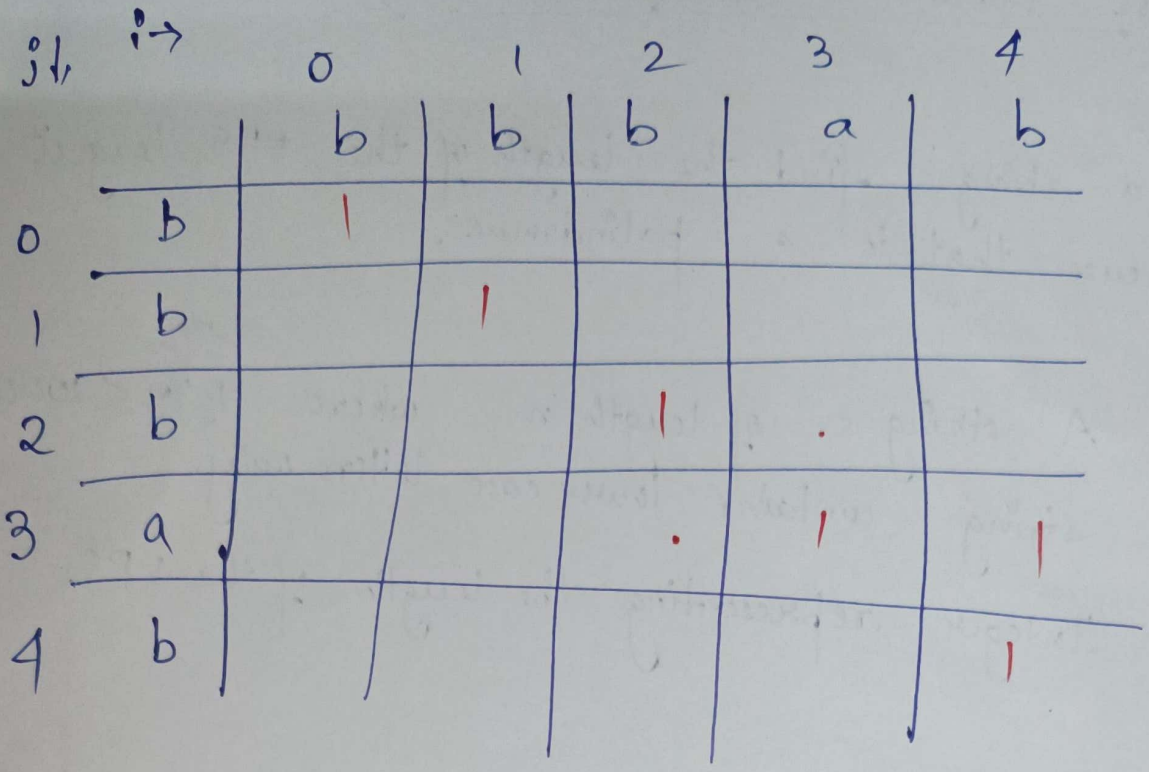
Ex :-
I/P :- S = "cbbd"
O/P :- 2 "bb"

Top-Down Memoization

memoization table ⇒ dp[i][j]

```
lps(s):
    n ← len(s)
    Create memoization table dp[n][n] and initialize to -1

    if i>j :                    } Base case :- Invalid range.
        return 0

    if i==j :                   } Base case :- only one character
        return 1

    if dp[i][j] != -1 :         } already computed
        return dp[i][j]

    if s[i] == s[j] :
        dp[i][j] ← 2 + lps(i+1, j-1)
    else
        dp[i][j] ← ~~max(dp[i+1][j], dp[i][j-1])~~ max(lps(i+1, j), lps(i, j-1))
    return dp[i][j]
```

Each subproblem - O(1)
total $O(n^2)$ subproblems



$lps(4,4) = 1$ ···· $lps(0,0) = 1$

$lps(3,4) = 1$ ["ab", $\max(lps(4,4), lps(3,3))$]

$lps(2,3) = 2$ ["ba", $2 + lps(3,2)$]
$= 2$

Greedy Approach :- Kruskal algo for MST

$O(E \log E)$

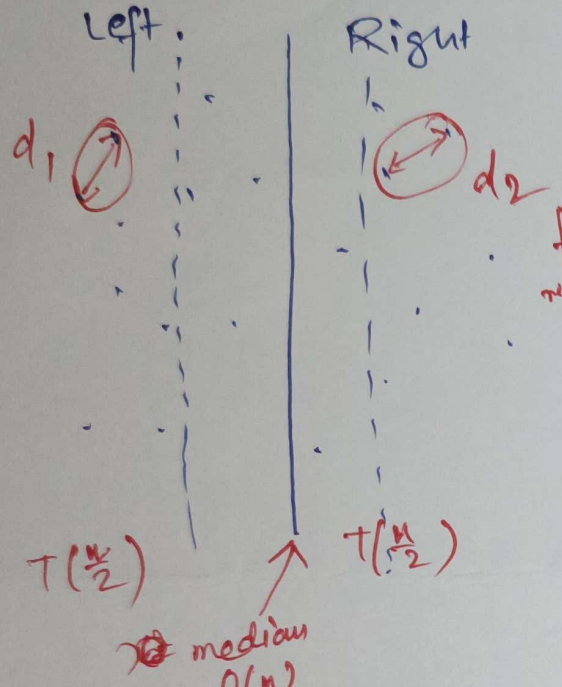
Outline:-

Closest Pair Prob (2D).

I/P: set of n points in 2D $\{(x_1,y_1), (x_2,y_2), \ldots (x_n,y_n)\}$

O/P: find the closet pair $\begin{bmatrix}(x_i,y_i)\\(x_j,y_j)\end{bmatrix}$

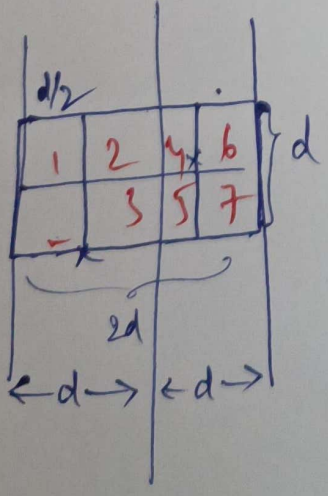
Brute force :- $O(n^2)$

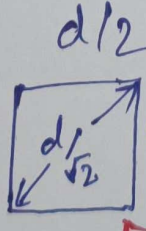


find recursively
1. Find closest pair in LEFT (d_1)
2. Find closest pair in RIGHT (d_2)
3. Check if there exists a pair, st one point lies on LEFT & one pt lies on RIGHT & the dist b/w them is $< d = \min\{d_1, d_2\}$

for the pts in the band region arrange in ascending order of y-coordinates → $P_1\ P_2\ P_3\ P_4 \ldots P_{50}\ P_{51} \ldots$ (7 other pts)

Correctness.





$\frac{d}{\sqrt{2}} < d$

here can't add two points.

if so then the dist b/w them will be $< \frac{d}{\sqrt{2}}$ which is trivially $< d$.

$$T(n) = 2T(\tfrac{n}{2}) + O(n) + O(n\log n)$$

$O(n)$: median; $O(n\log n)$: sorting

$$= 2T(\tfrac{n}{2}) + O(n\log n).$$

P → problems that are solvable in polynomial time
↓
n^k

NP → problems (rather decision problems) that are solvable in non-deterministic polynomial time.
↳ YES OR NO
↳ guess one out of many options (polynomial) in O(1) time.

CLIQUE PROBLEM

To prove a problem L is NP-complete, we have to do the following steps :-

(i) $L \in NP$.

(ii) $L' \leq_p L$, where L' is an established NP-complete problem.

Both L & L' are decision problems.

For proving CDP as NP-complete, we take 3CNF-SAT and take help from it, as the later is a known NP-complete problem.

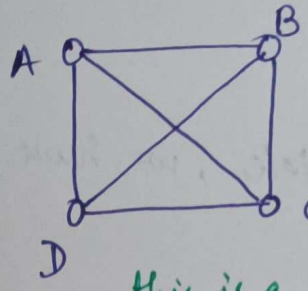
3CNF-SAT : $\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee \neg x_4) \wedge (x_4 \vee \neg x_5 \vee x_4)$

clause — literals

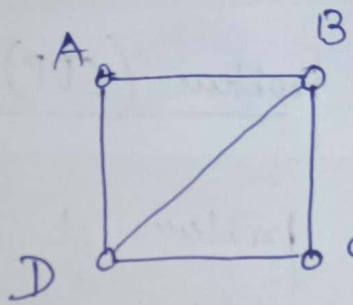
arrange the literals, s.t. ϕ turns out to be "true."

A clique is an undirected graph, $G = (V, E)$ is a subset $V' \subseteq V$ of vertices., where each pair is connected by an edge $e \in \{E\}$. Size of clique is the number of vertices in it.

($\{E\}$ ← edge set.)



this is a clique.



not a clique

A & C not connected.

Find the clique of maximum size in a graph G. } → optimization problem.

Does a clique of size k exist in a graph G. } → decision problem.

Brute force algo :-

list all K subsets of V, then check each subset to see if it's a clique. For checking if a subset is a clique, we need to check if an edge exists b/w each pair of vertices in that subset.

↑ not polynomial time algo.

Step 1

Clique $\in$ NP

If given a certificate, we need to be able to verify it in polynomial time. Given a certificate we can check if it is correct by verifying it for each pair (u, v) in the certificate, there is an edge.

Step 2

Prove 3CNF-SAT $\leq_p$ Clique.

The reduction algo begins with an instance of 3CNF-SAT.

Let $\phi = (C_1) \wedge C_2 \wedge \dots \wedge C_K$ be a boolean formula in 3CNF with K clauses. (clause.)

$C_r \rightarrow L_1^r, L_2^r, L_3^r$ are the three unique literals.

Construct a graph s.t. ϕ is satisfiable (iff) G has a clique of size K.

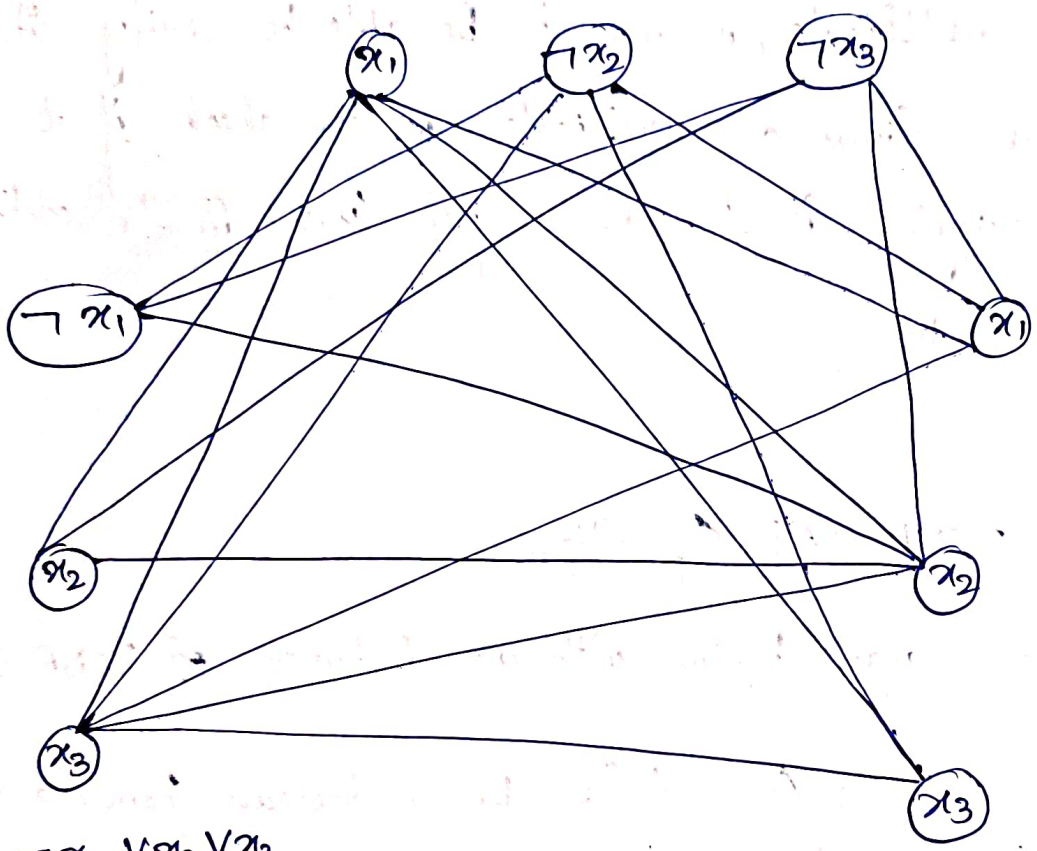
(iff) ← show the reduction both ways.

For each clause $C_r = (L_1^r \vee L_2^r \vee L_3^r)$ in ϕ we place a triple vertices V_1^r, V_2^r, V_3^r as follows.

For example,

$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$

$C_1 = x_1 \vee \neg x_2 \vee \neg x_3$



$C_3 = x_1 \vee x_2 \vee x_3$

$C_2 = \neg x_1 \vee x_2 \vee x_3$

① Note there is no edge b/w x_i and $\neg x_i$ if they are in separate groups.

② No edges b/w vertices in the same group.

We need to show this transformation from ϕ to G is a reduction.

① First let ϕ has a satisfying assignment.

↳ each clause C_r contains atleast one literal $L_t^r = 1$

↳ picking one "T" literal from each clause yields a set V' of K vertices. We claim that V' is a clique.

↳ For any two vertices $v_i^r, v_j^s \in V'$ $r \neq s$ because we only pick one vertex from each group.

↳ Literals l_i^r and l_j^s corresponding v_i^r and v_j^s both maps to 1 and thus l_i^r and l_j^s can't be complements.

↳ Hence by constructing a graph G an edge exists b/w v_i^r and v_j^s in V'. Thus the vertices in V' form a clique of size $|V'| = K$

② Suppose G has a clique V' of size k.

↳ no edges in G connect vertices in the same triple.

↳ ∴ V' contains exactly one vertex per triple.

↳ assign 1 to each literal l_i^r s.t. $v_i^r \in V'$. It is possible because G contains no edges b/w a literal and its complement, so we can assign $l_i^r = 1$.

↳ each clause C_r is satisfied and hence ϕ is satisfied.

Hence, we complete the reduction as we have shown the transformation holds both ways.

Summarization

① We took an instance of 3CNF-SAT and reduced it to Clique & not the other way around.

② We have shown that a specific instance of Clique is hard.