

Lecture 14 +15

Core Algorithms

N-V algs for TSP
 - $O(n^3 \phi)$
 - in expectation.

$$1, \dots, n \quad (x_1, \dots, x_n) \in \{0, 1\}^n$$

$$P_1, \dots, P_n$$

$$w_1, \dots, w_n$$

$$\text{maximize } P_1 x_1 + \dots + P_n x_n$$

$$\text{subject to } w_1 x_1 + \dots + w_n x_n \leq W$$

$$x_i \in \{0, 1\} \forall i$$

Fractional Knapsack
 → where fractions of items can be stored

Relaxation:

$$x_i \in [0, 1] \forall i$$

i.e. x_i allowed to be a real value.

Greedy for frac. Knapsack

Sort items in the descending order of

$$P_i / w_i \text{ ratio}$$

$$1, 2, \dots, n$$

$$P_i / w_i \geq \frac{P_{i+1}}{w_{i+1}}$$

Add items as long as weight does not exceed capacity.

$$x_1, x_2, \dots, x_{i-1}$$

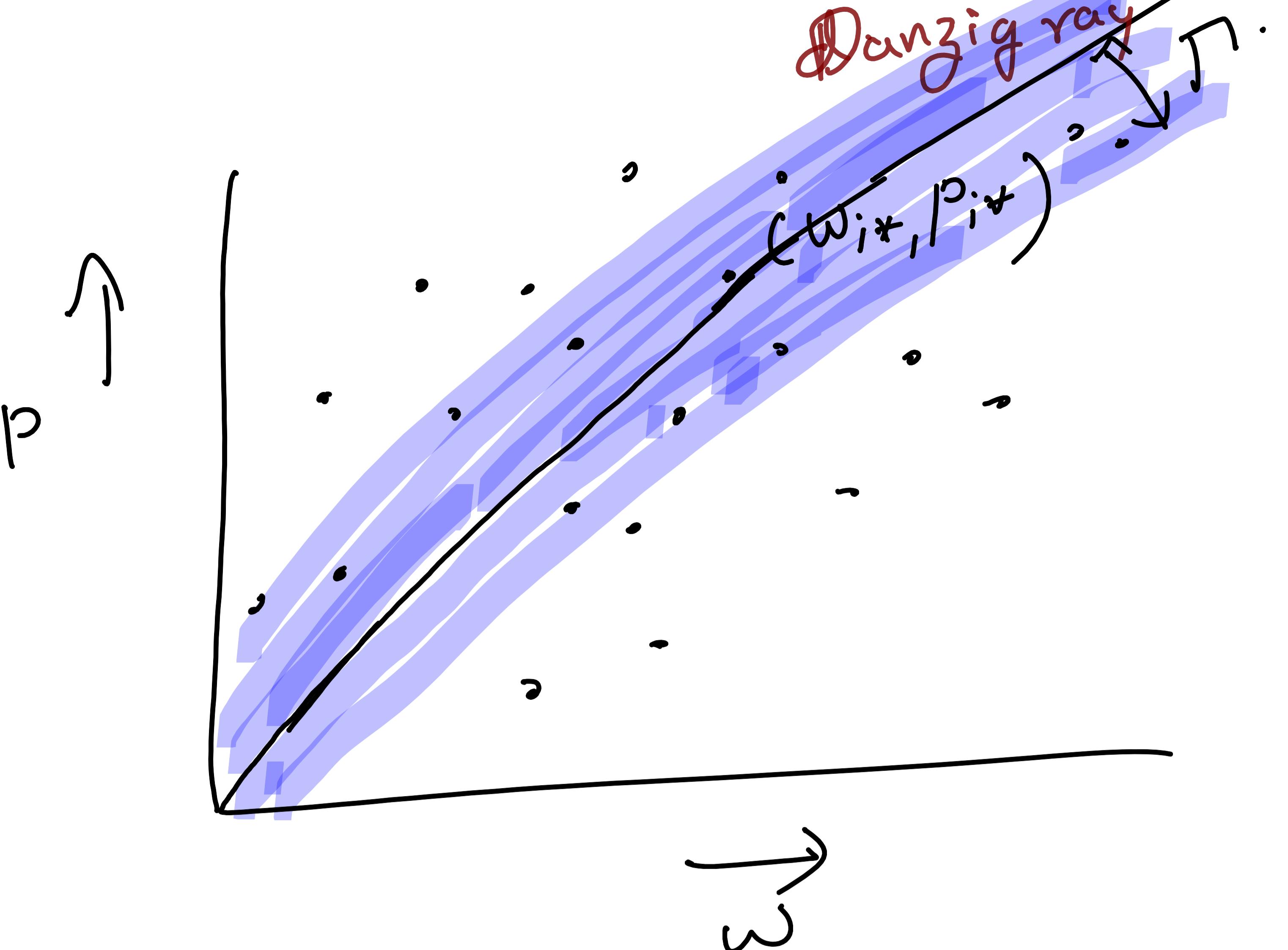
obs:- The solution is optimal & there is at one max one time it's set

$$x_i \notin \{0, 1\}$$

$$(w_i^*, P_i^*)$$

\hat{x} be an optimal solution.

stmt: if $w_1 \dots w_n$ &
 $p_1 \dots p_n$ are ϕ -perturbed
 $\exists n$ s.t # of items
 within a distance δ of n from
 the Danzig ray is $O(n^{\frac{1}{3}})$
 \therefore Expected running time = $\text{Sort}(n) + (n^{\frac{1}{3}})^3 \cdot \phi$.
 $\approx O(n \log n)$.

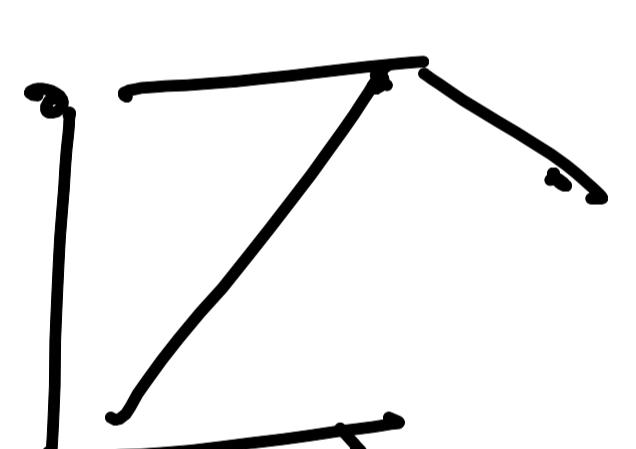


Details
 See Röglin's
 notes.

A quick review of Complexity classes

NP: non-deterministic polynomial time.

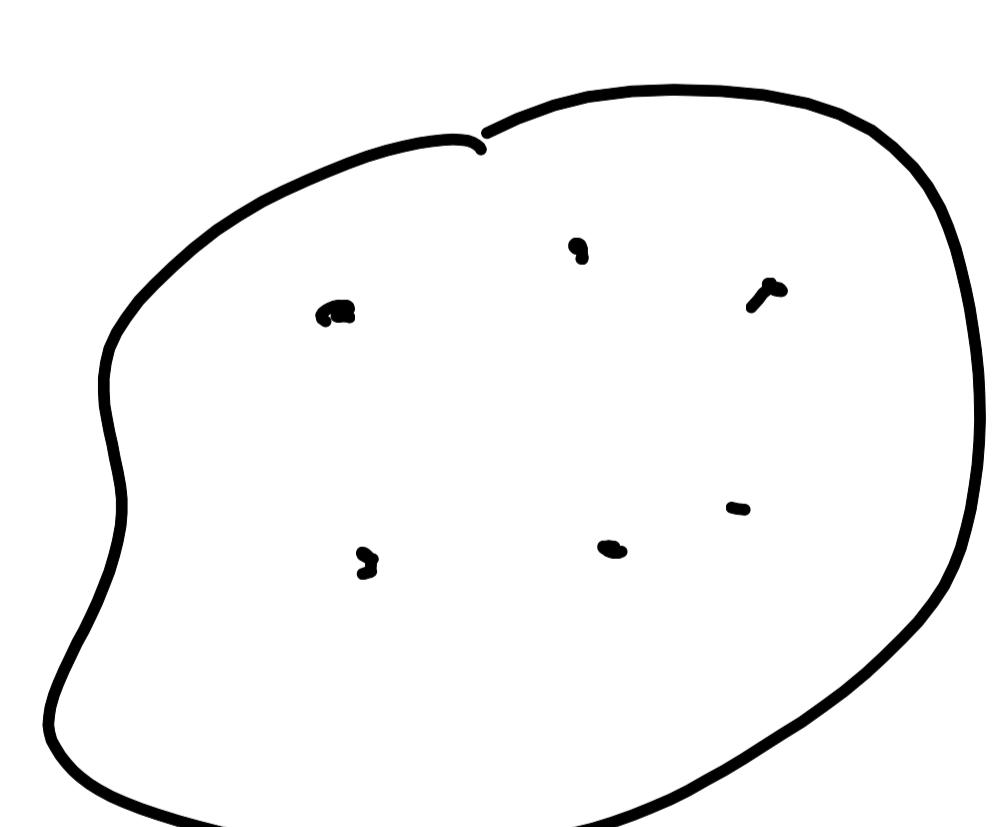
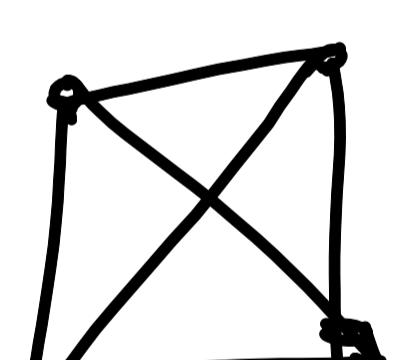
Set of all problems that have efficiently
 verifiable soln/witnesses.



e.g. CLIQUE problem

i/p: $G = (V, E)$ & $K > 0$

s/p: YES iff G has a clique on K vertices

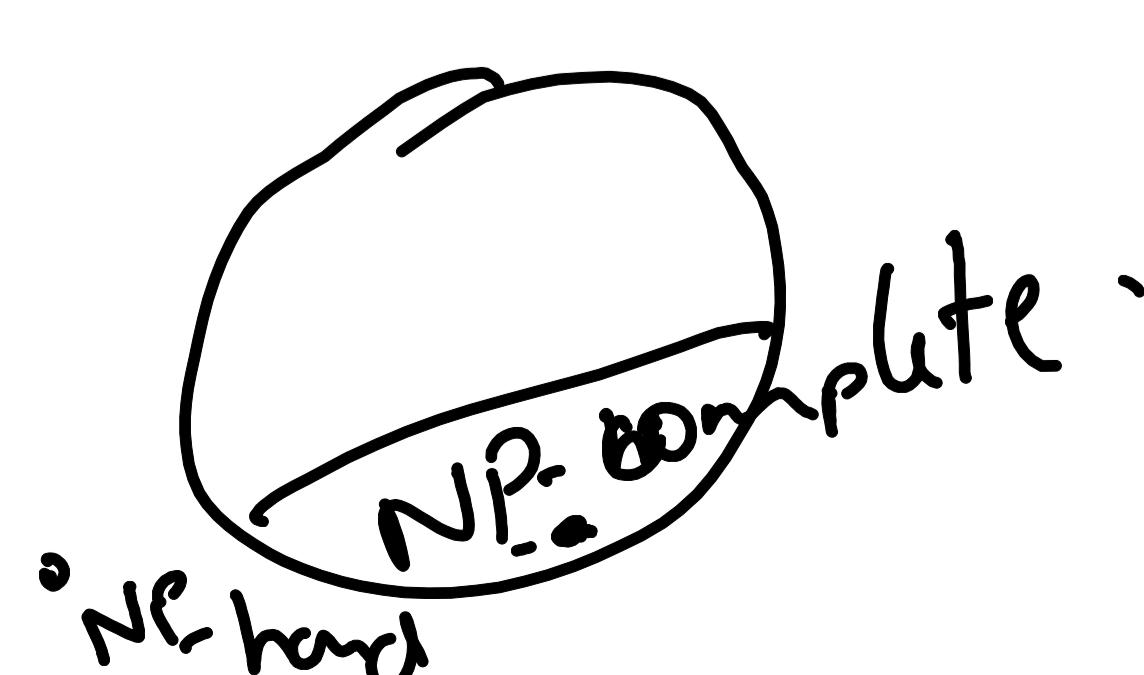


witness for YES instance

A subset $S \subseteq V$, $|S| = k$
 s.t. $\forall i \neq j \in S, (i, j) \in E$

Vertex cover
 Dominating
 set
 Hamiltonian
 cycle

$\therefore \text{CLIQUE} \in \text{NP}$



NP-hard

NP-complete

Given G & k
 is there a
 path of length
 k

NP-hard

A problem B is NP hard if there is a poly time algo for B then for every problem $A \in NP$, we have a polynomial time algorithm.

Note - CLIQUE, DomSet, VC, Hamcycle, k-path, k-cycle
 Knapsack, TSP, clustering.
 All are NP hard problem. (in fact $NP \neq P$ hypothesis)

For us:- No NP-complete problem is likely to have a polynomial time algorithm. ie $NP \neq P$.

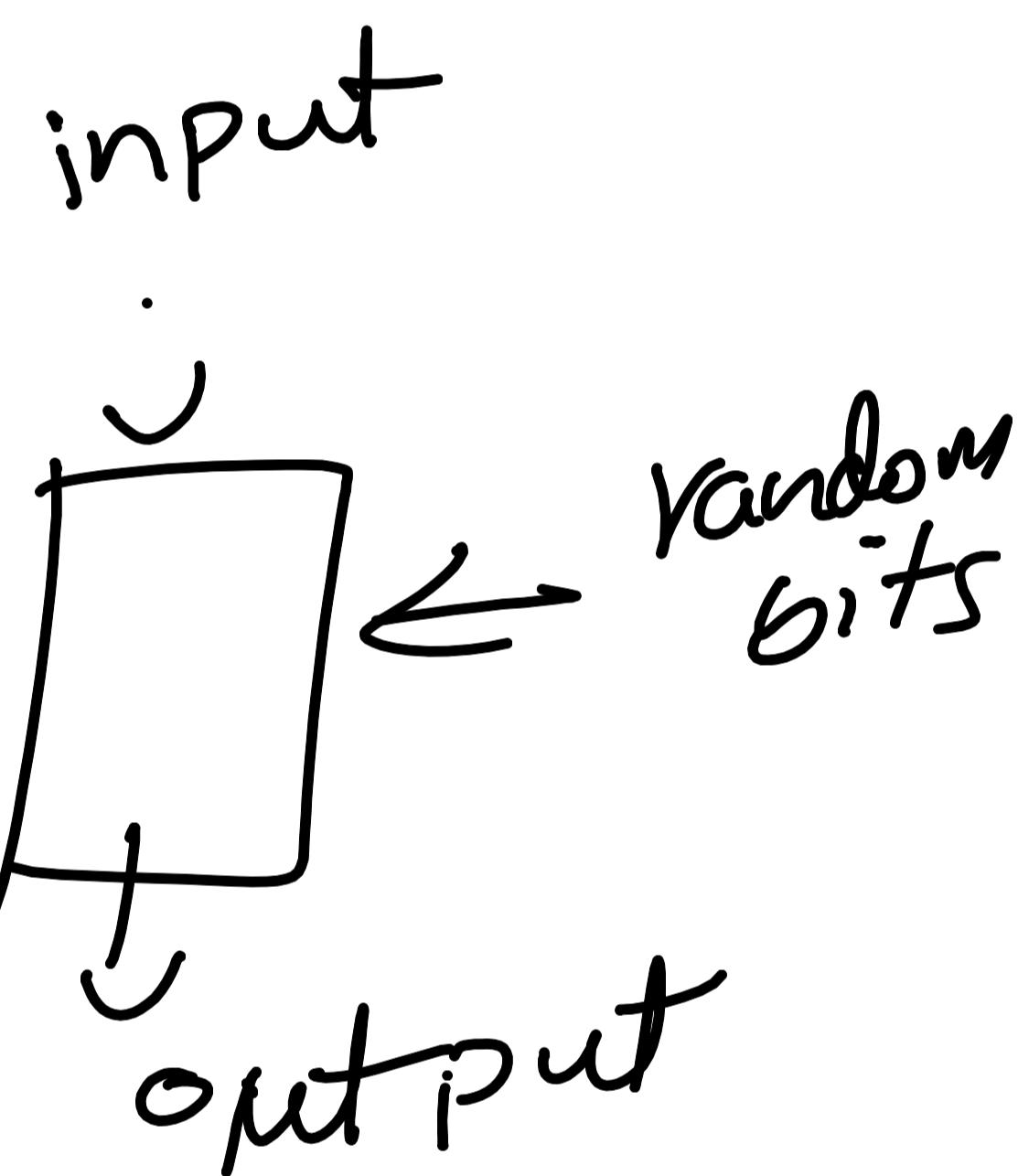
Suppose an algorithm A uses random coin tosses-

Monte Carlo & Las Vegas-

↓
runtime expectation
o/p: guaranteed

↓
run time guaranteed
o/p: expectation (ie possibility of errors).

TSP-implementations
Concorde -



Definition A problem B is said to be in ZPP
 (Zero error prob. poly time)

if it has a randomized algo which has expected poly time & outputs correct value.

Weakened P vs NP

Lecture - 15

Weakened P vs NP

NP \neq ZPP

Knapsack: has smoothed
poly time algo

TSP ? - 2OPT - does not
give exact soln.
Steiner tree ?
K-means ?

Strongly NP-hard

A problem B (binary optimization) is said to B

strongly NP hard if the restriction

where $|c_i| \in [0, n^c]$ for some $c > 0$

remains NP hard.

$$\max_{\min} c_1 x_1 + \dots + c_n x_n$$

$$\text{Subj: } \dots$$

$$x_1, \dots, x_n \in \{0, 1\}$$

Example:- TSP when the edge weights are 1 or 2

Examp :-

NP hard -

remains NP-hard.

\therefore TSP is strongly NP-hard.

Ex:-

① Show that TSP can be formulated as a binary

optimization problem.

② Suppose there is a poly time algorithm that
Solves TSP where weights are 1 or 2. Show that

This algo. can be used to solve the HamCycle problem.

Thm: Let π be a binary opt. problem which is
Strongly NP-hard. N - input size n - # of variables.
Then there does not exist an algo for π whose
expected run. time is bounded by $\text{poly}(N, \phi)$ for
 ϕ - perturbed instances with coefficients from $[-1, 1]$
unless $\text{NP} \subseteq \text{ZPP}$.

Corollary: TSP cannot have smoothed poly time
algo unless $\text{NP} \subseteq \text{ZPP}$.

Pf: A be an algo for ϕ -perturbed instances of π

$$\text{s.t. } \max_z \mathbb{E} [\text{time}(A, y)] = \text{poly}(N, \phi)$$

Now, π is strongly NP hard, let Q be a poly s.t
the special case of π where $|c_i| \leq Q^{(n)}$ remain
NP-hard.

[e.g. for TSP
 $Q^{(n)} = 2$ suffices]

I be an input instance of π of length N and

