CSE 201 3-10-20

NP completeness

P: Polynomial-time solvable Problems. "tractable"

NP: Poly-time verifiable Problems

NP complete: NP but harder (or eq.) than every NP Problem

Some Problems are not solvable.

handout: Halting Problem.

Problems:

- Shortest-Path (SP)

Instance: graph G, $x, y \in V(G)$

solution: length of a shortest x-y path in G.

optimization

Poly-time solvable (BFS, Dijkstra, ...)

- Hamiltonian-Cycle (HC)

Instance: graph G

solution: determine (Y/N) whether G contains a Hamiltonian cycle (a cycle C that includes all vertices in G.)

decision

No Poly-time algorithm is known for HC.

HC is in NP however.

To verify: given C and

$$C = (x_0, x_1, \ldots, x_n)$$

a seq. of vertices, check

(1) $\{x_{i-1}, x_i\} \in E(G)$ for $i = 1, \ldots n$.

and

(2) check $x_0 = x_n$ and

$$V(C) = \{x_1, \ldots, x_n\}.$$

languages:

$L \subseteq \{0,1\}^* = \{$finite length bit-strings$\}$

op. Problems $\leftrightarrow$ dec. Problems $\leftrightarrow$ languages

conversion encoding

- PATH

Instance: graph G, $x, y \in V(G)$, and a bound $K \in \mathbb{Z}^+$.

Solution: determine (Y/N) whether G contains an x-y path of length at most K. [decision]

observe: any algorithm for SP can be converted to one that solves ~~PATH~~. How?

given (G, x, y, k) instance of ~~PATH~~, run algorithm for SP, get an output ℓ. (length of a shortest x-y path).

if $\ell \leq k$

return YES

else

return NO.

Note, if alg. for SP runs in Poly-time, so does the derived algorithm for PATH.

so PATH is no harder than SP!

$$PATH \leq SP$$

We can generalize to any Optimization Problem (OP).

assume OP is a minimization Problem. Define a Decision Problem (DP) as follows

Take any instance $\hat{\alpha}$ of OP and add a bound $b \in \mathbb{R}$.

an instance of DP is (α, b)

DP

instance of OP

Instance: (α, b)

solution: does there exist (Y/N) a feasible solution to α whose value is at most b, i.e

$$v(\alpha) \leq b$$

↑

objective fcn for OP

again: any poly-time alg. for $\mathcal{O}\mathcal{P}$ can be converted to one for $\mathcal{N}\mathcal{P}$, showing

$$\mathcal{N}\mathcal{P} \leq \mathcal{O}\mathcal{P}$$

Can reverse this process:

Given instance α of $\mathcal{O}\mathcal{P}$, run algorithm for $\mathcal{N}\mathcal{P}$ successively on $(\alpha,1), (\alpha,2), (\alpha,3), \ldots\ldots (\alpha,\ell)$

no no no .... no, yes

stop when reach 1st yes, then return ℓ.

Exercise

do this in concrete case of

PATH $\leq_P$ SP.

Conclusion: in general

$$O_P \leq D_P$$

we lose nothing by concentrating on decision Problems.

Suppose we have 2 decision Problems A, B. Suppose we have an algorithm that converts any instance α of A to an instance β of B. Suppose also that YES instances are mapped to YES instances & NO to NO.

We call such a mapping a <u>reduction</u> from A to B.

Can now convert any alg. for B into one for A:

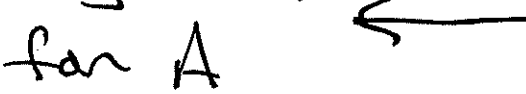
(1) given α instance of A, convert to β inst. of B

(2) run alg for B on β

(3) return answer (YES or NO) as answer for α.

Picture:

Decision Prob	A	B
Reduction	α	β
Conversion of Algorithms	Algorithm for A	Algorithm for B



Suppose reduction procedure runs in time $O(f(n))$.

Suppose Alg. for B runs in time $O(g(n))$. Then, derived algorithm for A runs in time $O(f(n) + g(n))$

In particular, if $f(n) = O(n^d)$, then conversion takes Poly-time algorithms for B to Poly-time algorithms for A. So A is no harder than B

Write: $A \leq_P B$

If B is 'easy' so is A.

Also if A is 'hard', so is B.

Ex. let A be played by HC

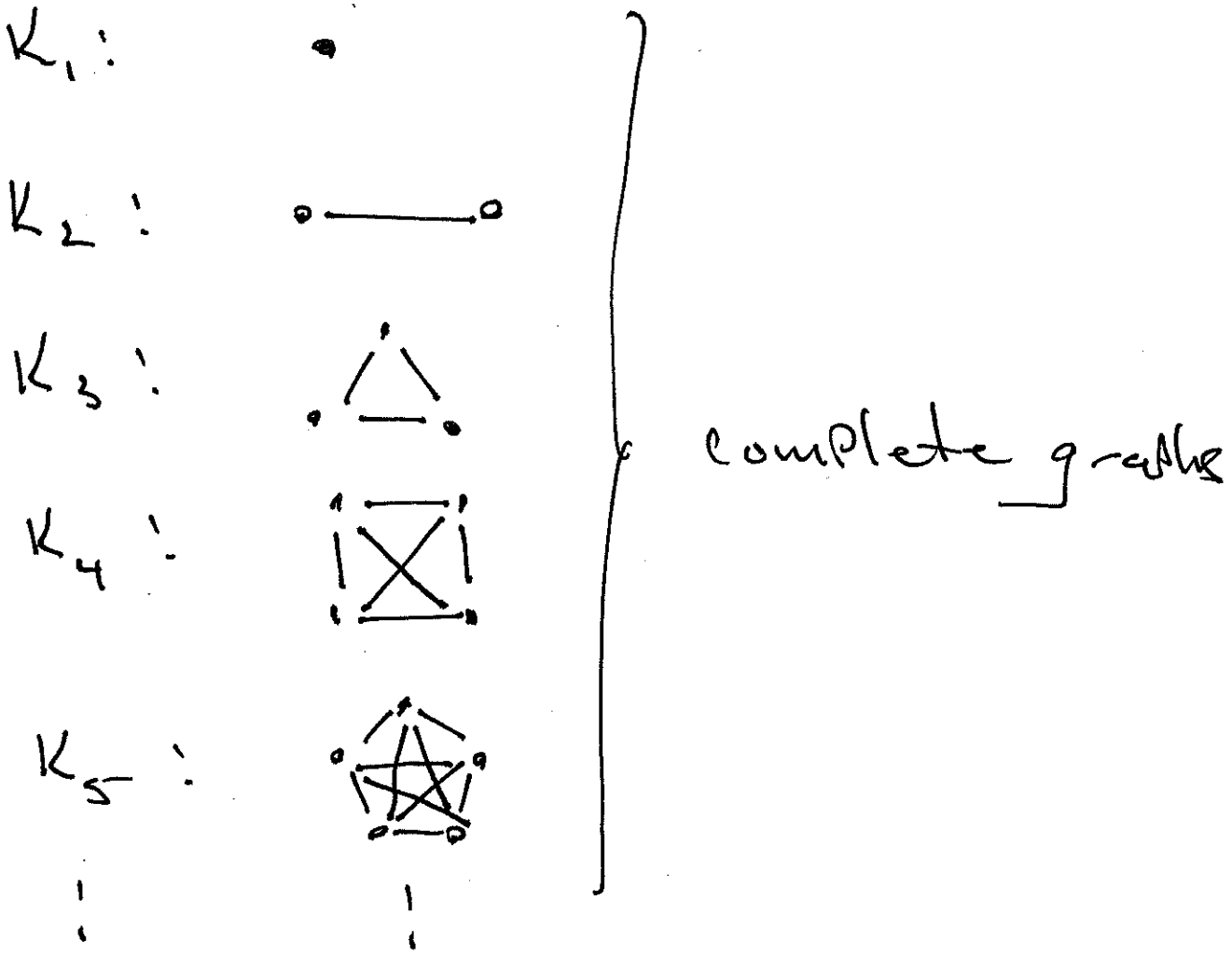
let role of B be played by

Traveling-Salesman-Problem (TSP)

Instance: $n \in \mathbb{Z}^+$, $w : E(K_n) \to \mathbb{R}^+$,

a bound $b > 0$

↑ complete graph on n vertices.



Solution: determine if there exists (YES/NO) a Hamiltonian cycle C in G of weight no more than b:

$$w(C) \leq b .$$