(ii) \( \textbf{WTime} \) is the class of languages recognizable
by some \( \text{NTM} \) running in (polynomial time) /
(time poly bounded by a polynomial \( p(n) \) of size
of input)

(i) "... deterministic TM ...

(iii) \( \textbf{PSPACE} \) is the class of languages recognizable
by a deterministic TM running using an amount
of space on its work tapes bounded by a
poly \( p(n) \) of the size of its input.

(iv)

b) (i) \( \textbf{Horn-SAT} \) (prop clauses with at most
one positive literal)

(ii) \( \textbf{SAT} \) (propositional clauses)

(\( 3\text{-SAT} \))

(iii) \( \textbf{QBF} \) (quantified Boolean logic)

c) \( \textbf{C}^c = \{ \overline{\ell} \mid \ell \in C \} \)

where
\( \overline{\ell} = \Xi^* \setminus \ell \) (assume \( \Xi \) is alphabet
of \( \ell \))
d) \( L \in \text{PTIME} \) if there is a deterministic TM, say \( M \), recognizing \( L \) and running in polynomial time. To recognize \( \overline{L} \) in polynomial time, (compute reverse the accepting states of \( M \)).

Next use TM \( M^* \) which outputs \( N \) just in case \( M \) outputs \( Y \), and outputs \( Y \) otherwise.

Thus \( \text{Co-PTIME} \subseteq \text{PTIME} \). The reverse inclusion is similar.

Similarly for \( \text{PSPACE} = \text{Co-PSPACE} \):

simply reverse outputs of recognizing TMs.

Does not work for showing \( \text{NP} = \text{Co-NP} \) because \( M \) accepts \( L \) iff for all \( x \), \( x \in L \iff M \) has some accepting run on \( x \). But reversing accepting states does not produce a TM \( M^* \) s.t. for all \( x \), \( x \in \overline{L} \iff M^* \) has some accepting run on \( x \).
e#) Trivially, \( \text{PSPACE} \subseteq \text{NPSPACE} \). For 
reverse inclusion suppose \( L \in \text{NPSPACE} \). 
Let \( M \) be a TM recognizing \( L \) 
using space bounded by \( p \) (a polynomial). 
By S.T. there is a deterministic TM 
\( M^* \) recognizing \( L \) in space bounded by 
\( p^2 \) — also a polynomial. Hence 
\( L \in \text{PSPACE} \).

f#) Since \( \text{PSPACE} = \text{NPSPACE} \) by part d.e \( \)
\( \text{Co-PSPACE} = \text{Co-NPSPACE} \) 
But \( \text{PSPACE} = \text{Co-PSPACE} \) by 
part d.l \( \)
Hence \( \text{NPSPACE} = \text{Co-NPSPACE} \).
4 a) An Eulerian circuit is a sequence $u_1, u_2, \ldots, u_n$, of vertices of a graph $G$ such that for all $i$ ($1 \leq i < n$), $(u_i, u_{i+1}) \in E$ and each $e \in E$ occurs exactly once in this path.

b) A Hamiltonian circuit is a path such that each vertex occurs on this path exactly once.

c) A graph $G$ has an Eulerian circuit if every vertex has even degree and $G$ is connected.

Thus, $\text{EULER-CIRC} \in \text{PRIME}$ because we may simply examine each vertex in turn and check if it has even degree. This can obviously be done in polynomial time.

d) Given a list $c_1, \ldots, c_n$ of "cities" and a (upper triangular) matrix $M$ of distances, where $M_{ij}$ is the distance from $i$ to $j$, return length of a shortest tour of all cities.
e) Define the related problem

\[ \text{TSP-Feasibility} \]

Given \( c_1, \ldots, c_n, M_{ij} \) and \( k \geq 0 \)

return \( \text{YES} \) if there is a tour of all cities of length \( \leq k \)

in our.

\[ \text{TSP-Feasibility is NP-complete} \]

This is easy in following sense. Suppose we have a fast method of solving \( \text{TSP-Feasibility} \). Then we have a fast (polynomial) method of solving \( \text{TSP using binary search} \).

f) We gave a reduction from

\[ \text{TSP-Feasibility} \] to \( \text{Hamiltonian Circuit} \).

to \( \text{TSP-Feasibility} \).

Suppose \( G = (V,E) \) is an instance of Hamiltonian Circ. We manufacture an instance of \( \text{TSP-Feasibility} \) with the same answer.

Let cities be the vertices \( V \).
Let distance \( d_{uv} \) between \( u, v \) be

\[ 1 \quad \text{if} \quad (u,v) \in E \]
Let the bound be \( n = |V| \)

\( \Rightarrow \) Suppose \( G \) has a Hamiltonian circuit \( u_1, u_2, \ldots, u_{n}, u_{n+1} = u_1 \).

Clearly this is a tour in a TSP of length \( n \). So our TSP-feasibility instance is positive.

\( \Leftarrow \) Suppose \( u_1, u_2, \ldots, u_n, u_{n+1} = u_1 \) is a minimal cost length tour of all cities in our TSP, and length of tour is \( \leq n \).

Since all distances are \( \geq 1 \), no distance between \( u_i, u_{i+1} \) is ever \( 2 \). But this is only possible if \((u_i, u_{i+1}) \in E\)

i.e. \( u_1, \ldots, u_n, u_{n+1} \) is a Hamiltonian tour.