

First, continuing the recitation part:

Show $3SAT \leq_m^P$ DOMSET.

$\mathcal{C} = C_1 \wedge C_2 \wedge \dots \wedge C_m \quad \mathcal{C} \rightarrow \mathcal{G} =$

Idea: \mathcal{G} has "clause gadgets" D_1, \dots, D_m .
 In fact, each D_j is just a single node!
 D_j is dominated $\equiv C_j$ is satisfied.
 Choosing $x_i \in S \equiv x_i = 1$; choosing $\bar{x}_i \in S \equiv x_i = 0$.

If you work with a general 3CNF ϕ the problem is mixed clauses like $(x_1 \vee x_2 \vee x_3)$ and $(\bar{x}_4 \vee \bar{x}_5 \vee x_6)$.
 To handle this, give each x_i an "alter ego" y_i which is intended to mean $\neg x_i$.
 Then replace these cases of clauses... "Stunt" double... by $(x_1 \vee y_2 \vee x_3)$ and $(y_4 \vee y_5 \vee x_6)$ and
 This idea is called "double-rail logic." and $(x_1 \vee y_1) \wedge (\bar{x}_1 \vee \bar{y}_1) \wedge (x_2 \vee y_2) \wedge (\bar{x}_2 \vee \bar{y}_2) \wedge \dots \wedge (x_n \vee y_n) \wedge (\bar{x}_n \vee \bar{y}_n)$.
 forces $y_i = \neg x_i$. This now has the needed "Same Sign" property.

Connect each clause to the iff ϕ is satisfiable, which is true iff \mathcal{G} has a dominating set of size k .
 Also take $k = n$.
 With the extra "double-rail" rule, we get consistency.

Then \mathcal{G} defines a set of recommendations that can be followed in entirety iff ϕ is satisfiable, which is iff $x \in A$. So $A \leq_m^P AS$.

Can we do this so that all positive recommendations (besides those with some x_i required) have three people and all negative ("don't choose all") ones have only two?

Yes: \mathcal{G} had $(\bar{u} \vee \bar{w}) \wedge (\bar{v} \vee \bar{w}) \wedge (u \vee v \vee w)$.
 Don't choose both u and w . Choose at least one of three.
 $u=1 \equiv$ person u is chosen
 0 not chosen.

fit $k = n$.
 extra rules, we need.

$(x_i) \equiv$ choose x_i
 $(\bar{x}_i) \equiv$ Don't choose (all of) $\{x_i\}$.
 $\mathcal{G} \rightarrow$ "Choose at least one of u or v "
 "Don't choose all of u, v, w , i.e. omit at least one of u, v, w ."

$(x_1 \vee x_2 \vee x_3)$ and $(\bar{x}_4 \vee \bar{x}_5 \vee x_6)$ and $(\bar{x}_1 \vee x_2 \vee x_3)$
 $(y_2 \vee x_3)$ and $(y_4 \vee y_5 \vee x_6)$ and $(y_1 \vee x_2 \vee x_3)$
 $(\bar{x}_1 \vee \bar{y}_1) \wedge (x_2 \vee y_2) \wedge (\bar{x}_2 \vee \bar{y}_2) \wedge \dots \wedge (x_n \vee y_n) \wedge (\bar{x}_n \vee \bar{y}_n)$.
 $y_i = \neg x_i$. This now has the needed "Same Sign" property.

Lecture component:

Padding and Translation Theorem: Let $f(n) \geq n+1$ be a fully conv (constructible) function.

Then $\text{DTIME}[t_1(N)] \subseteq \text{DTIME}[t_2(N)] \Rightarrow \text{DTIME}[t_1(f(n))] \subseteq \text{DTIME}[t_2(f(n))]$.

Proof: Let any $A \in \text{DTIME}[t_1(f(n))]$ be given. Define $A' = \{x10^{f(n)-|x|-1} : x \in A\}$ where string has length $N = f(n)$, where $n = |x|$.

Then $A' \in \text{DTIME}[t_1(N)]$. Algn: on input w :
 If it does not break as $x10^m$ where $m = f(|x|) - |x| - 1$, reject.
 Else, run the original M_A on x . M_A takes $t_1(f(n))$ steps, where $n = |x|$. But in terms of $|w|$, $|w| = N = f(n)$.
 So as a function of N , this takes $t_1(f(n)) = t_1(N) = t_1(|w|)$ steps, because the initial parse check on w takes at most $f(n) \leq t_1(f(n))$ steps.

By hypothesis, there is a machine M_2 that accepts A' in time $t_2(N)$.
 Now build M'_A as follows:

- Input x
- Compute $w = x10^{f(n)-|x|-1}$
- Run $M_2(w)$, which takes at most $t_2(|w|)$ steps, and accept x iff M_2 accepts w .
- Valid: this runs in $t_2(|w|) = t_2(f(n))$ steps. So $A \in \text{DTIME}[t_2(f(n))]$. \square

Meta-Theorem: This arg. works for all 16 combinations of $\{\text{DTIME}, \text{NTIME}, \text{DSPACE}, \text{NSPACE}\}$ as ① or ②. And it works when t_1 and t_2 are collection of banks, like $O(n)$ and $n^{(n)}$ or $2^{O(n)}$.

Examples:
 P=NP = EXP=NTIME (with $f(n) = 2^n$).
 L=P \Rightarrow PSPACE=EXP.

Notes:
 B1: Connect each clause in the Turing machine that solves it.
 B2: Also take $K=N$.
 B3: With the extra blank n's, we get conv .
 D1: mixed cases like (x, v, y_1) which is intended to be (x, v, y_2, v, x_3) and $(x, v, y_1) \wedge (x, v, y_1)$ forces $y_1 = -x_1$.