(1a)

$$\log_{255}(1000^n) < n \cdot \log^{1000}(n) < 2^{1000} n^{3.5} < n^4 < 2^{n/1000}$$

(16)

$$57 \cdot n^{\sqrt{5}} + 39 \cdot \sqrt{n} \cdot 3^{\log_2 n} = 57 \cdot n^{\sqrt{5}} + 39 \cdot n^{\frac{1}{2}} \cdot 3^{\frac{\log_3 n}{\log_3 2}}$$

$$= 57 \cdot n^{\sqrt{5}} + 39 \cdot n^{\frac{1}{2}} \cdot n^{\log_3 2}$$

$$= 57 \cdot n^{\sqrt{5}} + 39 \cdot n^{(\frac{1}{2} + \log_3 2)}$$

$$\frac{1}{2} + \log_3 2 < 2 < \sqrt{5} \Rightarrow n^{(\frac{1}{2} + \log_3 2)} < n^{\sqrt{5}}$$

Thus,

$$57 \cdot n^{\sqrt{5}} < 57 \cdot n^{\sqrt{5}} + 39 \cdot \sqrt{n} \cdot 3^{\log_2 n} < (57 + 39) \cdot n^{\sqrt{5}}$$

Therefore,

$$57 n^{\sqrt{5}} + 39 \sqrt{n} \cdot 3^{\log_2 n} \in \Theta(n^{\sqrt{5}})$$

(1 C)

$$2^{\pi \log_2 n} = (2^{\log_2 n})^{\pi} = n^{\pi}$$

$$\frac{2^{\pi \log_2 n}}{n^3 (\log_2 n)^{20}} = \frac{n^{\pi}}{n^3 (\log_2 n)^{20}} = \frac{n^{\pi-3}}{(\log_2 n)^{20}} = \left(\frac{n^{\frac{\pi-3}{20}}}{\log_2 n}\right)^{20}$$

$$\lim_{n\to\infty} \left(\frac{n^{\frac{\pi-3}{20}}}{\log_2 n}\right)^{20} = \lim_{n\to\infty} \frac{n^{\frac{\pi-3}{20}}}{\log_2 n}$$

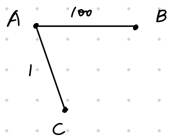
$$= \lim_{n\to\infty} \frac{\left(\frac{\pi-3}{20}\right)\cdot n^{\frac{\pi-23}{20}}}{\ln 2 \cdot \frac{1}{n}} \quad \text{by L'Hopital Rule}$$

$$= \lim_{n\to\infty} \left(\frac{\pi-3}{20\cdot \ln 2}\right)\cdot n^{\frac{\pi-3}{20}} \to \infty$$

Therefore, $2^{\pi \log_2 n}$ has a higher growth rate.

(2a) False.

counterexample:

G: 

G is a tree.

maximum weight edge: $w(A,B)=100$

minimum spanning tree: G.

(2b) False.

We know that the time complexity of Floyd-Warshall's dynamic programming algorithm is $O(n^3)$.
However, if we run Dijkstra's algorithm on all n vertices, we will have the time complexity $n \cdot O(m \cdot \log n)$. Since the graph is sparse, $m \in O(n)$. Thus, total run time of Dijkstra's algorithm is $n \cdot O(n \cdot \log n) = O(n^2 \log n) \in O(n^3)$.

Thus, Floyd-Warshall's dynamic programming algorithm is not asymptotically faster.

(2c.) False.

We can reduce CMBP to VERTEX COVER problem. thus, we can prove that VC problem is as hard as CBMP.

Thus, our conclusion can only be "if CMBP is NP-hard then VC is NP-hard". The reverse is not true.

(2d)

False.

PCP is undecidable problem since HALTING problem reduces to it. Thus, it has no solution.

(2e)

For ax and bz pairs,

we can see that company x prefers b over a,

while company b prefer x over z.

Thus, {ax, bz, cy} is instable.

(2f)

① a proposes to x, b proposes to x, c proposes to z.

$\Rightarrow$ x and b are provisionally engaged.
z and c are provisionally engaged.
a remains unengaged.

② a proposes to y.

$\Rightarrow$ (x, b) (y, a) (z, c)

(29)

Decision version :

Given n positive integers $a_1, \dots, a_n$, a number k and a positive W and a bound b.
Is it possible for us to find a subset $S \subseteq \{1, \dots, n\}$ such that $|S| = k$ and $|\sum_{i \in S} a_i - W| \leq b$?

(2h)

Assume we can solve the decision problem stated in my (2g) in polynomial time.

Then, if we do binary search on the bound b, we can find the smallest integer b that the answer to the decision problem is YES.

The set of that solution to the decision problem is the solution to the optimization problem.

(3)

We will design the algorithm by creating a "meta-graph" and then check if the "meta-graph" is bipartite.

① Constructing the "meta-graph":

For convenience, we let n_i represents the i-th participants in N.

Assume we have $V_0 = \{v_1, \cdots, v_n\}$ where v_i represents i-th participants, n_i, in N, and $E = \phi$.

Then, we check all k requests and create the edges and necessary "new vertex":

i) if the request is "I want to be in the same team as n_j" by n_i ($n_i, n_j \in N, i \neq j$),

then, we create a new vertex v_{ij}, connect v_i to v_{ij} and connect v_j to v_{ij}.

i.e. $V' = V \cup \{v_{ij}\}$, $E' = E \cup \{(v_{ij}, v_i), (v_{ij}, v_j)\}$

ii) if the request is "I do not want to be in the same team as n_j" by n_i ($n_i, n_j \in N, i \neq j$),

then, we connect v_i to v_j.

i.e. $E' = E \cup \{(v_i, v_j)\}$

Now, we have will have our "meta-graph", $G = (V', E')$

② Then, we will use the BFS-based Bipartite Checking Algorithm to check if G is bipartite.

If G is bipartite, we will have two sets of vertices that satisfy the definition of bipartite,

then we remove all vertices $v_{ij} \notin V_0$ and return the two sets as two teams.

If G is not bipartite, we return "Not possible".

Proof of correctness:

For ① i), if n_i wants to be on the same team with n_j, then by connecting v_i, v_j to the same vertex v_{ij}, we make sure that v_i, v_j have the same color, and when later bipartite checking n_i, n_j must be on the same team.

For ① ii), if n_i does not want to be on the same team with n_j, then by connecting v_i and v_j, we make sure that v_i and v_j have different colors, and when later bipartite checking n_i, n_j will not be on the same team for sure.

Time complexity:

For constructing G,

when initializing V, E, it requires $O(n)$ since there are n participants $\Rightarrow$ n vertices;

when adding edges and new vertices, we at most add k new vertices and $2k$ edges,

at least add k edges.

$\Rightarrow k \le |E'| \le 2k \quad n \le |V'| \le n+k$

and this requires $O(k)$ since there are k requests.

For checking bipartite,

the BFS-based Bipartite Checking Algorithm requires $O(n+m)$ where $n = |V|$, $m = |E|$. Thus, it requires $O(n+3k)$

Therefore,

for total time complexity, $T(n,k) = O(n) + O(k) + O(n+3k) \in O(n+k)$.

(4a)

Suppose the instance I is $G_D = (V_D, E_D)$ and d.

Assume the solution is a set of vertices, U.

My verification algorithm is:

① check for $\forall v \in U$, $v \in V_D$

② check for $\forall v \in V_D \setminus U$, $\exists u \in U$ such that $(u, v) \in E_D$

① can be done in $O(|U|)$ and ② can be done in $O((|V_D|-|U|)\cdot|E_D|)$.

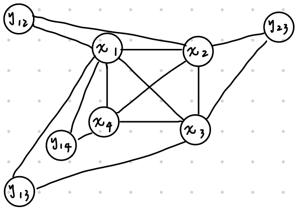
Since $|U|$, $|V_D|$, $|E_D|$ should be finite, then we can verify that I is indeed YES in polynomial time.

Thus, DS is in NP.

(4b) $f(I) = G_D(V_D, E_D)$:

$V_D = \{x_1, x_2, x_3, x_4, y_{12}, y_{23}, y_{13}, y_{14}\}$

$E_D = \{(x_1, x_2), (x_1, x_3), (x_1, x_4), (x_2, x_3), (x_2, x_4), (x_3, y_4),$
$(x_1, y_{12}), (x_1, y_{13}), (x_1, y_{14}), (x_2, y_{23}), (x_3, y_{23}), (x_3, y_{13}), (x_4, y_{14})\}$



(4c)

Suppose $I = (G(V,E), k)$ is a YES-instance of VC.

Then, $\exists S$ such that $|S| = k$ and $\forall (u,v) \in E$, either $u \in S$ or $v \in S$

Let $f(I) = (G_D(V_D, E_D), d)$ and $U = \{x_v : v \in S\}$.

We will show that U is a solution to DS:

First, we want to make sure that $U \subseteq V_D$:

$V_D = X \cup Y$ where

$X = \{x_v : v \in V\}$ and $Y = \{y_e : e \in E\}$.

Since $U = \{x_v : v \in S\}$ and $S \subseteq V$,

$U \subseteq X \Rightarrow U \subseteq V_D$

Next, let's investigate $V_D \setminus U$:

$V_D \setminus U = (X \cup Y) \setminus U = (X \setminus U) \cup (Y \setminus U)$

and $X \setminus U = \{x_v : v \in V \setminus S\}$

$Y \setminus U = Y$.

Thus, for every vertex $v \in V_D \setminus U$, $v \in \{x_v : v \in V \setminus S\}$ or $v \in \{y_e : e \in E\}$

i) If $v \in \{x_v : v \in V \setminus S\}$.

Since $T = \{(x_u, x_v) : x_u, x_v \in X, x_u \neq x_v\} \subseteq E_D$,

$\forall u \in U \subseteq X$, $(u, v) \in E_D$, that is v is adjacent to a vertex $u \in U$.

ii) If $v \in \{y_e : e \in E\}$.

$\exists u \in S$ such that $u \in e$.

Then $x_u \in U$.

Since $W = \{(x_u, y_e) : u \in V, e \in E, u \in e\}$,

x_u and y_e are adjacent.

Thus, in either case, $\forall v \in V_D - U$, v is adjacent to a vertex $u \in U$.

Therefore, U is indeed a solution to DS.

In other words, $f(I)$ is a YES-instance of DS.

(4 d)

Suppose $f(I)$ is a YES-instance of DS. Let the solution set be U.

By lemma, $\exists\, U' \subseteq X$ s.t. U' is also a dominating set.

We can see that all vertices in U' could corresponds to a vertex in E.

Assume that set to be S.

Since U' is a DS instance, for $v \in V_D \setminus U'$, $\exists\, u \in U'$ that $(v,u) \in E_D$. For this $v \in V_D \setminus U'$, it either $\in Y$ or $\in X \setminus U'$

Now,

i) If $v \in Y$, then,

if $v \in e$ then $u \in e$, which means all $v \in Y$ has a vertex $u \in S$ that $(v,u) = e \in E$.

ii) For $v \in X \setminus U'$, we won't have to look into it since the other vertex is not in V.

Thus, for $Y = \{y_e : e \in E\}$, there is a vertex $v \in S$ that $(v, y_e) \in E_D$. Since Y corresponds to every edge in $G(V,E)$, S is a CV set.

Thus, I is a YES-instance of CV problem with the solution set S.

Proof of lemma:

(5)

We want to construct a reduction from 3SAT to MS.

The input of 3SAT is a boolean function F in 3CNF form and n variable $\{x_1, \dots, x_n\}$ and m clauses $\{C_1, \dots, C_m\}$.

The output would be a set of n meetings, $\mathcal{M} = \{M_1, \dots, M_n\}$, a subset $\mathcal{M}_S = \{S_1, \dots, S_\ell\} \in \mathcal{M}$ of size ℓ, and a positive number K.

We use the following algorithm for the reduction:

① Let $\mathcal{M} = \{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n, y\}$ where y is a new element and $|\mathcal{M}| = 2n+1$.

② For each $C_i = \alpha_{i1} \vee \alpha_{i2} \vee \alpha_{i3}$ where $i \in \{1, \dots, m\}$, we construct $S_i = \{\alpha_{i1}, \alpha_{i2}, \alpha_{i3}, y\}$

And for each $i \in \{1, \dots, n\}$, we construct a subset $S_{m+i} = \{x_i, \bar{x}_i\}$.

③ We let $K = 2$.

Then, we will get an input to our MS problem. And we can see that this can be done in $O(n+m)$ time, which is polynomial time.

Now, we will prove that:

For 3SAT, $\exists$ an assignment that satisfies all clauses $\Leftrightarrow$ $\exists$ a mapping $f: \mathcal{M} \to \{1, \dots, K\}$ such that $\{c \in S_i : f(c) = j\} \neq S_i$ for all i, j.

① "$\Rightarrow$", suppose we have a YES-instance of 3SAT.

We try to define the following mapping:

f: $f(y) = 2$

$f(\alpha) = 1$ if α is assigned true

$f(\alpha) = 2$ if α is assigned false.

Then, we can see that

$\{c \in S_i : f(c) = 1\}$ has the true literals in C_i;

$\{c \in S_i : f(c) = 2\}$ has the false literals in C_i and y,

and neither set is equal to S_i.

For $\{c \in S_{m+i} : f(c) = 1\}$ and $\{c \in S_{m+i} : f(c) = 2\}$, they will have only one element and so they are not equal to S_{m+i}

Therefore, $\{c \in S_i : f(c) = j\} \neq S_i$ for all i, j.

Therefore, the MS instance above is a YES-instance of MS problem.

② "$\Leftarrow$", suppose we have a YES-instance of MS

We will try construct the following assignment:

For each literal α, assign α to be true, if $f(\alpha) \neq f(y)$, and false if $f(\alpha) = f(y)$

Then,

for $i \in \{1, \dots, n\}$, one of x_i and $\bar{x}_i$ will be one true, one false since $\{c \in S_{m+i} : f(c) = j\} \neq S_{m+i}$ for $j = 1, 2$.

for $i \in \{1, \dots, m\}$, not all 3 literals in clause C_i are false since $\{c \in S_i : f(c) = j\} \neq S_i$ for $j = 1, 2$.

Thus, C_i is satisfied for all i.

Therefore, we can prove that MS problem is NP-complete.