Problem 1. (a) In the minimum cycle cover, each vertex in the graph will have exactly one incoming and one outgoing edge in order to be part of a cycle. Thus, we can create an auxiliary graph G', and for each vertex $v \in G$, we will create two nodes $v_{in}, v_{out} \in G'$, such that the two sets of in-nodes and out-nodes will form a bipartition of G'. Additionally, for each edge $(u, v) \in E(G)$, we will add a corresponding edge (u_{in}, v_{out}) to G'. Then we can run the bipartite perfect matching algorithm (in polynomial time) on G' to obtain a perfect matching of in-nodes to out-nodes, which is equivalent to a minimum cycle cover since each vertex has one incoming and one outgoing edge. Thus, we can find a minimum cycle cover in polynomial time.

(b) By definition, a cycle must contain at least 2 nodes, so in the worst case there might be $n/2$ cycles consisting of two nodes each, and therefore $n/2$ representative nodes. The optimum tour traversing only the representative nodes has a cost that is at most equal to the original optimum; we can see this by noting that regardless of whether we are traversing the full graph or only representative nodes, we still need to "hop" from one cycle to the next using the minimum-cost edges between cycles, and these hops do not change in the representative node traversal. Also, for each cycle, the representative node traversal need only traverse part of it (i.e. the representative node to the hopping node), whereas the full traversal needs to traverse the entire cycle. Thus, the representative node traversal costs at most as much as the full traversal.

(c) If we continuously apply the algorithm from part (b), traversing $n/2$ nodes each iteration and marking them from the graph, then after $O(\log n)$ iterations we will eventually be left with a single node. Each iteration's traversal over the corresponding representative nodes constitutes a cycle itself, so if we unravel all the iterations and their respective traversals, we get a minimum cycle cover that constitutes a tour of the whole graph. Thus, since there are $O(\log n)$ iterations, each of which produces a representative node traversal with cost at most as much as the full traversal, we can produce a tour of the whole graph with cost $O(\log n)$ times the optimum.

Problem 2. (a) Suppose the total completion time is $T = \sum C_j$. If a job j completes at time t, we know that $x_{jt} = 1$ and $x_{jt'} = 0$ for all $t' \neq t$. Thus, to enforce that every job completes, we have:

$$\sum_{t=0}^{T} x_{jt} = 1 \qquad \forall j$$

Secondly, each job j completing at time C_j must not be processed before its predecessors $A(j)$, so for each predecessor job $i \in A(j)$, there must be a time $t < C_j$ such that $x_{it} = 1$. Thus, we have

$$\sum_{t=0}^{C_j} x_{it} \geq \sum_{t=0}^{C_j} x_{jt} \qquad \forall i \in A(j), i$$

Lastly, the total processing time of jobs completed before time t must be at most t, so we have

$$\sum_{s=0}^{t} \sum_{j} x_{js} p_j \leq t \qquad \forall t \in \{1, 2, \ldots, T\}$$

We want to minimize the weighted average completion time, which is $\sum w_j C_j = \sum w_j \sum_t x_{jt} t$. Thus, we have the following ILP:

$$\begin{aligned}
\min \qquad & \sum w_j \sum_{t=0}^{T} t x_{jt} \\
\text{subject to} \\
\sum_{t=0}^{T} x_{jt} &= 1 && \forall j \\
\sum_{t=0}^{C_j} x_{it} &\geq \sum_{t=0}^{C_j} x_{jt} && \forall i \in A(j), i \\
\sum_{s=0}^{t} \sum_{j} x_{js} p_j &\leq t && \forall t \in \{1, 2, \ldots, T\} \\
x_{jt} &\in \{0, 1\} && \forall j, t
\end{aligned}$$

(b) Note that at the halfway point h_j for a job j, at least half of the job has already been completed, so we have

$$\sum_{t=0}^{h_j} t x_{jt} \geq \sum_{t=h_j+1}^{T} t x_{jt}$$

and since we also have $\sum_{t=0}^{T} x_{jt} = 1$, it follows that $\sum_{t=0}^{h_j} x_{jt} = \sum_{t=h_j+1}^{T} x_{jt} = \frac{1}{2}$. Then, if we rewrite $\overline{C}_j$ into halves, we get

$$\overline{C}_j = \sum_{t=0}^{T} t x_{jt} = \sum_{t=0}^{h_j} t x_{jt} + \sum_{t=h_j+1}^{T} t x_{jt} \geq \sum_{t=0}^{h_j} t x_{jt} + \sum_{t=h_j+1}^{T} h_j x_{jt} \geq h_j \sum_{t=h_j+1}^{T} x_{jt} \geq \frac{h_j}{2}$$

Thus, $\overline{C}_j \geq h_j/2$.

(c) Scheduling the jobs in order of halfway point preserves the invariant that no job runs before its predecessors. To see this, suppose that for a job j there exists a predecessor $i \in A(j)$ such that $h_j < h_i$ so that j runs before i in halfway point order. In this case, we would have

$$\sum_{t=0}^{h_j} x_{jt} > \sum_{t=0}^{h_j} x_{it}$$

but this violates the second constraint in part (a), which is

$$\sum_{s=0}^{t} x_{is} \geq \sum_{t=0}^{t} x_{js} \qquad \forall i \in A(j), i, t$$

and therefore we have a contradiction, so no job runs before its predecessors.

(d) After scheduling the jobs in order of halfway point (so that job i runs before job j if $h_i < h_j$), the third constraint from part (a) implies that at time $t = h_j$, we have

$$\begin{aligned}
\sum_{s=0}^{h_j} \sum_{j} x_{js} p_j &\leq h_j \\
\sum_{i=1}^{j} \sum_{s=0}^{h_i} x_{is} p_i &\leq h_j \qquad \text{from part (c)} \\
\sum_{i=1}^{j} \frac{p_i}{2} &\leq h_j \qquad \text{since} \sum_{s=0}^{h_i} x_{is} = \frac{1}{2} \\
\sum_{i=1}^{j} p_i &\leq 2h_j \\
C_j &\leq 4\overline{C}_j \qquad \text{from part (b)}
\end{aligned}$$

Thus, the actual completion time for job j in the given order is at most $4\overline{C}_j$.

(e) Using the LP relaxation from part (b), we can solve for this relaxed LP and process the jobs in order of their halfway points. This results in a schedule that is at most 4 times the optimum schedule for each job, and therefore we have a constant 4-approximation of $1|prec|\sum w_j C_j$.

Problem 3. (a) Given subsets $S_1, S_2, \ldots S_n$, let $x_i = 1$ if S_i is chosen and 0 otherwise. Then we can formulate an integer LP as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ \text{subject to} \quad & \\ \sum_{i|v\in S_i} x_i \;&\geq\; 1 \qquad && \forall v \in U \\ x_i \;&\in\; \{0,1\} \qquad && \forall i \end{aligned}$$

which we can then relax into

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ \text{subject to} \quad & \\ \sum_{i|v\in S_i} x_i \;&\geq\; 1 \qquad && \forall v \in U \\ x_i \;&\geq\; 0 \qquad && \forall i \\ x_i \;&\leq\; 1 \qquad && \forall i \end{aligned}$$

(b) Applying randomized rounding and treating each x_i as the probability of choosing that set to be included in the cover, we see that the probability of v being covered is

$$p(v) = 1 - \prod_{i|v\in S_i} (1 - x_i)$$

By the second and third constraint from part (a), each of the individual $(1 - x_i)$ terms are bounded between $0 \leq (1 - x_i) \leq 1$, and by the first constraint $\sum_{i|v\in S_i} x_i \geq 1$, there must be at least some non-zero x_i, and therefore at least one term $(1 - x_i) \neq 1$. It follows that $0 \leq \prod_{i|v\in S_i}(1 - x_i) < 1$, which implies that $0 < p(v) \leq 1$ for all v. Thus, there is a non-zero, constant probability $\prod p(v)$ that each vertex is covered.

(c) After running the randomized rounding from part (b) k times to produce covers $C_1, C_2, \ldots C_k$, for each subset S_i, let X_{ij} be the indicator variable for the event that S_i is chosen by cover C_j. Then by the union bound, the probability that S_i is chosen by at least one of the covers $C_1, C_2, \ldots C_k$ is

$$P\left(\bigcup_{j=1}^{k} X_{ij}\right) \leq \sum_{j=1}^{k} P(X_{ij})$$

Suppose we run the randomized rounding for $k = \log(n)$ times. Then each element will have a 99/100 probability of being covered by at least one of the covers. Thus, if we combine all the $\log n$ covers, we get a cover that covers every element with probability 99/100 while using $O(\log n)$ times the optimum number of sets.

Problem 4. (a) Let $f_{ij}^k = 1$ if the k-th demand pair path uses (i, j), and denote the source and sink as s_k and t_k respectively. For each vertex, we must ensure conservation of flow (except for the sources and sinks), and we also want each edge to be part of at most only one path, in order to get disjoint paths. Lastly, we also want the flow through each path to be 1 (i.e. the outgoing flow from s_k and incoming flow to t_k should be 1). Thus, we can devise the following integer LP:

$$\begin{aligned}
\min \quad & \sum_{i,j,k} f_{ij}^k \\
\text{subject to} & \\
\sum_j f_{ij}^k - f_{ji}^k &= 0 && \forall i \notin \{s_k\} \cup \{t_k\} \\
\sum_j f_{ij}^k - f_{ji}^k &= 1 && \forall i \in \{s_k\} \\
\sum_j f_{ij}^k - f_{ji}^k &= -1 && \forall i \in \{t_k\} \\
\sum_k f_{ij}^k - f_{ji}^k &\leq 1 && \forall i, j \\
f_{ij}^k &\in \{0, 1\} && \forall i, j, k
\end{aligned}$$

(b) The relaxation of the ILP makes the last constraint becomes $0 \leq f_{ij}^k \leq 1$ for all i, j, k. Thus, within each demand pair, each source still sends out 1 total unit of flow to each sink, but these flows can be split among multiple fractional paths.

To read the paths from the solution, we can run a depth-first search on each source. For each source s_k, the DFS will find a fractional path to the corresponding sink t_k; we can then subtract the value of the edge with the minimum flow from the path, and run the DFS again until all the flow is exhausted for each demand pair. A single DFS iteration requires $O(m)$ time, and since at least one edge is exhausted on each iteration, the total number of iterations is at most $O(m)$ for each source, and there can be at most $O(n^2)$ sources (from $\binom{n}{2}$ demand pairs). Thus, the overall runtime for reading the paths is $O(n^2m^2)$.

(c) Applying randomized rounding to the relaxation from part (b) and treating each of the fractional path flows for a given demand pair k as the probability of sending all the flow of the demand pair through that particular path, we see that the probability of edge (i, j) being chosen as part of the final path for k is $p_{ij}^k = f_{ij}^k - f_{ji}^k$. Let X_{ij}^k be the corresponding indicator variable for this edge being chosen by demand pair k. From the fourth constraint in part (a), we have $\sum_k f_{ij}^k - f_{ji}^k \leq 1$. Thus, we have for each edge (i, j):

$$\mathbb{E}\left[\sum_k X_{ij}^k\right] = \sum_k \mathbb{E}\left[X_{ij}^k\right] = \sum_k f_{ij}^k - f_{ji}^k \leq 1$$

Thus, by the Chernoff bound,

$$P\left(\sum_k X_{ij}^k > O(\log n)\right) < 1/n^{10}$$

so the probability of an edge carrying more than $O(\log n)$ paths is less than $1/n^{10}$, and since there are at most $O(m) = O(n^2)$ edges, by applying the union bound we see that the probability that *any* edge carries more than $O(\log n)$ paths is bounded by $O(1/n^8)$. Thus, after a polynomial number of randomized draws of the algorithm, we will have obtained a solution where every edge carries at most $O(\log n)$ paths in polynomial time.