Problem 1

First find the min cut of G in polynomial time. Suppose this min cut is C, and has edge $e_1, e_2, \ldots, e_k \in C$.

For each edge e_i, increase its capacity by 1 and name this new graph G_i. First find the min cut of G_i in polynomial time. Suppose this min cut is C_i.

- if the volume of C = the volume of C_i, then it means there exist another min cut which doesn't include edge e_i. Therefore G doesn't have a unique min cut.
- if the volume of C < the volume of C_i for all i, then G has a unique min cut.

The number of edges in C is $O(E)$, therefore we run the algorithm to find min cut at most $O(E)+1=O(E)$ times. Since the algorithm to find min cut runs in polynomial time, this whole algorithm runs in polynomial time.

Problem 2

construct the flow network, name this graph G:

1. Let the cell $(1,1)$ be the source node s.
2. Let the cell (n,m) be the sink node t.
3. For each cell (x,y) in the grid:
    - If cell (x,y) is an obstacle, then no node is connected to or from it.
    - Otherwise, create edges to adjacent cells $(x,y+1)$ and $(x+1,y)$ with capacities 1 if those cells are within the grid bounds and not obstacles.

After the construction, run the polynomial time algorithm to find the max flow value f on this graph G. The answer will be f.

Problem 3

Construct a flow network:

1. Each menu item i corresponds to a node v_i.
2. Each customer j corresponds to a node u_j.
3. There is a source node s and a sink node t.
4. Connect the source node s to each customer node u_j with an edge of capacity 1.
5. Connect each customer node u_j to the menu items they are willing to order with an edge of capacity 1.
6. Connect each menu item node v_i to the sink node t with an edge of capacity d_i.

Find the $s-t$ max flow on this graph. The maximum number of customer we can serve will be the value of max flow.

Problem 4

1. Convert the vertex capacities to edge capacities:
    1. Split every vertex v into two vertices v_{in} and v_{out}
    2. Create an edge (v_{in}, v_{out}) with capacity c_v, which is the capacity of v.
    3. Connect all the original incoming edges to v_{in} and all the original outgoing edges to v_{out}. In this case, it means connecting all the neighbors of v to both v_{in} and v_{out}.
2. Construct the flow network:
    1. Create a source node s. Connect s to all the m start vertices with capacity 1.
    2. Create a sink node t. Connect all the boundary vertices to t with capacity ∞.
3. Apply a max flow algorithm on the transformed graph from s to t.

    if $f = m$, then it is possible. Otherwise, no.

Problem 5

Verifier

- Certificate y is k subsets from D.
- Check whether these subsets are mutually disjoint.
- If so, output 1, else output 0.

If x is yes instance

- Then there exist k subsets from D which are mutually disjoint.
- Give y to V, and V outputs 1.

If x is no instance

- Then every k subsets from D which are not mutually disjoint.
- So V outputs 0, no matter what k subsets it gets.

V runs in polytime

- Checking whether these k subsets are mutually disjoint takes $O(C_k^2) = O(k^2)$ time.

Problem 6

First prove the problem is in NP.

Verifier

- Certificate y is a course schedule.
- Check whether the courses conflict under this schedule by checking the course assignment of each student.
- If so, output 0, else output 1.

If x is yes instance

- Then there exist a satisfiable schedule that can schedule the courses without conflicts.
- Give y to V, and V outputs 1.

If x is no instance

- Then every possible schedule cannot schedule the courses without conflicts.
- So V outputs 0, no matter what schedule it gets.

V runs in polytime

- Check whether the course assignment of each student conflicts takes $O(|R||C|)$ time.

Then reduce form the 3-COLOR problem.

Given an undirected graph $G = (V, E)$ and an integer $k = 3$, we construct $f(\langle G \rangle) = \langle C, S, R \rangle$ as follows:

1. $|C| = |V|$. Construct C with $|V|$ distinct elements.
2. $|S| = 3$. $S = \{1, 2, 3\}$.
3. For each edge e that connects the two vertices u and v, make $\{u, v\} \in R$.

Our reduction takes polynomial time because:

1. Constructing C with $|V|$ distinct elements takes $O(|V|)$ time.
2. Constructing $S = \{1, 2, 3\}$ takes $O(1)$ time.
3. Constructing R takes $O(|E|)$ time.

Therefore the whole reduction takes $O(|V| + |E|)$ time.

Then we prove the correctness of our reduction as follows:

1. Let $\langle G \rangle$ be a yes-instance of 3-COLOR. Then for each vertex in G, it has distinct color with any its adjacent vertex. For any pair $\{u, v\} \in R$, u and v are adjacent in G, so they must have different colors. Since the 3 colors are directly mapped to the 3 different time slots in S, they must have different time, which means that they never conflict.
2. Let $\langle G \rangle$ be a no-instance of 3-COLOR. Then for any 3-color assignment to V, there exist at least 2 adjacent vertices u, v in G that have the same color. Therefore, the corresponding pair $\{u, v\} \in S$ must have the same time slot, leading to a conflict.

Hence, this problem of determining whether a conflict-free schedule exists is NP-complete