1. 1. Algorithm: Turn it into a net flow problem. Construct a graph with $A_1, A_2, \ldots, A_k$ and all the elements that are in at least one of them as nodes. There's an edge between a set and an element of the element is contained in the set. Add two nodes s and t. s is connected to all the sets and t is connected to all the elements. Each of the edges has a capacity of 1. Then we can run Hopcroft–Karp–Karzanov algorithm on it.

   Correctness: The problem is actually a bipartite matching from A to the elements, so it can be solved by this algorithm.

   Time complexity: To construct the graph we need O(kn). As each of the edge has a capacity of 1, it's a special case of Hopcroft–Karp–Karzanov algorithm and its time complexity is $O(E\sqrt{V}) = O(kn\sqrt{k+n})$. So totally $O(kn\sqrt{k+n})$.

   2. Algorithm: We can just slightly adjust the graph in the former question: For the k nodes of sets, we connect the i_{th} to an element if the element is contained in both A_i and B_i. Then we do the same thing to it.

      Correctness: The problem is still a bipartite matching from A to the elements, so it can be solved by this algorithm.

      Time complexity: Still the same as problem 1. $O(kn\sqrt{k+n})$.

2. 1. First, constructing a level graph takes $O(|E| + |V|)$. Then for the iteration, finding a blocking flow in a level graph takes $O(|E|)$, as every edge is went through at most once. The number of iterations is $O(\sqrt{|E|})$ because all capacity are 1 and there's no parallel edges, and after $\sqrt{|E|}$ iterations, $dist^{G_f}(s,t) \geq \sqrt{|E|}$. There are at most $\sqrt{|E|}$ edge disjoint paths, so at most $\sqrt{|E|}$ iterations are left. So totally it runs in $O(|E|^{\frac{3}{2}})$ time.

   2. Also, constructing a level graph takes $O(|E| + |V|)$. Finding a blocking flow still takes $O(|E|)$.

      For the iteration: Let f be the flow after $2|V|^{2/3}$ iterations of the algorithm. Let D_i be the set of vertices at distance i from s in the residual network G_f. Suppose that for all i, $|D_i \cup D_{i+1}| > |V|^{\frac{1}{3}}$. Then $\sum |D_i| = \sum_i^{|V|^{\frac{2}{3}}} (|D_{2i-1} \cup D_{2i} > |V|^{\frac{2}{3}} \times |V|^{\frac{1}{3}} = |V|)$ Contradiction!

      So there must exist i such that $|D_i \cup D_{i+1}| \leq |V|^{\frac{1}{3}}$. As the max flow in G_f must go through D_i and D_{i+1}, the max flow in G_f if $O(|V|^{2/3})$. Because at least one is added to the flow, there are at most $O(|V|^{2/3})$ iterations left. So totally there are $O(|V|^{2/3})$ iterations.

      Overall the time complexity is $O(|V|^{2/3}|E|)$

3. 1. LP of maximum matching problem;

      maximum $\sum_{e \in E} x_e$

      subject to $\sum_{e:e=(u,v)} x_e \leq 1$

      $$x_e \in \{0, 1\}$$

      The two s.t. statements indicate at most one match and matching has two status respectively.

      Its LP_relaxation turns the second into $0 \leq x_e \leq 1$. And it is equivalent to $x_e \geq 0$ because $x_e \leq 1$ is weaker than the first statement.

      So it is an LP-relaxation of maximum matching problem.

   2. Dual:

minimize $\sum_{v \in V} y_v$

subject to $y_u + y_v \geq 1 \ \forall (u, v) \in E$

$$y_u \geq 0 \ \forall v \in V$$

LP for minimum vertex cover problem:

minimize $\sum_{v \in V} y_v$

subject to $y_u + y_v \geq 1 \ \forall (u, v) \in E$

$$y_v \in \{0, 1\} \ \forall v \in V$$

For same reason, the dual is an LP-relaxation to the minimum vertex cover problem.

3. 1. Base situation: 1*1 matrix in the incident matrix is either 1 or 0, so the determinant is either 1 or 0.

   2. Induction: For a $(k+1) \times (k+1)$ matrix, assume that for any sub-matrix below $k \times k$, the det is -1, 0 or 1.

      There at most two 1 in a column.

      If a column has no1 1, then the det of the $(k+1) \times (k+1)$ matrix, det = 0

      If a column has only one 1, then if we remove the 1 and the row and column of it, it makes a $k \times k$sub matrix. So, det is also -1, 0 or 1.

      If all column have two 1, All the rows are linearly independent because all vertices are connected to different vertices to each other in bipartite problem. So det = 0 and the incident matrix is totally unimodular.

   So the incident matrix is unimodular.

4. Coefficients in the original LP form the incident matrix and the dual forms the transpose of it. As the incident matrix is unimodular, the transpose is also unimodular. RHS of corresponding LP is also integral, so two LP have integral optimal.

   Consider integral optimal of two LPs, and we have $x*, y* \in \{0, 1\}$, making it the LP relaxation. So we can use the two LP-relaxations to represent max bipartite matching and minimum vertex cover. So we have proved König-Egerváry Theorem.

5. Counterexample: In a triangle, max matching is 1 and minimum vertex cover is 2.

4. 1. Modify the network: Introduce super s' and t'. For every $e = (u, v)$, set its capacity to $c_e - d_e$. Add edges to the graph: For every $e = (u, v)$, add (u,t') and (s',v) with capacity d_e .

   Then run the original max flow algorithm to determine the existence of a feasible flow by using the original max flow algorithm. If the out-edge of s' and in-edge of t' all reach full flow, then there's a feasible flow.

   Correctness: Checking if the out-edge of s' and in-edge of t' all reach full flow can guarantee that the demands can be met, and changing the original capacity guarantees that trying to meet the demands won't exceed the capacities.

   Time complexity: The number of vertices in the modified graph is $|V| + |E|$, so the time complexity will be $O((|V| + |E|)^2 |E|)$.

   2. Algorithm: First construct the network as 1 describes. Then run the original max flow algorithm on it to get the maximum flow of the modified network. Then remove new edges and s', t'. Find the maximum flow from original s to t in the residual network. Finally add the two flows and we can get the maximum flow from s to t.

Correctness: After meeting the demands with the algorithm of finding feasible flow. The problem is then turned into an original maximum network flow problem. So we can just perform the original max flow algorithm on the residual network for the extra flow that can be added to the feasible solution found in former steps. Finally, we just need to add them.

Time complexity: For the feasible flow, it is $O((|V|+|E|)^2|E|)$; For the original max flow on the residual network, it is $O(|V|^2|E|)$. So totally it will be $O((|V|+|E|)^2|E|)$.