1. The following figure shows a flow network on which a flow \( f \) has been computed. The value of the flow through an edge appears in the dotted arrow beside the edge. Edges without a dotted arrow carry zero flow.

- (10 marks) Is this a maximum flow? If so, prove it. If not, give a maximum flow.

\begin{itemize}
  \item This is not a maximum flow. The residual network for the flow in Figure (a) is shown in Figure (b). The path highlighted with thick edges is an augmenting path of residual capacity 2. After pushing 2 units of flow through the edges in this path we obtain the flow shown in Figure (c). By the max flow min cut theorem, this is a maximum flow as its residual network shown in Figure (d) has no augmenting paths. The value of the maximum flow is then 8.
  \item (10 marks) Find a minimum cut separating \( s \) from \( t \).
  
  In the residual network shown in above Figure (d) we mark all nodes that can be reached from the source, i.e. nodes \( s \) and \( c \). These nodes belong to the \( S \) side of a minimum cut; all other nodes are in the \( T \) side of the cut. Hence, a minimum cut for the network is \( \{s, c\}, \{b, d, e, t\} \). The capacity of this cut is \( 2 + 2 + 4 = 8 \). Note that edge \( (b, c) \) does not contribute to the capacity of the cut as this edge goes from the \( T \) side of the cut to the \( S \) side of the cut.
\end{itemize}
2. Let \( G = (V, E) \) be a flow network with source \( s \), sink \( t \), and integer capacities. As always we denote with \( n \) the number of vertices of \( G \) and with \( m \) the number of edges. A minimum cut for \( G \) is a cut with minimum capacity. \( G \) might have several minimum cuts, some of them having more edges than others. For example for the following flow network the following are some of its minimum cuts: \((\{s\}, \{1, 2, 3, 4, 5, 6, t\})\), \((\{s, 1\}, \{2, 3, 4, 5, 6, t\})\), \((\{s, 1, 2\}, \{3, 4, 5, 6, t\})\), \((\{s, 1, 2, 3, 4, 5, 6\}, \{t\})\). All these cuts have capacity 4, but the first cut has 2 edges, the second one has 3 edges, the third one has 4 edges, and the last one has 1 edge. A maximal minimum cut is a minimum capacity cut with the largest number of edges. For the following flow network a maximal minimum cut is \((\{s, 1, 2\}, \{3, 4, 5, 6, t\})\).

- (10 marks) Write a polynomial time algorithm for computing a maximal minimum cut of a given flow network \( G \). You can assume that all capacities are integers multiples of \( m \).

**Note.** Since the time complexity of the algorithm must be polynomial do not try to list all the minimum cuts to find one with maximum number of edges, as such an algorithm would need exponential time in the worst case.

**Algorithm MaximalMinCut**

**In:** Flow network \( G = (V, E) \) in which all edge capacities are multiples of \( m \), where \( m = |E| \).

**Out:** A maximal minimum cut.

1. Build a new graph \( G' = (V, E) \) with the same vertices and edges as \( G \) but in which the capacity of each edge has been decreased by 1.
2. Compute a minimum cut \( C \) for \( G' \).
3. Output \( C \)

- (15 marks) Prove that your algorithm is correct.

To prove the correctness of the algorithm we need to show that it terminates and it produces the correct output.

**Termination.** Since the flow network \( G \) has a finite number of vertices and edges, then constructing \( G' \) takes finite time. Since all edges have integer capacities a minimum cut of \( G' \) can be computed in finite time (for example by using the algorithm of Ford and Fulkerson to compute a maximum flow and then computing the minimum cut from the corresponding residual network), and since a minimum cut of \( G' \) has finite size, it can be output in finite time as well.

**Correct output.** We need to prove two things: (i) that the cut \( C \) is a minimum cut of \( G \) and (ii) that \( C \) has the maximum number of edges among all minimum cuts of \( G \).

(i) We show that a cut \( C \) that is not a minimum cut in \( G \) cannot be a minimum cut in \( G' \); therefore, this will show that only minimum cuts in \( G \) can be minimum cuts in \( G' \) and since \( C \) is a minimum cut in \( G' \) then it must also be a minimum cut in \( G \).
Let $c_G(C)$ and $c_G(C')$ be the capacities of $C$ and $C'$ in $G$, respectively; let $c_{G'}(C)$ and $c_{G'}(C')$ be the capacities of $C$ and $C'$ in $G'$, respectively. Let $n(C)$ be the number of edges in $C$ and $n(C')$ be the number of edges in $C'$. Since $C'$ is not a minimum cut in $G$ then

$$c_G(C') > c_G(C)$$

Since the capacities of the edges in $G$ are multiples of $m$, then $c_G(C')$ and $c_G(C)$ must differ by at least $m$, or in other words

$$c_G(C') \geq c_G(C) + m \quad (1)$$

Because $G'$ was constructed by reducing in $G$ all edge capacities by 1, then

$$c_{G'}(C) = c_G(C) - n(C), \quad \text{and} \quad c_{G'}(C') = c_G(C') - n(C') \quad (2)$$

Then

$$c_{G'}(C') = c_G(C') - n(C') \quad \text{by (2)} \quad (3)$$

$$\geq c_G(C') - m \quad \text{because $C'$ has at most $m$ edges} \quad (4)$$

$$\geq c_G(C) + m - m = c_G(C) \quad \text{by (1)} \quad (5)$$

$$= c_{G'}(C) + n(C) \quad \text{by (2)} \quad (6)$$

$$\geq c_{G'}(C) + 1 \quad \text{because any cut has at least one edge} \quad (7)$$

$$> c_G(C) \quad (8)$$

Therefore, $C'$ is not a minimum cut in $G'$.

(ii) We now show that $C$ has the maximum number of edges among all minimum cuts of $G$. Let $C''$ be a minimum cut of $G$ with fewer edges that $C$, i.e. $n(C'') < n(C)$. Then

$$c_{G'}(C'') = c_G(C'') - n(C'') \quad (9)$$

$$= c_G(C) - n(C'') \quad \text{because $C$ and $C''$ are minimum cuts of $G$} \quad (10)$$

$$> c_G(C) - n(C) = c_{G'}(C) \quad \text{because $n(C'') < n(C)$} \quad (11)$$

Therefore algorithm MAXIMALMINCUT will not choose cut $C''$, and thus the algorithm always select a maximal minimum cut.

• (5 marks) Compute the time complexity of your algorithm in the worst case. The running time must be polynomial in the size of the input.

Constructing the graph $G'$ requires $O(m + n)$ time as the graph has $m$ edges and $n$ vertices.

We can compute a minimum cut for $G'$ by first using the algorithm of Edmonds and Karp to compute a maximum flow in $O(m^2n)$ time. Then the residual network corresponding to the maximum flow can be computed in $O(m+n)$ time. Finally, determining which vertices can be reached from the source in the residual network can be done in $O(m+n)$ time using breadth first search. Hence, a minimum cut for $G'$ can be computed in $O(m^2n)$ time. Outputting the minimum cut requires $O(n)$ time as we need to specify the vertices in each side of the cut.
3. A set \( B = \{B_1, B_2, \ldots, B_r\} \) of object needs to be transported from some city \( X \) to another city \( Y \). In order to ensure that the objects are not damaged during the transportation, each object needs to be placed inside a container.

There is a set \( C = \{C_1, C_2, \ldots, C_s\} \) of containers; each container can store only one object. The container where an object \( B_i \) is placed must have specialized straps to secure the object and climate control to provide certain temperature and humidity level. Hence, for each object \( B_i \) there is only a subset of containers \( S_i \subseteq C \) that can store it.

Each container \( C_i \in C \) has size \( d_i \). Several containers can have the same size. There is also a set \( T \) of trucks that will transport the containers. Each truck has compartments of fixed dimensions, so a truck can only transport containers of one size. A truck \( T_j \) can transport up to \( t_{jd_i} \) containers of size \( d_i \), but it cannot transport containers of any other size.

The problem is to determine whether all the objects \( B \) can be stored in the containers \( C \) and transported by the trucks \( T \) to their destination subject to the above constraints.

For example consider the sets of objects \( B = \{B_1, B_2, B_3, B_4\} \) and containers \( C = \{C_1, C_2, C_3, C_4, C_5\} \). Object \( B_1 \) can be stored only in \( C_1 \) or \( C_4 \), so \( S_1 = \{C_1, C_4\} \); the other sets \( S_i \) are as follows \( S_2 = \{C_1, C_4\}, S_3 = \{C_1, C_3\}, S_4 = \{C_2, C_3, C_5\} \). Containers \( C_1 \) and \( C_4 \) have size 3 and containers \( C_2, C_3, C_5 \) have size 2.

There are 2 trucks \( T = \{T_1, T_2\} \), where truck \( T_1 \) can transport 2 containers of size 3, i.e., \( t_{13} = 2 \), and truck \( T_2 \) can transport 2 containers of size 2, i.e., \( t_{22} = 3 \). This problem has a solution: store \( B_1 \) in \( C_1 \) and truck \( T_1 \), store \( B_2 \) in \( C_4 \) and truck \( T_1 \), store \( B_3 \) in \( C_3 \) and truck \( T_2 \), and store \( B_4 \) in \( C_2 \) and truck \( T_2 \). Note that this solution is not unique.

However, if truck \( T_1 \) can only transport one container of capacity 3, i.e., \( t_{13} = 1 \) and truck \( T_2 \) can transport 3 containers of size 2, then there is no solution for the problem.

- (20 marks) Write an algorithm for solving this problem. If there is a way of storing the objects in the containers and the containers in the trucks, the algorithm must return the value \textit{true}, otherwise it must return the value \textit{false}.

\textbf{Algorithm} \hspace{1em} \textbf{TransportObjects}(\( B, C, T, S \))

\textbf{In:} Set \( B \) of objects, \( C \) of containers, \( T \) of trucks, and family of sets \( S \).

\textbf{Out:} \textit{True} if all the objects can be stored in the containers and the containers in the trucks; \textit{false} otherwise

\begin{itemize}
  \item Build a flow network \( G_T = (E \cup C \cup T \cup \{s, t\}, E) \), where \( E = \{(s, B_i) \mid B_i \in B\} \cup \{(B_i, C_j) \mid B_i \in B, C_j \in C, C_j \in S_i\} \cup \{(C_i, T_j) \mid C_i \in C, T_j \in T, t_{jd_i} > 0\} \cup \{(T_i, t) \mid T_i \in T\} \). All edges have capacity 1, except the edges from nodes \( T_i \) to the sink \( t \). The capacity of edge \((T_i, t)\) is equal to the number of containers that truck \( T_i \) can transport.
  \item Compute an integer maximum flow \( f \) for \( G \).
  \item If \( f \) saturates all edges incident on \( s \) then \textbf{return} \textit{true}; else \textbf{return} \textit{false}
\end{itemize}

The flow network that the algorithm would construct for the first example given in the question is shown in the following figure.
(25 marks) Prove that your algorithm correctly solves the above problem.

To show that the algorithm is correct, we need to show that it terminates and it produces the correct output.

**Termination.** Note that flow network $G$ has finite size, therefore we can construct it in finite time. Since all edge capacities are integer, we can compute an integer maximum flow in finite time. Since there is a finite number of edges incident on the sink, we can check in finite time whether all of them are saturated. Hence, the algorithm must finish in a finite amount of time.

**Correct output.** To show that the output produced by the algorithm is correct we will prove that the following statement is true:

There is a way of assigning all objects to containers and containers to trucks if and only if $G_T$ has a flow that saturates all edges incident on $s$.

- If there is a way of assigning all objects to containers and containers to trucks then $G_T$ has a flow that saturates all edges incident on $s$.

Let $\pi$ be an assignment of objects to containers and truck so that all objects are packed according to the conditions stated in the question. We show how from $\pi$ we can build a flow function $f$ for $G_T$ that saturates all edges incident on the source.

  - Set $f(s, B_i) = 1$ for all $B_i \in B$.
  - If $\pi$ assigns object $B_i$ to container $C_j$, then set $f(B_i, C_j) = 1$. Note that since $\pi$ assigns $B_i$ to only one container, then flow conservation is satisfied at the nodes $B_i \in B$.
  - If $\pi$ assigns container $C_i$ to truck $T_j$, then set $f(C_i, T_j) = 1$. Since $\pi$ assigns a container to only one truck then flow conservation is satisfied at all nodes $C_i \in C$.
  - If $\pi$ assigns $d$ containers to truck $T_i$, then set $f(T_i, t) = d$. Note that $d$ is also equal to the flow coming into node $T_i$ as node $T_i$ only receives flow from the containers that $\pi$ assigned to it, hence flow conservation is satisfied at all nodes $T_i$.

It is easy to see that $f$ satisfies the capacity constraints at all nodes with capacity 1, as all these edges carry flow of value either 0 or 1. For the nodes $T_i$, $f(T_i, t) =$ number of containers assigned to truck $T_i \leq$ capacity of edge $(T_i, t)$, so capacity constraints are also satisfied at the edges $(T_i, t)$.

We have shown that $f$ is a valid flow function. Note that $f(s, B_i) = 1$ for all nodes $B_i$, hence $f$ saturates all edges incident on $s$. 

5
- $G_T$ has a flow $f$ that saturates all edges incident on $s$ then there is a way of assigning all objects to containers and containers to trucks.

* If $f(B_i, C_j) = 1$ then pack object $B_i$ in $C_j$. Notice that by the way in which the flow network $G_T$ was constructed, it is allowed for object $B_i$ to be stored in container $C_j$.

* If $f(C_i, T_j) = 1$ then store $C_i$ in truck $T_j$. Notice that by the way in which we selected the edges in $G_T$ from nodes $C_j$ to nodes $T_j$, truck $T_j$ can transport container $C_i$.

Observe that since $f$ saturates all edges incident on the source, then all objects are assigned to containers and each container storing an object is placed into a truck. Because of the way in which we selected the capacities for the edges, each truck carries at most as many containers as it is allowed to transport, so this assignment of objects to containers, and containers to trucks satisfies all conditions of the question.

• (5 marks) Compute the worst case time complexity of your algorithm.

Flow network $G_T$ has $O(|B| + |C| + |T|)$ nodes and at most $O(|B| + |B||C| + |C| + |T|)$ edges, hence $G_T$ can be constructed in $O(|B||C| + |T|)$ time.

If we use the algorithm of Ford and Fulkerson to compute a maximum flow, the time complexity of this step if $O(m|f^*|)$ where $m$ is the number of edges in $G_T$ and the value of the maximum flow $|f^*|$ is at most $\min\{|B|, |C|\}$ as the cut $(\{s\}, B \cup C \cup T \cup \{t\})$ has capacity $|B|$ and the cut $(\{s\} \cup B \cup C, T \cup \{t\})$ has capacity $|C|$. Hence, the time complexity of the algorithm of Ford and Fulkerson is $O((|B||C| + |T|)\min\{|B|, |C|\})$.

Finally, to check whether all edges incident on the source are saturated requires $O(|B|)$ time, so the time complexity of the algorithm is $O((|B||C| + |T|)\min\{|B|, |C|\})$. 