CS4445/9544 Analysis of Algorithms II
Solution for assignment 1

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.

The flow given is not maximum. If you build the residual network, you will find out that there is an augmenting path: $s, 1, 5, 2, 3, t$ of residual capacity 1. A maximum flow is given in the following figure.

![Flow Network](image)

(5 marks) Find a minimum cut separating $s$ from $t$.

A minimum cut is $(\{s, 1\}, \{2, 3, 4, 5, 6, t\})$ of capacity 10. Another minimum cut is $(\{s, 1, 2, 3, 5, 6\}, \{4, t\})$.

2. (10 marks) Consider the following two claims.

Let $G = (V, E)$ be a flow network with source $s$, sink $t$ and positive integer edge capacities. Let $f$ be a maximum flow for $G$. Determine whether the following claims are true or false. If a claim is true, give a proof for it. If a claim is false, give an example proving the claim wrong.

(10 marks) Claim 1. For any flow network $G = (V, E)$ with integer capacities, there is at least one edge $(u, v) \in E$ such that if the capacity of $(u, v)$ is increased by 1, the value of the maximum flow increases by 1.

This claim is false. If a flow network has at least two minimum cuts $(S, T)$ and $(S', T')$ where the sets of edges across the cuts, $C_1 = \{(u, v) \mid u \in S, v \in T\}$ and $C_2 = \{(u, v) \mid u \in S', v \in T'\}$, are disjoint, i.e. $C_1 \cap C_2 = \emptyset$, then increasing the capacity of one edge will not increase the value of a maximum flow. This is because in the modified flow network obtained by increasing the capacity of a single edge, at least one of $C_1$ or $C_2$ will still be a minimum cut and by the max-flow min-cut theorem, the value of the maximum flow will not change. For example, for the following network there are two minimum cuts. Increasing the capacity of one edge will not increase the value of the maximum flow.

![Minimum Cut](image)
Claim 2 For any flow network \( G = (V, E) \) with integer capacities, there
is at least one edge \((u, v) \in E\) such that if the capacity of \((u, v)\) is decreased by 1,
the value of the maximum flow decreases by 1.

This claim is true. Consider a minimum cut \((S, T)\) of \( G \) and any edge
\((u, v)\) with positive capacity, for which \(u \in S\) and \(v \in T\). Decreasing the capacity of this edge by 1
decreases the capacity of \((S, T)\) by 1 and thus by the max-flow min-cut theorem the value of the maximum
flow also decreases by 1.

Optional question: Write an algorithm for finding an edge \((u, v)\) as described
in Claim 1, if such an edge exists. Assume that a maximum flow \( f \) for
\( G \) is known. The algorithm must run in \( O(m + n) \) time, where \( n \) is the number of vertices and \( m \) is the number
of edges in \( G \).

3. Let \( G = (V, E) \) be a flow network with source \( s \), sink \( t \), and integer capacities. As always, let
\( m \) be the number of edges and \( n \) be the number of vertices of \( G \). A basal minimum cut is a
minimum capacity cut with the smallest number of edges. For example, for the following flow
network there are two minimum capacity cuts: \((\{s\}, \{1, 2, 3, t\})\) and \((\{s, 1, 2, 3\}, \{t\})\). Both
cuts have capacity 4, but the first cut has 2 edges while the second has 1 edge; hence the
second cut is a basal minimum cut.

![Diagram of a flow network](image)

• (10 marks) Write an efficient algorithm for computing a basal minimum cut
of a given flow network \( G \). Assume that all capacities are integers multiples
of \( m \).

The algorithm increases the capacity of every edge in \( G \) by 1 and then it finds a minimum
cut in the resulting graph. We show below that this is a basal minimum cut.

Algorithm BasalMinCut(\( G \))

Input: Flow network \( G = (V, E) \) with edge capacities \( c \). Each edge has capacity that
is a multiple of \( m \), the number of edges.

Output: A basal minimum cut.

1. Build a graph \( G' = (V, E) \) where the edge capacities are as follows: \( c'(u, v) = c(u, v) + 1 \) if \( c(u, v) > 0 \), and \( c'(u, v) = 0 \) if \( c(u, v) = 0 \).
2. Compute a maximum flow \( f \) of \( G' \) by using the algorithm of Edmonds and Karp
and determine a minimum cut \((S, T)\) from the residual network \( G'_f \).
3. Return \((S, T)\)

• (10 marks) Prove that your algorithm is correct.

To show that the algorithm is correct, we need to show that it terminates and that it
produces the correct answer. To show that the algorithm terminates, note that building
flow network \( G' \) requires a finite amount of time as it has a finite number of vertices and
edges. Since all edge capacities in \( G' \) are integer, then the algorithm of Edmonds and
Karp ends in a finite amount of time. Computing the residual network and a minimum cut also requires a finite amount of time, so the algorithm will terminate for any flow network \( G \) given as input.

To show that the output produced by the algorithm is correct, we need to show that it is feasible, i.e. a cut, and that it is a basal minimum cut. Note that since \( G \) and \( G' \) have the same set of vertices and edges, a cut in \( G' \) is also a cut in \( G \). Therefore, the output produced by the algorithm is feasible, as it is a cut of \( G \).

We now show that any minimum cut \((S, T)\) of \( G' \) is a basal minimum cut of \( G \); this will prove that the algorithm is correct. Let \((S', T')\) be any cut of \( G \). Note that the capacity of \((S', T')\) in \( G' \) is

\[
c'(S', T') = \sum_{u \in S', v \in T'} c'(u, v) = \sum_{u \in S', v \in T'} (c(u, v) + 1) = \sum_{u \in S', v \in T'} c(u, v) + n(S', T') = c(S', T') + n(S', T')
\]

where \( n(S, T) = \{|(u, v) | u \in S, v \in T \} \) is the number of edges across the cut \((S, T)\).

Similarly, \( c'(S, T) = c(S, T) + n(S, T) \). If we assume that all edge capacities are multiples of \( m \), then \( c(S, T) = km \) and \( c(S', T') = k'm \) for some integers \( k \) and \( k' \).

Since \((S, T)\) is a minimum cut in \( G' \), then \( c'(S, T) \leq c'(S', T') \). Therefore,

\[
c(S, T) + n(S, T) = km + n(S, T) \leq c(S', T') + n(S', T') = k'm + n(S', T'). \tag{1}
\]

Note that if \( k' < k \), then \( k' \leq k - 1 \) as \( k \) and \( k' \) are integer. Therefore, if \( k' < k \) then

\[
k'm + n(S', T') \leq (k - 1)m + n(S', T') = km + (n(S', T') - m) \leq km < km + n(S, T)
\]

where the second to last inequality follows from \( n(S', T') \leq m \) and the last inequality holds because \( n(S, T) \geq 1 \). Note that the above inequalities contradict (1), so it cannot be the case that \( k' < k \). This also means that \((S, T)\) must be a minimum cut of \( G \), as if it was not then there would be a cut \((S', T')\) with capacity \( c(S', T') = k'm < c(S, T) = km \), implying \( k' < k \).

Assume that \((S', T')\) is also a minimum cut of \( G \), so \( k' = k \). From (1) we get that \( n(S, T) \leq n(S', T') \), so \((S, T)\) is a basal minimum cut as it has no more edges than any other minimum cut of \( G \).

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

The construction of the flow network \( G' \) requires \( O(m + n) \) time as \( G' \) consists of \( n \) vertices and \( m \) edges. The algorithm of Edmonds and Karp requires \( O(m^2n) \) time.

Building the residual network needs \( O(m + n) \) time and finding a minimum cut needs \( O(m + n) \) time. Hence, the time complexity of the algorithm is \( O(m^2n) \).

4. Consider a group \( C = \{C_1, C_2, \ldots, C_{n_C}\} \) of companies. Each company \( C_i \) owns a set \( T_i \) of retail stores; sets \( T_i \) are disjoint. The total set of stores is \( S = \{S_1, S_2, \ldots, S_{n_S}\} \), where \( n_S = \sum_{i=1}^{n_C} |T_i| \). A group of warehouses \( W = \{W_1, W_2, \ldots, W_{n_W}\} \) can distribute computers to the stores. Warehouse \( W_i \) has \( q_i \) computers that it can distribute to the stores and a store \( S_i \) can receive up to \( r_i \) computers from warehouses. A warehouse \( W_i \) can only distribute computers to a set \( V_i \subseteq S \) of stores.
Each company $C_i$ has a total demand of $d_i$ computers, so the total number of computers that
the warehouses must distribute to the stores owned by $C_i$ must be exactly equal to $d_i$. The
problem is to determine whether the warehouses can distribute computers to the stores to
satisfy the demands of the companies without exceeding the capacities of the stores.

For example, suppose that $C = \{C_1, C_2\}$, $T_1 = \{S_1, S_2\}$, $T_2 = \{S_3, S_4, S_5\}$, $W = \{W_1, W_2\}$,
$V_1 = \{S_1, S_2, S_3, S_5\}$, and $V_2 = \{S_2, S_3, S_4\}$. Let $d_1 = 6$, $d_2 = 10$, $q_1 = 8$, $q_2 = 9$, $r_1 = 3$,
$r_2 = 4$, $r_3 = 5$, $r_4 = 5$, and $r_5 = 2$. A solution is this: $W_1$ sends 3 computers to $S_1$, 3 to $S_2$
and 2 to $S_5$, while $W_2$ sends 5 computers to $S_3$ and 3 to $S_4$. Note that the stores owned by $C_1$
receive in total 6 computers and the stores owned by $C_2$ receive 10 computers, as required.

(16 marks) Write an algorithm for solving this problem. If there is a way of delivering computers to stores to satisfy the demands of the companies, the algorithm must return the value $true$, otherwise it must return the value $false$.

**Algorithm** Delivery $(W, C, S, T, d, q, r)$

**Input:** Set $W$ of warehouses with inventories $q$, set $C$ of companies with demands $d$, set $S$
of stores with capacities $r$, set $V$ of stores serviced by each warehouse, and set $T$ of stores
owned by each company.

**Output:** True, if the warehouses can send computers to the stores to satisfy the companies
demands without exceeding their inventories and store capacities; false otherwise.

1. Build a flow network $G = (V, E)$, where $V = \{s, t\} \cup W \cup S \cup C$ and $E = \{(s, W_i) | W_i \in W, c(s, W_i) = q_i\} \cup \{(W_i, S_j) | W_i \in W, S_j \in V_i, c(W_i, S_j) = r_j\} \cup \{(S_j, C_k) | S_j \in T_k, C_k \in C, c(S_j, C_k) = r_j\} \cup \{(C_k, t) | C_k \in C, c(C_k, t) = d_k\}.$

2. Compute a maximum flow $f$ of $G$ using the algorithm of Edmonds and Karp.

3. **If** $f$ saturates all edges incident on $t$ **then** return $true$ else return $false$.

(16 marks) Prove that your algorithm correctly solves the above problem.

To prove that the algorithm is correct, first we need to show that it terminates. Since graph
$G$ has a finite number of vertices and edges, then it can be constructed in finite time. Since $G$
has finite size and all capacities are integer, the algorithm of Edmonds and Karp finishes in
finite time. Testing whether the flow $f$ saturates all edges incident on the sink also requires
a finite amount of time.

The algorithm returns the value $true$ or $false$, so the solution produced by it is feasible. To show
that the output of the algorithm is correct we need to show that flow $f$ saturates all edges incident on $t \iff$ there is a way of delivering computers from the warehouses to the stores that satisfies the demands of all companies.

- We show first that if there is a flow $f$ that saturates all edges incident on $t \Rightarrow$ there is a
way of delivering computers from the warehouses to the stores that satisfies the demands of all companies. For this we need to show how to construct a feasible way of delivering computers from warehouses to stores given that we know the value of the flow $f$ on every edge of $G$:
  - Warehouse $W_i \in W$ delivers $f(s, W_i) \leq q_i$ computers in total.
  - Warehouse $W_i$ delivers $f(W_i, S_j)$ computers to store $S_j$, for each $W_i \in W$ and
$f(W_i, S_j) > 0$. Note that by the way in which $G$ was constructed, node $W_i$ is
adjacent only to nodes for stores $S_j \in V_i$; hence, it is allowed for $W_i$ to deliver
computers to these stores.
The total number of computers delivered to store $S_j \in S$ is equal to $\sum_{i=1}^{n_W} f(W_i, S_j) = f(S_j, C_k) \leq c(S_j, C_k) = r_j$, where store $S_j$ belongs to company $C_k$; the first equality holds because $f$ is a feasible flow and so flow conservation is satisfied at each node $S_j$. Therefore, no store $S_j$ receives more computers than its capacity $r_j$.

Finally, the total number of computers received by the stores that belong to company $C_k \in C$ is $\sum_{S_j \in T_k} f(S_j, C_k) = f(C_k, t) = c(C_k, t) = d_k$. The first equality holds by flow conservation and the third one holds because $f$ saturates all the edges incident on $t$.

Hence, the above is a feasible way of delivering computers from the warehouses to the stores that satisfies the demands of all the companies.

- Finally, we show that if there is a way of delivering computers from the warehouses to the stores that satisfies the demands of all companies $\Rightarrow$ there is a flow $f$ that saturates all edges incident on $t$

We now show how to build the flow $f$ from a valid solution for the computer delivering problem:

- $f(s, W_i) =$ total number of computers delivered by warehouse $W_i$, for each $W_i \in W$. Since in a feasible solution warehouse $W_i$ delivers at most $q_i$ computers, then $f(s, W_i) \leq c(s, W_i) = q_i$.

- $f(W_i, S_j) =$ number of computers that warehouse $W_i$ sends to store $S_j$, for each $W_i \in W$ and $S_j \in T_i$. Since in a feasible solution warehouse $W_i$ sends no more than $r_j$ computers to store $S_j$, then $f(W_i, S_j) \leq c(W_i, S_j) = r_j$. Note that $f(s, W_i) = \sum_{S_j \in T_i} f(W_i, S_j)$ as $f(s, W_i)$ is the total number of computers delivered by warehouse $W_i$, so flow conservation is satisfied at each node $W_i$.

- $f(S_j, C_k) =$ total number of computers received by store $S_j$. Since in a feasible solution a store $S_j$ receives at most $r_j$ computers, then $f(S_j, C_k) \leq c(S_j, C_k) = r_j$. Furthermore, $\sum_{W_i \in W} f(W_i, S_j) = f(S_j, C_k)$, so flow conservation is satisfied at each node $S_j$.

- $f(C_k, t) =$ total number of computers received by the stores owned by company $C_k$. Since in a feasible solution the total number of computers received by the stores owned by company $C_k$ is $d_k$, then $d_k = f(C_k, t) = \sum_{S_j \in T_k} f(S_j, C_k)$ and so flow conservation is satisfied at each node $C_k$.

The above is a feasible flow as it satisfies the capacity constraints and flow conservation conditions. Furthermore, $f$ saturates all edges incident on $t$.

(8 marks) **Compute the time complexity of your algorithm.**

The flow network $G$ constructed by the algorithm has $n_W + n_S + n_C + 2$ nodes and at most $n_W + n_W n_S + n_S + n_C$ edges. Hence building the flow network requires $O((n_W + n_S + n_C) + (n_W + n_W n_S + n_S + n_C)) = O(n_W n_S)$ time as without loss of generality we can assume that $n_C \leq n_S$. The algorithm of Edmonds and Karp requires $O(n_W^2 n_S^2 (n_W + n_S))$ time. Finally, checking whether all edges incident on the sink are saturated requires $O(n_C)$ time. Hence, the time complexity of the algorithm is $O(n_W^2 n_S^2 (n_W + n_S))$.

(5 marks) **Optional question:** If the above problem is too easy for you, here is a slightly more challenging one. Assume that the computers are colored either pink or blue. Each warehouse $W_i$ stores $q_{pi}$ pink computers and $q_{bi}$ blue ones. Furthermore, each store $S_i$ has a limit $p_i$ on the maximum number of pink computers that it can receive. The problem is
to determine whether the warehouses can distribute computers to stores so as to satisfy the
demands of the companies subject to these additional constraints.