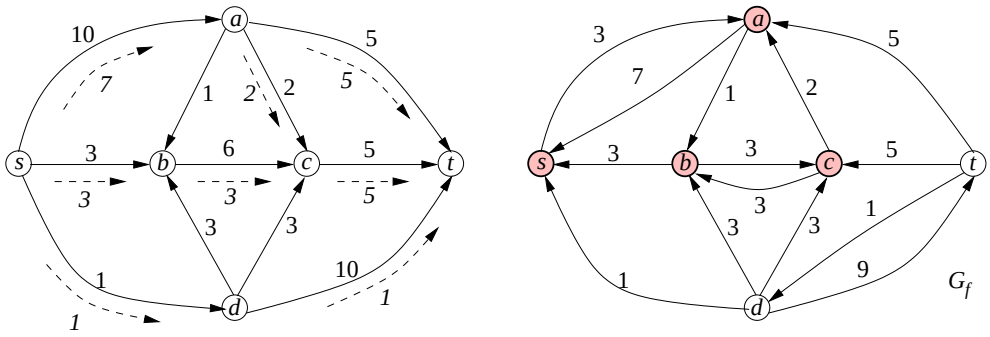
CS4445/9544 Analysis of Algorithms II

Solution for Assignment 1

1. Consider the following flow network.

   - *(10 marks) In the following network a minimum cut has capacity 10. Either prove that this statement is true, or show that it is false.*

     Using the algorithm of Ford and Fulkerson we compute the maximum flow f shown in Figure (a). The flow is indicated by the dashed arrows and the numbers below them. Since the flow has value 11, the claim that a minimum cut has capacity 10 is false as by the maximum flow minimum cut theorem a minimum cut in the given network has capacity 11.



(a) (b)

   - *(5 marks) Find a minimum cut (S, T) separating s from t. You need to indicate which vertices are in S and which are in T.*

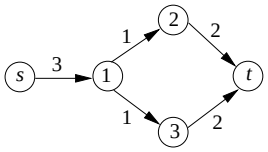
     Build the residual network G_f for the maximum flow in Figure (a) and mark all the vertices reachable from the source vertex (shaded nodes in Figure (b) above). These vertices are in the S side of a minimum cut. A minimum cut for the given flow network is then $(\{s, a, b, c\}, \{d, t\})$.

2. *(10 marks) Let $G = (V, E)$ be a flow network with source s, sink t and a positive integer capacity $c(u, v)$ on every edge $(u, v) \in E$. Let (S, T) be a minimum cut of G separating s from t. Let $k > 0$ be a positive integer.*

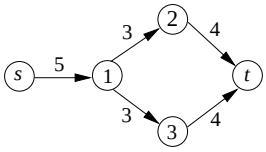
   Claim. *If the capacity of every edge $(u, v) \in E$ is increased by k, so the new capacity of edge (u, v) is $c'(u, v) = c(u, v) + k$, the cut (S, T) will still be a minimum cut with respect to the new capacities.*

   Determine whether the above statement is true or false. If it is true, give proof for it. If it is false, give an example proving the statement wrong.

   The claim is false. Consider the following flow network in Figure (c) with minimum cut $(\{s, 1\}, \{2, 3, t\})$. Let $k = 2$. If the capacity of every edge is increased by 2, as shown in Figure (d), then$(\{s, 1\}, \{2, 3, t\})$ is not a minimum cut anymore as $c(\{s\}, \{1, 2, 3, t\}) = 5 < c(\{s, 1\}, \{2, 3, t\}) = 6$.



(c)



(d)

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. Assume that we are given a maximum flow f of G, i.e. we know the value of the flow $f(u, v)$ on every edge (u, v) of G.*

   a. *(10 marks) Let the capacity of a single edge $(u, v) \in E$ be increased by an integer constant value $k > 0$. Write in pseudocode, similar to the one used in class, an $O(m + n)$-time algorithm to compute a maximum flow for the modified flow network.*

   Algorithm UpdateFlow $(G = (V, E), f, k)$
   Input: Flow network $G = (V, E)$, flow function f and integer constant value k; f is a maximum flow function for the network before the capacity of an edge was increased by k, G is the flow network after the capacity of an edge was increased by k.
   Output: Maximum flow function for G.

   Build the residual network G_f.
   while G_f has at least one augmenting path **do** {
   Find an augmenting path p in G_f.
   Compute the residual capacity $c_f(p)$ of p.
   For each edge $(u, v) \in p$ **do**
   $f(u, v) \leftarrow f(u, v) + c_f(p)$
   Build the new residual network G_f.
   }
   Return f

   b. *(10 marks) Prove that your algorithm computes a maximum flow.*

   To show that the algorithm computes a maximum flow for the modified network, we first need to prove that it terminates. Note that by increasing the capacity of a single edge by k the capacity of a minimum cut, and thus the value of a maximum flow, increases by at most k as well. Hence the algorithm terminates as each iteration of the while loop increases the value of the flow by at least 1, so after a most k iterations the algorithm must terminate. Note that building the residual network, computing a residual network and updating the value of the flow on the edges of an augmenting path all take finite time.
   The algorithm computes a maximum flow of the modified network because by the maximum flow minimum cut theorem, a flow function f is a maximum flow for a flow network $G = (V, E)$ if and only if G_f has no augmenting paths. The while loop of the above algorithm ends only when G_f has no augmenting paths and hence the flow returned is a maximum flow.

   c. *(10 marks) Show that the time complexity of your algorithm is $O(m + n)$.*

Building the residual network requires time $O(m+n)$. To determine whether G_f has augmenting paths and to find an augmenting path p we can use a depth first search algorithm, which runs on $O(m+n)$ time. Updating the value of the flow on the edges of an augmenting path requires $O(n)$ time as an augmenting path has at most $n-1$ edges.

As mentioned above, the while loop performs at most k iterations, hence the running time of the while loop is $O(k(m+n))$ which is $O(m+n)$ as k is constant. Finally, to return the flow function f we need to specify the value of the flow on every edge, which takes $O(m)$ time. The time complexity of the algorithm is then $O(m+n)$.

4. *A set $P = \{P_1, P_2, \ldots, P_k\}$ of k computer programs are to be executed by a group $C = \{C_1, C_2, \ldots, C_\ell\}$ of computers. The computer programs are classified into b types $T_1, T_2, \ldots, T_b$, depending on how many computer resources they require. Each computer program P_i has a unique type, $T_{\pi(i)}$.*

   Each computer C_i can execute a set of at most $a(i)$ programs, but it can process at most $a(i,j)$ programs of type T_j, for each $j = 1, 2, \ldots, b$. The problem is to assign all the programs in P to the computers so the the above conditions are satisfied, or to show that such an assignment is not possible.

   For example, assume that there are 7 programs $\{P_1, P_2, P_3, P_4, P_5, P_6, P_7\}$ of 3 types T_1, T_2, and T_3. Programs P_1 and P_2 are of type T_1, programs P_3, P_4, and P_5 are of type T_2 and the other ones are of type T_3. There are 3 computers, C_1, C_2, and C_3:

   - *C_1 can process at most 2 programs, of which 1 could be of type T_1 and at most 2 could be of type T_2, so $a(1) = 2$, $a(1,1) = 1$, $a(1,2) = 2$, and $a(1,3) = 0$.*
   - *For C_2 we have $a(2) = 3$, $a(2,1) = 1$, $a(2,2) = 1$, $a(2,3) = 1$.*
   - *For C_3 we have $a(3) = 2$, $a(3,1) = 2$, $a(3,2) = 0$, $a(3,3) = 1$.*

   A solution is to assign P_3 and P_4 to C_1, P_1, P_5, and P_6 to C_2 and P_2 and P_7 to C_3.

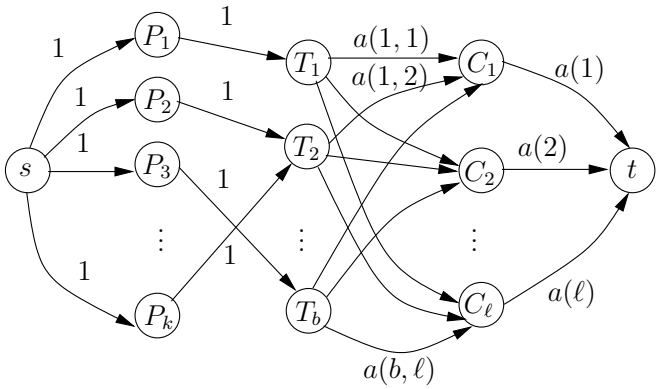
   i. *(20 marks) Write in pseudocode an algorithm for solving this problem. If there is a way of assigning programs to computers, the algorithm must return the value* true*, otherwise it must return the value* false*.*

   Algorithm ProgramAssignment (P, T, C, a)
   Input: Set P of computer programs, set $T = \{T_1, T_2, \ldots, T_b\}$ of program types, set C of computers with capacities a.
   Output: *True*, if all the programs can be assigned to the computers without exceeding the capacities $a(i)$ and $a(i,j)$; *false* otherwise.
   1. Build a flow network $G = (V, E)$, where $V = \{s, t\} \cup P \cup T \cup C$ and $E = \{(s, P_i) \mid P_i \in P, c(s, P_i) = 1\} \cup \{(P_i, T_{\pi(i)}) \mid P_i \in P, T_{\pi(i)} \in T, c(P_i, T_{\pi(i)}) = 1\} \cup \{(T_j, C_k) \mid T_j \in T, C_k \in C, c(T_j, C_k) = a(k,j)\} \cup \{(C_k, t) \mid C_k \in C, c(C_k, t) = a(k)\}$.
   2. Compute a maximum flow f of G using the algorithm of Ford and Fulkerson.
   3. **If** f saturates all edges incident on s **then return** *true* **else return** *false*

   The following figure illustrates the construction of the flow network.

   ii. *(15 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. Graph G has a finite number of vertices and edges, so it can be constructed in finite time. Since G



has finite size and all capacities are integer, the algorithm of Ford and Fulkerson finishes in finite time. Testing whether the flow f saturates all edges incident on the source also requires a finite amount of time. Hence, the algorithm always terminates.

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

there is a flow f that saturates all edges incident on $s \iff$ there is a feasible assignment A of programs to computers that does not exceed the capacities of the computers.

a. We show first that if there is a flow f that saturates all edges incident on $s \Rightarrow$ there is a feasible assignment A of programs to computers. For this we need to show how to construct a feasible assignment A given that we know the value of the flow f on every edge of G. A feasible assignment A assigns all programs to computers so that the capacities $a(i)$ and $a(i,j)$ of the computers are not exceeded. The assignment A is constructed as follows.
   For each program type T_i let $\mathcal{P}_i = \{P_j \mid f(P_j, T_i) = 1\}$, so $\mathcal{P}_i$ is the set of programs of type T_i that are to be assigned to computers. Programs in $\mathcal{P}_i$ are assigned to computers as follows:
   - The first $f(T_i, C_1)$ programs of $\mathcal{P}_i$ are assigned to C_1.
   - The next $f(T_i, C_2)$ programs of $\mathcal{P}_i$ are assigned to C_2.

     $\vdots$
   - The last $f(T_i, C_\ell)$ programs of $\mathcal{P}_i$ are assigned to C_ℓ.

   Note that since f satisfies the flow conservation property, the above process assigns all programs in $\mathcal{P}_i$ to computers.
   Since all edges incident on s are saturated, then $f(t, P_i) = f(P_i, T_{\pi(i)}) = 1$ for each program P_i and thus each program belongs to some set $\mathcal{P}_j$. Therefore, by the above argument each program is assigned to a computer.
   The number of programs of type T_j assigned to computer C_i is $f(T_j, C_i) \leq c(T_j, C_i) = a(i,j)$ and the total number of programs assigned to computer C_i is $\sum_{T_j \in T} f(T_j, C_i) = f(C_i, t) \leq a(i)$, the first equality holds because of the flow conservation property; hence the computer capacities are not exceeded and thus the above is a feasible assignment of programs to computers.

b. Finally, we show that if there is a feasible assignment A of all programs to computers $\Rightarrow$ there is a flow f that saturates all edges incident on s
   We now show how to build the flow f from the assignment A:

- $f(s, P_i) = 1$ for each $P_i \in P$.
- $f(P_i, T_{\pi(i)}) = 1$ for each $P_i \in P$.
- $f(T_j, C_i)$ = number of programs of type T_j that A assigns to computer $C_i \leq a(i, j)$, as A is a feasible assignment of programs to computers.
- $f(C_i, t)$ = total number of programs assigned to computer $C_i \leq a(i)$, as A is a feasible assignment of programs to computers.

The above flow function f satisfies the capacity constraints. We show now that it also satisfies the flow conservation property.

- $f(s, P_i) = f(P_i, T_{\pi(i)}) = 1$ for each program $P_i \in P$, so flow conservation is satisfied at each node P_i.
- $\sum_{P_j \in P} f(P_j, T_i) = \sum_{C_j \in C} f(T_i, C_j)$ for each program type T_i, as $\sum_{P_j \in P} f(P_j, T_i)$ is equal to the number of programs of type T_i and this value is equal to $\sum_{C_j \in C} f(T_i, C_j)$ as all programs of type T_i are assigned by A to computers. Hence, flow conservation is satisfied on all nodes $T_i \in T$.
- $\sum_{T_j \in T} f(T_j, C_i) = f(C_i, t)$ as $\sum_{T_j \in T} f(T_j, C_i)$ is the total number of programs assigned to C_i and this number is equal to $f(C_i, t)$. Hence flow conservation is satisfied on all nodes $C_i \in C$.

The above then is a feasible flow function f that saturates all edges incident on s.

(10 marks) Compute the time complexity of your algorithm.

The flow network G constructed by the algorithm has $k + b + \ell + 2$ nodes and at most $k + k + b\ell + \ell$ edges. Hence building the flow network requires $O(k + b\ell)$ time. The algorithm of Ford and Fulkerson requires $O(\#edges \times |f^*|) = O(k(k + b\ell))$ time as $|f^*| \leq k$ since the cut $(\{s\}, V \setminus \{s\})$ has capacity k. Finally, checking whether all edges incident on the source are saturated requires $O(k)$ time. Hence, the time complexity of the algorithm is $O(k(k + b\ell))$.