Introduction to Algorithms	**Solution Set 7**
CS 4820, Spring 2018	

(1) *(10 points) Suppose we are given an $n \times n$ matrix $A = (a_{ij})$ with non-negative entries, and we are interested in the question of whether it can be expressed as a sum of two matrices R, C such that:*

1. *The entries of R and C are non-negative.*
2. *The row sums of R are bounded above by 1.*
3. *The column sums of C are bounded above by 1.*

When there exist matrices R, C satisfying these three constraints, such that $R + C = A$, let us call the pair (R, C) a row-plus-column (RPC) decomposition of A.

Design a polynomial-time algorithm that takes a non-negative matrix A as input, and outputs either an RPC decomposition or a set of rows, S, and a set of columns, T, such that

$$\sum_{i \in S, j \in T} a_{ij} > |S| + |T|.$$

(In the latter case it follows easily that there is no RPC decomposition of A, since the row set S and column set T identify an $|S| \times |T|$ submatrix of A whose entries sum up to more than $|S| + |T|$, whereas the entries in the corresponding submatrices of R and C must have sums bounded above by $|S|$ and $|T|$, respectively.)

Solution. Construct a flow network with the following vertices and edges.

- source s, sink t
- vertex u_{ij} for each matrix entry
- vertex v_i for each row i
- vertex w_j for each column j
- for all i, j:
  - edge (s, u_{ij}) with capacity a_{ij}
  - edge (u_{ij}, v_i) with capacity $5n$
  - edge (u_{ij}, w_j) with capacity $5n$
- edge (v_i, t) with capacity 1 for all i
- edge (w_j, t) with capacity 1 for all j

Compute a maximum flow in this network. If the value of the maximum flow equals $\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}$ then a RPC decomposition of A is obtained by setting $R = (r_{ij})$ and $(C = c_{ij})$ to be the matrices defined by $r_{ij} = f(u_{ij}, v_i)$ and $c_{ij} = f(u_{ij}, w_j)$. Otherwise, compute the minimum cut (X, Y), and define $S = \{i \mid v_i \in X\}$ and $T = \{j \mid w_j \in X\}$.

The capacities in this flow network are not integers, so the maximum flow must be computed using an algorithm that is guaranteed to terminate even when capacities are not integers, such as the Edmonds-Karp algorithm (second heuristic — the one that always chooses the augmenting path with the fewest edges). The running time of that algorithm is $O(|E|^2|V|)$, and the flow network created by our reduction has $|E| = O(n^2)$ and $|V| = O(n^2)$, so the algorithm's running time is $O(n^6)$.

As for the correctness guarantee, first suppose the algorithm outputs a pair of matrices R and C, then it means that the maximum flow value was $\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}$ hence the maximum flow saturates every

edge (s, u_{ij}). Flow conservation now implies that

$$a_{ij} = f(u_{ij}, v_i) + f(u_{ij}, w_j) = r_{ij} + c_{ij}$$

for every i, j, hence $A = R + C$. Flow conservation also implies that the values $f(v_i, t)$ and $f(w_j, t)$ are the sum of entries in the i^{th} row of R and the j^{th} row of C, respectively. The capacity constraints of edges (v_i, t) and (w_j, t) thus ensure that the row sums of R and column sums of C satisfy the prescribed bounds, making $A = R + C$ a RPC decomposition.

For the remaining half of the correctness guarantee, suppose the algorithm outputs a pair of sets S, T. Since the cut $(V \setminus \{t\}, \{t\})$ has capacity $2n$, and all edges of the form (u_{ij}, v_i) or (u_{ij}, w_j) have capacity $5n$, we know that none of them can belong to the minimum cut (X, Y). Hence the edge set $E(X, Y)$ contains three types of edges: those from s to some $u_{ij} \in Y$, those from some $v_i \in X$ in t, and those from some $w_j \in Y$ to t. The combined capacity of the latter two types of edges is $|S| + |T|$, hence

$$c(X, Y) = \left(\sum_{u_{ij} \in Y} a_{ij} \right) + |S| + |T|. \tag{1}$$

Recalling again that $E(X, Y)$ contains no edges of the form (u_{ij}, v_i) or (u_{ij}, w_j), we can conclude that for every vertex $u_{ij} \in X$, the set X also contains v_i and w_j, implying $i \in S$ and $j \in T$. This justifies the inequality in the middle of the following line:

$$c(\{s\}, V \setminus \{s\}) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} = \left(\sum_{u_{ij} \in Y} a_{ij} \right) + \left(\sum_{u_{ij} \in X} a_{ij} \right) \leq \left(\sum_{u_{ij} \in Y} a_{ij} \right) + \left(\sum_{i \in S, j \in T} a_{ij} \right). \tag{2}$$

The fact that the algorithm chose to output two sets X, Y means that $c(\{s\}, V \setminus \{s\}) > c(X, Y)$. Combining this strict inequality with (1) and (2) implies $\sum_{i \in S, j \in T} a_{ij} > |S| + |T|$ as desired.

Remarks. Problems that reduce to max-flow are all about assigning stuff to capacitated resources without overflowing them. In this case the stuff consists of the entries of A, and the capacitated resources are the rows of R and columns of C. With that sentence, you can start to design the flow network that solves the problem.

The capacity of $5n$ on the edges in the middle layer is an arbitrary number chosen to be bigger than $2n$. Other capacities greater than $2n$ would work as well.

In my opinion, the harder half of the problem is the min-cut analysis which justifies that the sets S, T satisfy the necessary inequality in case an RPC decomposition is not found. Some hints relevant to this half of the problem are the following. (The second is a "stronger hint" than the first one.)

1. Whenever you encounter a problem where there's an "obvious obstruction" to the existence of a solution, and it turns out that the problem has a solution whenever there is no "obvious obstruction", you should suspect that the max-flow min-cut theorem is lurking in the background. In this case, a pair of sets S, T satisfying the inequality $\sum_{i \in S, j \in T} a_{ij} > |S| + |T|$ is an obvious obstruction to the existence of an RPC decomposition, for the reason given in the last (parenthetical) paragraph of the problem, so you should begin to suspect that this obstruction has some sort of interpretation in terms of min cuts.

2. The flow network has a structure very similar to the flow network that was used in class to find a maximum bipartite matching. After presenting that algorithm we devoted quite a bit of time to analyzing the implications of the max-flow min-cut theorem in the context of that construction, and we proved Hall's Marriage Theorem. That proof could constitute a template for the style of reasoning you need to use to surmount the second part of problem 1.

(2) *(10 points) At Ford-Fulkerson University there are many committees that need to be staffed with professors. The university has n professors organized into d departments; each professor belongs to only one department. There are m committees, and the following constraints must be satisfied when staffing the committees.*

1. *The required number of professors on committee k is specified by a positive integer r_k.*
2. *No professor is allowed to serve on more than c committees.*
3. *No committee is allowed to have more than one professor from the same department.*
4. *For each professor j, there is a list L_j of the committees on which he or she is qualified to serve. Professor j is not allowed to serve on committee k unless $k \in L_j$.*

Design a polynomial-time algorithm to determine whether it is possible to staff each committee without violating any of the constraints listed above. If it is possible to staff the committees, your algorithm should output an assignment of professors to committees that satisfies all of the constraints. The input to the problem is specified by the numbers $n, c, d, m, r_1, \ldots, r_m$, the lists $L_1, \ldots, L_n$, and an array of length n that identifies which department each professor belongs to.

Solution. Construct a flow network with the following vertices and edges.

- source s, sink t
- vertex u_i for each professor, i
- vertex v_{jk} for each committee, j, and department, k
- vertex w_j for each committee, j
- edge (s, u_i) with capacity c for each professor, i
- edge $(u_i, v_{j,k[i]})$ with capacity 1 for each professor, i and committee $j \in L_i$. Here $k[i]$ denotes the department that i belongs to.
- edge (v_{jk}, w_j) with capacity 1 for all committees, j, and departments, k
- edge (w_j, t) with capacity r_j for all committees, j.

Compute an integer-valued maximum flow in this network. If it saturates every edge (w_j, t) then output an assignment of professors to committees by specifying that professor i belongs to each committee j such that $f(u_i, v_{j,k[i]}) = 1$. If the maximum flow does not saturate every edge (w_j, t) then it is impossible to staff the committees.

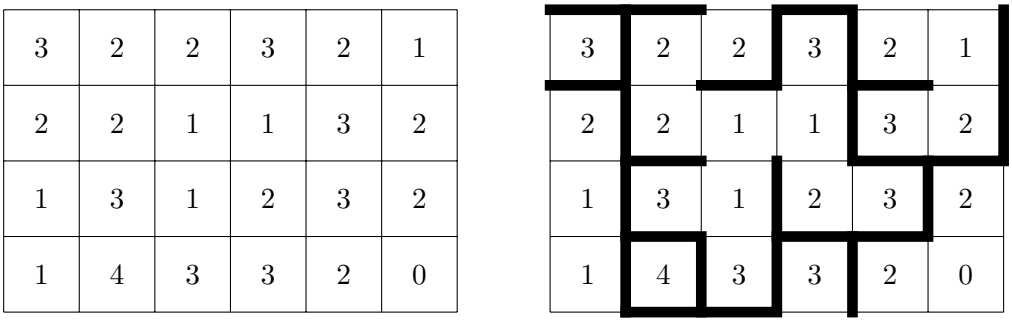
Assume that d (the number of departments) is less than or equal to n (the number of professors); otherwise modify the flow network construction above to omit the vertices v_{jk} whenever k is a department that has no professors. The graph has $O(nm)$ edges, and its minimum cut capacity is bounded above by dm, which is the capacity of the cut separating $\{s\} \cup \{u_i\} \cup \{v_{jk}\}$ from $\{w_j\} \cup t$. Hence the $O(|E| \cdot C)$ running time bound for Ford-Fulkerson in this case translates to $O(ndm^2)$. Students may come up with different running time bounds depending which flow algorithm they use and how they choose to bound the parameters which appear in the analysis of that algorithm's running time.

For the proof of correctness, first assume that the algorithm outputs an assignment of professors to committees. Then the flow conservation equation at node u_i says that the number of committees to which i is assigned equals $f(s, u_i)$, which is at most c because of the capacity constraint of edge (s, u_i). Using flow conservation, and the fact that f is an integer-valued flow which saturates each edge (w_j, t), along with the fact that each incoming edge of w_j has capacity 1, we may conclude that each node w_j has exactly r_j incoming edges with flow value 1. Each of those edges is of the form (v_{jk}, w_j), and again using flow conservation and integrality we can deduce that for each committe j there are exactly r_j distinct nodes of the form v_{jk} (corresponding to r_j distinct departments) such that each of the nodes

receives one unit of flow on an edge of the form (u_i, v_{jk}). That flow unit must come from a professor, i, who belongs to department k and has $j \in L_i$, since otherwise (u_i, v_{jk}) would not be an edge of the graph. Summarizing these considerations, we can deduce that the committee assignment derived from the flow satisfies the first, third, and fourth constraints specified in the problem statement.

Conversely, suppose that there exists an assignment of professors to committees satisfying the problem constraints. Then for every professor i assigned to committee j there is a path in the flow network of the form $s \to u_i \to v_{j,k[i]} \to w_j \to t$, and there is a flow of value 1 that sends one unit of flow along each edge of this path. Summing up those flows for each professor-to-committee assignment, we obtain a flow that satisfies the capacity constraints on every edge, and saturates every edge into t, because no professor is assigned to more than c committees, no two professors on committee j belong to the same department, and each committee j has r_j professors assigned to it.

(3) *(10 points)* Consider a puzzle in which you are given an n-by-n square grid, with an integer in the range $\{0, \ldots, 4\}$ written inside each grid cell. You are asked to select a subset F of the edges of the grid, such that for each grid cell the number written inside the cell matches the number of elements of F that belong to the cell's boundary. The following figure shows an example of a puzzle and a valid solution to the puzzle.



Design a polynomial-time algorithm that, when given such a puzzle, either outputs a valid solution or decides (correctly) that there is no valid solution.

Solution. Let n_i denote the number written in square i. Color the squares of the grid black and white, as in a chessboard, so that each black square is adjacent only to white squares and vice-versa. Let B denote the sum of n_i over all black squares i, and let W denote the sum of n_j over all white squares j. Construct a flow network with the following vertices and edges.

- source s, sink t
- vertex u_i for each black square i
- vertex v_j for each white square j
- vertex b representing the boundary of the grid
- edge (s, u_i) with capacity n_i for each black square i
- edge (v_j, t) with capacity n_j for each white square j
- edge (u_i, v_j) with capacity 1 for each pair of adjacent squares i, j
- edge (u_i, b) for each boundary black square i, with capacity equal to the number of edges of i that belong to the boundary.
- edge (b, v_j) for each boundary white square j, with capacity equal to the number of edges of j that belong to the boundary.
- edge (s, b) with capacity $W - B$ if $W > B$.
- edge (b, t) with capacity $B - W$ if $B > W$.

Compute a maximum flow in this network. If it does not saturate every edge leaving s and every edge entering t then there is no solution to the puzzle. Otherwise, output a set of edges F defined as follows: if i, j is a pair of adjacent squares such that $f(u_i, v_j) = 1$, put the edge dividing square i from square j into the set F. If i is a black square on the boundary, insert $f(u_i, b)$ boundary edges of square i into the set F. If j is a white square on the boundary, insert $f(b, v_j)$ boundary edges of square j into F.

The graph has $m = O(n^2)$ edges and $C = O(n^2)$ min-cut capacity, so running Ford-Fulkerson to compute a maximum flow takes $O(n^4)$. Other running time bounds are acceptable, for example $O(n^6)$ if using Edmonds-Karp and bounding the running time solely in terms of the number of edges and vertices of the flow network.

Note that the combined capacity of edges leaves s and the combined capacity of edges entering t are both equal to $\max\{B, W\}$. Hence, a flow satuates the edges leaving s if and only if it saturates the edges entering t. If both of these edge sets are saturated, flow conservation implies that each node u_i has exactly n_i units of flow on its outgoing edges; since these units of flow correspond exactly to the edges which are selected in boundary of square i, it follows that the edge set F satisfies the puzzle constraint represented by square i. Similarly, for a white square j, flow conservation implies that v_j has exactly n_j units of flow on its incoming edges. Again since these units of flow correspond exactly to the edges which are selected in boundary of square j, it follows that the edge set F satisfies the puzzle constraint represented by square j.

Conversely, suppose that the puzzle has a solution and let F be an edge set that solves the puzzle. We can convert F into a flow as follows. Saturate every edge leaving s or entering t. For an edge (u_i, v_j) let $f(u_i, v_i) = 1$ if F contains the edge between squares i and j, $f(u_i, v_j) = 0$ otherwise. For an edge (u_i, b) or (b, v_j) let the flow on that edge equal the number of boundary edges of the corresponding square that belong to F. This flow satisfies capacity constraints by construction. It satisfies flow conservation at each u_i or v_j because the corresponding square i or j has exactly n_i or n_j edges in F. The most subtle thing is to check flow conservation at b. Define a boundary edge of the grid to be black or white according to the color of the (unique) grid cell to which the boundary edge belongs. Let $B(F)$ and $W(F)$ denote the number of black boundary edges and white boundary edges, respectively, in the set F. Then the incoming flow minus the outgoing flow at node b is equal to $[B(F) - W(F)] - [B - W]$. The expression equals zero because the expression $B - W$ counts each edge of F with a coefficient of $+1$ if it is a black boundary edge, -1 if it is a white boundary edge, and 0 if it is an interior edge.