Solutions to End Semester Exam Questions

Q1. We can assume, without loss of generality, that $0 \leq a_{ij} < 1$ by subtracting $\lfloor a_{ij} \rfloor$ from a_{ij} and then adding it to b_{ij}. Construct a network with $m + n + 2$ vertices, $R_1, R_2, \ldots, R_m$, $C_1, C_2, \ldots, C_n$, source s and sink t. Add edges R_iC_j with capacity 1 for all $1 \leq i \leq m$ and $1 \leq j \leq n$. Add edges sR_i with a lower bound on flow $\left\lfloor \sum_{1 \leq j \leq n} a_{ij} \right\rfloor$ and an upper bound $\left\lceil \sum_{1 \leq j \leq n} a_{ij} \right\rceil$, for $1 \leq i \leq m$. Similarly, add edges C_jt with a lower bound $\left\lfloor \sum_{1 \leq i \leq m} a_{ij} \right\rfloor$ and upper bound $\left\lceil \sum_{1 \leq i \leq m} a_{ij} \right\rceil$, for $1 \leq j \leq n$. The matrix A gives a feasible fractional flow in this network, by setting $f(R_iC_j) = a_{ij}$, $f(sR_i) = \sum_{1 \leq j \leq n} a_{ij}$, and $f(C_jt) = \sum_{1 \leq i \leq m} a_{ij}$, for $1 \leq i \leq m$, $1 \leq j \leq n$. Because the lower and upper bounds on flow are integers, there exists a feasible flow f' with integer values. Then setting $B_{ij} = f'(R_iC_j)$ gives an integer matrix with the required properties.

Q2(a) The problem is to decide whether a given graph has a *maximal* independent set of size *at most* K. It is easy to show that the problem is in NP. Given a subset of vertices, check that no two are adjacent and every vertex not in the subset is adjacent to some vertex in the subset, and there are at most K vertices in the subset. To prove NP-Completeness, reduce 3-SAT to this problem. For every variable x_i in the instance of 3-SAT, construct a triangle with 3 vertices $x_i, \overline{x_i}, y_i$ in the graph. For every clause c_i with literals l_i, l_j, l_k, add a vertex c_i adjacent to the vertices corresponding to the literals l_i, l_j, l_k. The claim is that this graph has a maximal independent set of size at most n (number of variables in 3-SAT), iff the formula is satisfiable.

Suppose the formula is satisfiable. Take any satisfying assignment, pick the vertex x_i in the independent set if x_i is true in the assignment, else pick the vertex $\overline{x_i}$. This forms an independent set and it is maximal since any vertex c_i must be adjacent to the vertex corresponding to the true literal in the clause. Conversely, suppose there exists a maximal independent set of size at most n. Any maximal independent set must contain one of the vertices $x_i, \overline{x_i}, y_i$, since y_i is adjacent only to the two vertices $x_i, \overline{x_i}$. Therefore a maximal independent set of size at most n cannot contain any clause vertex. Now let x_i be true if the vertex x_i is in the maximal independent set, otherwise set x_i to false. This gives an assignment that satisfies all clauses.

Q2 (b) Essentially the same reduction shows that finding a c-approximation is NP-Hard. Instead of a single vertex for a clause, take $cn + 1$ copies of the clause vertex and connect all the copies to the literals in the clause. Now, if any one of these $cn + 1$ vertices is in a maximal independent set, then all of them must be, as they have the same neighbors. As before, if there is a satisfying assignment, there exists a maximal independent set of size at most n. On the other hand, if there is no satisfying assignment, any maximal independent set must contain a clause vertex and must have size at least $cn+1$. Thus even a c-approximate algorithm can determine whether there exists a satisfying assignment or not. Hence finding a c-approximation is NP-Hard.

Q3(a) For any $m \geq 1$, consider the following instance. Let $W = m^2$, $K = m$. There are $m(m-1)$ containers of weight 1, followed by an alternating sequence of $2m$ weights $m, m^2 - m + 1, m, m^2 - m + 1, \ldots, m, m^2 - m + 1$. The simple algorithm will first load $m - 1$ trucks with m containers of weight 1 in each, and will put the remaining weights in $2m$ different trucks. The optimal solution is obtained by putting all the m containers with weight m in one truck, and

loading m trucks with 1 container of weight $m^2 - m + 1$ and $m - 1$ containers of weight 1. Thus optimal needs only $m + 1$ trucks.

Q3(b) Suppose the simple algorithm requires $t + 1$ trucks. Then for each of the first t trucks, either it must contain K containers or the weight in the truck + weight of the next container must exceed W. This is the only reason the simple algorithm uses the next truck. Suppose there are t_1 of these trucks containing K containers and $t_2 = t - t_1$ for which the weight exceeds W if the next container is added to it. The algorithmic solution is $t_1 + t_2 + 1$. If $t_2 + 1 \leq 2t_1$, then the algorithmic solution is at most $3t_1$, and the optimal is at least t_1 since the number of containers is at least $t_1 K$. Suppose $t_2 + 1 > 2t_1$, that is $t_2 \geq 2t_1$. Then the algorithmic solution is at most $3t_2/2 + 1 < 3(t_2 + 1)/2$. Now considering the weights in the t_2 trucks, together with the weight of the next container, we get $2 \sum w_i > t_2 W$ which implies the optimal is at least $(\sum w_i)/W > t_2/2$. Thus optimal is at least $(t_2 + 1)/2$ and the algorithmic solution is at most 3 times the optimal.

Q4(a) The probability that a process gets access to a particular resource in a given time slot is $\frac{1}{m}\left[\frac{m-1}{m}\right]^{n-1}$. Since $m \geq 2$ this is at least $\frac{1}{m4^{(n-1)/m}}$. The probability that the process does not get access to a specific resource in a given time slot is at most $1 - \frac{1}{m4^{(n-1)/m}}$. The probability that the process does not get access to a specific resource in k time slots is at most $\left[1 - \frac{1}{m4^{(n-1)/m}}\right]^k$. Therefore if $k \geq m4^{(n-1)/m} \ln \frac{1}{\epsilon}$, this probability is at most ϵ. Note that this is for a specific resource. To ensure that this holds for all resources, the probability for each resource should be at most $\frac{\epsilon}{m}$. Also, to ensure that it holds for all processes, we require the probability for a given process and given resource to be at most $\frac{\epsilon}{mn}$. Therefore $N(n, m, \epsilon) \leq m^2 n 4^{(n-1)/m} \ln \frac{1}{\epsilon}$.

Q4(b) To ensure that $N(n, m, \epsilon)$ is bounded by a polynomial in $n, m, \ln \frac{1}{\epsilon}$, a process should not access a resource in each time slot. If we take the probability of accessing a resource to be $\min(\frac{m}{n}, 1)$ and choosing a resource randomly with equal probability, the probability of accessing a specific resource in a given time slot is at least $\frac{1}{en}$, as in the problem with a single resource.The same analysis then gives a polynomial bound.