**WARNING:** if you haven’t at least tried hard to solve the practice problems before reading these solutions, you are missing the point. If you can’t make *any* progress, talk to me or to the TAs before reading these solutions. Otherwise, you should come up with a solution of your own that you can compare to the one shown here.

1. Consider the following LP-rounding algorithm for $P$. First, compute the LP optimum $\bar{X}$. Then, form an integer solution $\hat{X}$ as follows: for every $i$, if $\bar{x}_i > 0$, set $\hat{x}_i = 1$. Return the solution $\hat{X}$.

   We first note that $\hat{X}$ is feasible. Indeed, every constraint in the problem is a lower bound, so if the constraint is satisfied in the (feasible) LP optimum $\bar{X}$, then increasing the values of the variables cannot make it unsatisfied.

   To show the claimed ratio, let $S(X) = \sum_i c_i \cdot x_i$ be the value of any solution to $P$. Let $X^*$ be an optimal integer-valued solution to $P$. We know that $S(\bar{X}) \leq S(X^*)$, since the LP optimum is no worse than the ILP optimum. To obtain an upper bound, note that, because no non-zero $\bar{x}_i$ is less than $1/k$, our rounding procedure increases the value of all such $\bar{x}_i$ by a factor of at most $k$. Conclude that

   $S(\hat{X}) = \sum_i c_i \cdot \hat{x}_i$
   $\leq \sum_{\bar{x}_i > 0} c_i \cdot k \cdot \bar{x}_i$
   $= k \sum_{\bar{x}_i > 0} c_i \cdot \bar{x}_i$
   $= kS(\bar{X})$
   $\leq kS(X^*)$.

2. **(a)** The generalization of the ILP for $m$ machines includes the following constraints:
   - For each job $i$, $\sum_j x_{ij} = 1$.
   - For each machine $j$, $\sum_i p_{ij} x_{ij} \leq t$.

   There are now $nm + 1$ variables (t plus indicators for each job, for each machine), $n + m$ “nontrivial” constraints, and $nm + 1$ constraints of the form $v \geq 0$. Hence, there exists an optimal solution to the LP relaxation of this ILP that makes at least

   $nm + 1 - (n + m)$

   variables zero. Now $t$ cannot be zero, so conclude that at most $n + m - 1$ of the $x_{ij}$s can be nonzero.
Now, how many jobs can be split across machines? Let $a$ and $b$ be the number of split and unsplit jobs, respectively. We have that $a + b = n$, since there are $n$ total jobs. Each split job makes at least $2$ $x_{ij}$s nonzero, while each unsplit job makes exactly one variable nonzero, so we have that $2a + b \leq n + m - 1$. Subtracting the equation from the inequality gives us $a \leq m - 1$; that is, at most $m - 1$ jobs can be split.

(b) We generalize our two-machine algorithm as follows. First, set up the general ILP with constraints as described above and objective $t$. Second, solve the LP relaxation to obtain an LP solution $X$. Third, we construct a fully integral solution as follows:

- Let $S$ be the set of at most $m - 1$ jobs split by the LP solution, and let $N$ be the remaining jobs.
- Let $X_N$ be the (integral) schedule given by $X$ to the jobs in $N$.
- Consider all possible ways of scheduling the jobs of $S$ on $m$, and let $X_S$ be the optimal solution found.
- Finally, place the jobs of $N$ and $S$ on the machines specified by $X_S$ and $X_N$, respectively, and return the joint schedule $\hat{X}$.

We claim that this algorithm is a 2-approximation. Let $S(X)$ be the max length of schedule $X$, and let $X^*$ be an optimal schedule. Because the schedule $X_N$ is a subset of $X$, we have

$$S(X_N) \leq S(X) \leq S(X^*).$$

Moreover, because $S$ is a subset of the input set of jobs, we have

$$S(X_S) \leq S(X^*).$$

Conclude that

$$S(\hat{X}) \leq S(X_S) + S(X_N) \leq S(X^*) + S(X^*) = 2S(X^*).$$

The cost of the algorithm includes $O(mn)$ to set up the ILP, the cost of solving the LP, the $O(nm)$ cost of identifying the sets $N$ and $S$, and the cost of trying all possible schedules for $S$. To bound the number of such schedules, note that the number of ways to assign $m - 1$ jobs to $m$ machines is $m^{m-1}$, and that we can surely determine the cost of each such schedule in time $O(m)$. Conclude that the cost of the algorithm is $O(mn + m^m)$, which is fine because $m$ is a constant!

Note: there is actually a 2-approximation algorithm for any number of machines that is polynomial in both $n$ and $m$. The algorithm relies on a faster way to find a good assignment of the $m - 1$ split jobs to processors. For details, see Chapter 17 of Vazirani’s book, or Chapter 1 of Hochbaum’s book.

3. (a) Let $S \subseteq S'$ and $x \notin S'$ be given. First, it is clear that $g$ is monotone, since if $f(S) \leq f(S')$, then this inequality continues to hold if we replace any value $> c$ on either side by $c$.

To see submodularity, observe first that by our monotonicity constraints, one of the following two total orders holds:

$$f(S) \leq f(S \cup \{x\}) \leq f(S') \leq f(S' \cup \{x\}),$$

or

$$f(S) \leq f(S') \leq f(S \cup \{x\}) \leq f(S' \cup \{x\}).$$
Now suppose we again replace any value \( c > c \) by \( c \). Under the first total order, if \( c \leq f(S') \), then the difference \( g(S' \cup \{x\}) - g(S') \) is 0, so submodularity must hold for \( g \). Otherwise, \( g(S' \cup \{x\}) - g(S') \) is still \( \leq f(S') - f(S' \cup \{x\}) \), which is \( < f(S) - f(S \cup \{x\}) \) by submodularity of \( f \); hence \( g \) is still submodular.

A similar argument can be made for the second total order, except that if \( c < f(S \cup \{x\}) \), both the differences \( f(S) - f(S \cup \{x\}) \) and \( \leq f(S') - f(S' \cup \{x\}) \) will be reduced – but the latter will be reduced by more, and hence submodularity of \( g \) follows from that of \( f \).

(b) Let \( S \subseteq S' \) and \( x \not\in S' \) be given. Let \( V = U - S - \{x\} \), and let \( V' = U - S' - \{x\} \). Observe that \( V' \subseteq V \), and hence by submodularity of \( f \),

\[
f(V \cup \{x\}) - f(V) \leq f(V' \cup \{x\}) - f(V').
\]

Expanding the definitions of \( V \) and \( V' \), we have

\[
f(U - S - \{x\} \cup \{x\}) - f(U - S - \{x\}) \leq f(U - S' - \{x\} \cup \{x\}) - f(U - S' - \{x\}),
\]

or, to simplify slightly,

\[
f(U - S) - f(U - S - \{x\}) \leq f(U - S') - f(U - S' - \{x\}).
\]

Finally, substituting \( \overline{f}(S) = f(U - S) \) gives us

\[
\overline{f}(S) - \overline{f}(S \cup \{x\}) \leq \overline{f}(S') - \overline{f}(S' \cup \{x\}).
\]

Multiplying each side by \(-1\) yields exactly the definition of submodularity for \( G \).