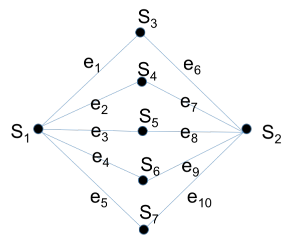
**Solutions:**

**(1)(a)**




**(b)**

$$\begin{aligned}
\max \quad & Z = y_1 + y_2 + \cdots + y_{10} \\
s.t. \quad & y_1 + y_2 + y_3 + y_4 + y_5 \leqslant 2 \\
& y_6 + y_7 + y_8 + y_9 + y_{10} \leqslant 2 \\
& y_1 + y_6 \leqslant 1 \\
& y_2 + y_7 \leqslant 1 \\
& y_3 + y_8 \leqslant 1 \\
& y_4 + y_9 \leqslant 1 \\
& y_5 + y_{10} \leqslant 1 \\
& y_1, y_2, \ldots, y_{10} \geqslant 0.
\end{aligned}$$

**(c)** Initially: $y_i = 0$ for all $i$. Then the y-variables are increased one by one. The solution that we get depends on the order that we take. Here, we take order $y_1, y_2, ..., y_8$. So start with increasing $y_1$. When $y_1 = 1$, the constraint $y_1 + y_6 \leq 1$ becomes tight. So set $y_1 = 1$ and add the corresponding set $S_3$ to the solution. Next, increase $y_2$. When $y_2 = 1$ then the constraints for $S_4$ and $S_1$ becomes tight (since now $y_1 + y_2 = 2$.) So add $S_1$ and $S_4$ to the solution. The next variable that we can still increase is $y_8$. Set $y_8 = 1$ and add $S_5$ to the solution. Then, set $y_9 = 1$ and add $S_6$ and $S_2$ to the solution. The solution obtained is $\{S_1, S_2, S_3, S_4, S_5, S_6\}$ and the value is $w_1 + w_2 + w_3 + w_4 + w_5 + w_6 = 8$.

**(2)(a)** (See book or lecture notes.) Let $k$ be the job that completes last and let $s_k$ be its start time. Then all machines are busy before time $s_k$. So $s_k \leq \sum_j p_j/m \leq \text{OPT}$. Also, $p_k \leq \text{OPT}$. The length of the schedule is

$$s_k + p_k \leq \sum_j p_j/m + p_k \leq 2\text{OPT}.$$

**(b)** First we prove the hint. Let $k$ be job that completes last. By definition of the algorithm, any job either starts on a machine directly after another job finishes on that machine or it starts at its release time. So idle time between $r_k$ and $s_k$ can only occur when som job $j < k$ starts at its release time $r_j > r_k$. But this cannot happen since $r_j \leq r_k$ for all $j \leq k$.

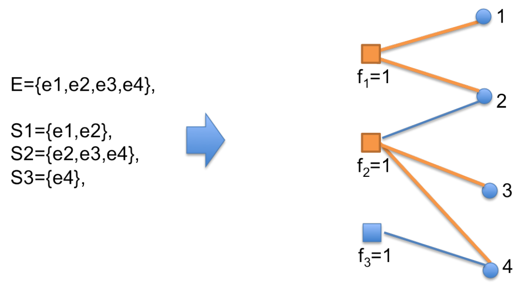
Now we use the hint.Since all machines are busy between $r_k$ and $s_k$ we have $s_k - r_k \leq \sum_j p_j/m \leq \text{OPT}$. Also, $r_k + p_k \leq \text{OPT}$ since job $k$ cannot complete before this time. The length of the schedule is

$$s_k + p_k = (s_k - r_k) + (r_k + p_k) \leq \text{OPT} + \text{OPT} = 2\text{OPT}.$$

**(3) (a)** The DP is a simplified version of the DP for knapsack. Let $A_j$ be the set of all $b$ such that there is a subset of the first $j$ items that add up to $b$. Then, $A_1 = \{0, s_1\}$. And for $j \geqslant 2$ we find $A_j$ as follows:
$A_j \leftarrow A_{j-1}$ and for any $b \in A_{j-1}$, add $b + s_j$ to $A_j$ if $b + s_j \leqslant B$.
The optimal value is given by the largest value in $A_n$

**(b)** Say that an item $i$ is large if $s_i \geqslant \varepsilon B$. There are at most $1/\varepsilon$ large items in the optimal solution. This gives $n^{1/\varepsilon}$ possible combinations of large items. For each one, add the small jobs in a greedy way. In one of these rounds, the algorithm chooses the same large items as OPT. If all small items fit then $\text{ALG} = \text{OPT}$. Otherwise, $\text{ALG} \geqslant B - \varepsilon B = (1-\varepsilon)B \geqslant (1-\varepsilon)\text{OPT}$.
(NB. Note that no rounding of values is needed here. We just try all combinations of large items.)

**(4)** See the figure. Given an instance $E, S_1, \ldots, S_m$ of the (unweighted) set cover problem we




model it as a UFL problem as follows. For each set $S_j$ we define one facility with opening cost $f_j = 1$. For each element $e_i \in E$ we define one client $i$. The cost for connecting client $i$ with facility $j$ is taken 0 if $e_i \in S_j$ and infinite otherwise (or some very large number). If there is a set cover of value $k$, that means all elements can be covered with $k$ sets, then there is a solution to the defined instance of the UFL problem with value $k$ as well: simply open the facilities that correspond to the sets in the set cover. The converse is also true: if there is a solution to the UFL problem of value $k$ then there is a set cover of size $k$. Hence we showed that the optimal value for the set cover instance is $k$ if and only if the optimal value of the UFL instance is $k$.

So any $f(|D|)$-approximation algorithm for facility location (without triangle inequality) implies an $f(n)$-approximation algorithm for set cover. Since set cover cannot be approximated better than $O(\log n)$, facility location without triangle inequality cannot be approximated better than $O(\log |D|)$.

**(5)**
(a)Assign uniformly at random to the sets. The probability that $e$ is in the cut is exactly $(k-1)/k$. So the expected total weight of the edges in teh cut is at least $(k-1)/k$ time the total weight of the edges, which is at least $(k-1)/k$ times the optimal value.

(b) Use this approach: First, solve the VP that was used for the 3-coloring problem. This gives a set of vectors $v_1, \ldots, v_n$. We know that the optimal value is at most $-0.5$. That means, for any edge $(i, j)$, the angle between the two vectors $v_i$ and $v_j$ is at least $2\pi/3$. Now take two random hyperplanes. This gives a partition in 4 sets. The probability that an edge has endpoints in different sets is at least $1 - (1/3)^2 = 8/9$. Hence, the expected total weight of the cut is at least $8/9$ times the total weight of the edges, which is at least $8/9$ times OPT.

(c) Let $k = 2^q$. Take $q$ hyperplanes. Then for any edges of the graph, the probability that it is not in the cut is at most $(1/3)^q = ((1/2)^{(\log_2 3)q} = k^{-\log_2 3}$. Hence, the expected total weight of the cut is at least $k^{-\log_2 3}$ times the total weight of the edges.

**(6)** This is just a set cover problem and we can use LP-rounding. Take a variable $y_i$ for each boolean variable $x_i$. Then the ILP becomes

$$\begin{array}{lll} \min & Z = \sum y_i & \\ s.t. & \sum\limits_{i \in C_j} y_i \geqslant 1 & \text{for each clause } C_j \\ & y_i \in \{0,1\} & \text{for each } i \end{array}$$

Now solve the LP-relaxation and round to 1 if $y_i \geqslant 1/3$ and round to zero otherwise. Then the solution is feasible since at least one of the $y_i$'s is rounded to 1 in each constraint. The total value is increased by at most a factor 3.