Task 1

a) Let x_S be a binary variable that is 1 if and only if the set S with associated weight w_S is selected.

$$\text{minimize} \sum_{S \in \mathscr{S}} w_S \cdot x_S \tag{1}$$

$$\text{when} \sum_{S : e \in S} x_S \geqslant 1 \quad e \in \overset{\text{Set of elements}}{\mathrm{E}} \tag{2}$$

$$x_S \in \{0,1\} \quad S \in \underset{\text{Set of subsets}}{\mathscr{S}} \tag{3}$$

b) The dual of the LP relaxation of the integer program in a) becomes

$$\text{maximize} \sum_{e \in E} y_e \tag{4}$$

$$\text{when} \sum_{e \in S} y_e \leqslant w_S \quad S \in \overset{\text{Set of subsets}}{\mathscr{S}} \tag{5}$$

$$y_e \geqslant 0 \quad e \in \underset{\text{Set of elements}}{\mathrm{E}} \tag{6}$$

As we minimize the integer program from a), it is not necessary to include the restriction $x_S \leqslant 1$ in the LP relaxation, which simplifies the dual.

c) Here we follow the standard formula [a] for complementary slackness and set up the equations based on these:

$$\overset{\mathbf{x}}{x_S} \left(\overset{\mathbf{c}}{w_S} - \overset{\mathbf{yA}}{\sum_{e \in S} y_e} \right) = 0 \quad S \in \mathscr{S} \tag{7}$$

$$\underset{\mathbf{y}}{y_e} \left(\underset{\mathbf{b}}{1} - \underset{\mathbf{Ax}}{\sum_{S : e \in S} x_S} \right) = 0 \quad e \in E \tag{8}$$

d) Introduces the approximation factors α and β when x_S and y_e are not 0.

$$x_S = 0 \quad \vee \quad \alpha \cdot w_S \leqslant \sum_{e \in S} y_e \leqslant w_S \quad S \in \mathscr{S} \tag{9}$$

$$y_e = 0 \quad \vee \quad 1 \leqslant \sum_{S : e \in S} x_S \leqslant \beta \quad e \in E \tag{10}$$

e) For example, we can create the following primal-dual algorithm.

```
1 Initialize: $x_S = 0$ for $S \in \mathscr{S}$, and $y_e = 0$ for $e \in E$.
2 Choose an uncovered element $e$, i.e., $\sum_{S:e\in S} x_S < 1$ and increase $y_e$ until you get a tight
  dual constraint for at least one S.
3 Set $x_S := 1$ for all S that received tight restrictions in the previous step.
4 If a set cover is found exit, otherwise go to 2.
```

f) We can see that in the cases where $x_S \neq 0$ in our solution, we know that $\sum_{e\in S} y_e = w_S$. Otherwise, we would not set $x_S = 1$ in step 3 and x_S would still be 0. Thus, the approximate complementary slackness condition applies if we set $\alpha = 1$.

Let f be the maximal number of sets an element appears in. Since the number of sets an element $e \in E$ appears in is maximal f, and $x_S \in \{0, 1\}$ we know that $\beta = f$.

g) We have an f-approximation algorithm:

$$\sum_{S\in\mathscr{S}} w_S \cdot x_S \underset{\text{Follows from (9)}}{\leqslant} \frac{1}{\alpha}(\sum_{S\in\mathscr{S}}(\sum_{e\in S} y_e)x_S) = \frac{1}{\alpha}(\sum_{e\in E}(\sum_{S:e\in S} x_S)y_e) \underset{\text{Follows from (10)}}{\leqslant} \frac{\beta}{\alpha}\sum_{e\in E} y_e = f\sum_{e\in E} y_e \tag{11}$$

[a]See e.g. here for a review.

Task 2

a) The naive greedy algorithm does not give a constant approximation factor. An example of this is having two items, one with value 1 and weight 1 and one with value $x > 1$ and weight W. The algorithm may end up selecting the item with value 1, even though the optimal solution is to select the item with weight W. This gives an approximation factor of x. Since we can choose x as large as we want, the algorithm cannot have a constant approximation factor.

b) In the fractional knapsack problem, we use the method of sorting the items according to their value to weight ratio. And we also bring a part of the next item. This gives an optimal solution for the fractional knapsack problem, FOPT, which incidentally has an optimal solution at least as good as the binary knapsack problem (all solutions to the binary knapsack problem are solutions to the fractional knapsack problem).

This means, if the algorithm picks the k first objects after sorting, then taking the $k+1$-th object will not be allowed (due to weight), but will lead to a value that is higher than the optimal the value, OPT. Therefore,

$$\sum_{i=1}^{k+1} v_i > \text{FOPT} \geqslant \text{OPT} \geqslant \sum_{i=1}^{k} v_i$$

That means that either the first k items are worth at least OPT/2 or item $k+1$ is worth at least OPT/2. In both cases, we are guaranteed that the algorithm returns a set consisting of items worth at least OPT/2 and thus we have an approximation factor of 1/2.

Task 3

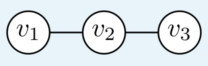
a) For this we can use a greedy algorithm that is quite similar to the one in the compendium for the k-center problem. In each step, we select the client that is furthest away from the current subset of suppliers and add the supplier closest to this client.

We have to show that this gives an approximation factor of 3. Let r be the largest distance from a client to the nearest supplier in an optimal solution. Whenever we pick a client, i, and add the closest supplier to our set, we know that the supplier that was picked is at most r away from this client. Everyone who was covered by the same supplier as client i in the optimal solution is at most at a distance $2r$ from client i. Thus these will be at a distance of at most $3r$ from the supplier that was selected for client i in the optimal solution. Therefore, each step will cover a set of clients with a maximal distance $3r$.

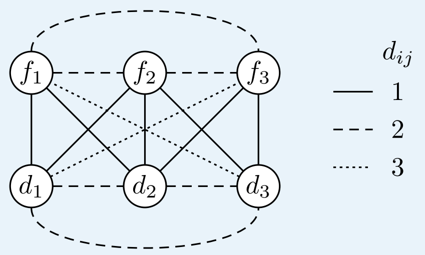
Following the logic above, we will never pick two clients who shared the same supplier in the optimal solution, until the distance from all clients to their nearest supplier is at most $3r$. Since we had k suppliers in the optimal solution, this means that after k such selections, we have covered all the clients with a supplier that is at most a distance $3r$ away. Thus, we have an approximation factor of 3.

b) To show this, we shall reduce from the NP-complete problem of determining whether there is a dominating set consisting of k vertices in a graph (*the dominating set problem*). A dominating set is a subset of vertices such that every vertex in the graph is either in the set or has at least one neighbor that is. The way we reduce is similar to that of the k-center problem. What we want to achieve is that an optimal solution has largest distance of 1 if there is a dominant set of size k and otherwise largest distance of 3. In such a situation, an approximation factor of $\alpha < 3$ will mean that we can distinguish between the situations and thus solve the NP-complete problem in polynomial time.

Given a graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, we want to reduce to the k-supplier problem in polynomial time. We begin by creating a supplier, f_v, and a client, d_v for each vertex v in the graph. These get a distance of 1 from each other. We also set the distance from each supplier f_v to each client d_u to be 1 if $(v, u) \in \mathrm{E}$ and 3 if $(v, u) \notin \mathrm{E}$. That is, clients and suppliers have a distance of 1 if they are neighbors/the same vertex in the graph and 3 otherwise. To satisfy the triangle inequality, we need to set the distance between each pair of clients to be 2 and the distance between each pair of suppliers to the same. We can see that if it is possible to pick a set of suppliers that gives distance 1, then we are guaranteed that this is a dominating set in the graph. This is because all clients are linked to a supplier that either represents their node or a neighboring node (a dominating set). Otherwise, we see that the distance must be 3. The figure below shows an example of the reduction.



a) Original graph



b) Instance of the k-supplier problem

Task 4

a) We can show that (E, I) forms an independence system by showing that:

1. $\emptyset \in \mathrm{I}$;
2. if $\mathrm{A} \subset \mathrm{B}$ and $\mathrm{B} \in \mathrm{I}$, then $\mathrm{A} \in \mathrm{I}$.

The first point is trivial to show, since a graph without edges is always a forest.

For the second point, we know that if we remove an edge from a forest, we remove an edge from a tree. If we remove an edge from a tree, we end up with two trees. Thus we can see that if we remove one or more edges from a forest, we end up with a forest with more trees. Thus A must be a forest if B is a forest and we have point two.

b) All maximal bases of (E, I) will be a spanning forest, and these must naturally have the same cardinality. One of the properties that defines a matroid (p. K152) is precisely this, that all the maximal bases have the same cardinality, and we thus have a matroid.

c) E is just all the edges in the graph, that is, $\mathrm{E} = \{(a,b),(a,c),(b,c),(c,d)\}$. I, on the other hand, are all acyclic subsets of edges in the graph. This gives us the following $\mathrm{I} = \{\emptyset, \{(a,b)\}, \{(a,c)\}, \{(b,c)\}, \{(c,d)\}, \{(a,b),(b,c)\}, \{(a,b),(a,c)\}, \{(a,b),(c,d)\}, \{(a,c),(b,c)\}, \{(a,c),(c,d)\}, \{(b,c),(c,d)\}, \{(a,b),(b,c),(c,d)\}, \{(a,b),(a,c),(c,d)\}, \{(a,c),(b,c),(c,d)\}\}$

d) In each iteration, we pick out the edge with the highest weight, which ensures $\mathrm{T}_n \in \mathrm{I}$. We thus obtain the following:

$$\begin{aligned}
\mathrm{T}_0 &= \emptyset \\
\mathrm{T}_1 &= \{(c,d)\} \\
\mathrm{T}_2 &= \{(c,d),(b,c)\} \\
\mathrm{T}_3 &= \{(c,d),(b,c),(a,c)\}
\end{aligned}$$

Task 5

Yes, we still have an independence system where all maximal bases have the same cardinality. Given that the original matroid has maximal bases with cardinality less than or equal to k, the matroid will not change after the constraint is added. If the original matroid had maximal bases with cardinality greater than k, maximal bases will now be of size k.

Task 6

We have that $\mathrm{APX} \geqslant \frac{1}{4}\mathrm{OPT}$ since we have the intersection of four matroids.

- The graphic matroid, which prevents cycles.
- The tail matroid, which prevents us from leaving the same place twice.
- The head matroid, which prevents us from arriving at the same place twice.
- The «terrain matroid», which prevents us from visiting the same terrain twice.

The easiest way to show that the terrain matroid is a matroid is by seeing that we have an independence system where maximal bases have cardinality equal to the number of different terrains.

Task 7

Firstly, we have an independence system, since any subset of a set of edges that has at most one cycle will also have at most one cycle.

We choose to show that we have a matroid using the exchange property: given $|\mathrm{J}| = |\mathrm{K}| + 1$ there must exist a $x \in \mathrm{J}/\mathrm{K}$ such that $\{x\} \cup \mathrm{K} \in \mathrm{I}$, where I is the family of all subgraphs with at most one cycle. We build up a basis by adding an edge. When an edge is added to the basis, either the number of connected components will be reduced by one, or a cycle will be introduced. Thus, K must have as many or more connected components than J. In the case where K has more connected components than J, J must have an edge that connects two connected components in K. If J and K have the same number of connected components, J must have a cycle, and K cannot have a cycle. Then the case must either be that you have exactly the same connected components, in which case you can choose x so that you introduce the same cycle in K that exists in J, otherwise there has to exists as an edge in J, like earlier, which binds together two connected components in K. Note that connecting two connected components does not form any cycles.

If we allow two cycles, then we no longer have a matroid. An example of this is the graph below, where all the edges in the graph are shown. There we have two *maximal* bases with different cardinalities. Both share the solid lines, but also contain the dotted lines of each color. Note that the basis containing the red dashed line has only one cycle. However, we cannot add any of the gray dashed edges, as this would result in three cycles.