1 - Exam preparation

We consider the following problem :

Problem 1. *Taking the right set of courses for each semester is always challenging – in this problem you will design an algorithm for this task.*
Suppose that there are n courses available for the next semester and the exam season for the next semester starts at day 1. Also, for any $j \in 1, 2, ..., n$, the j'th course has c_j credits and its exam is on day d_j. In order to pass a course you need to allocate exactly one day to prepare for its exam. You want to obtain the maximum number of credits possible in the next semester. Design an efficient algorithm that finds a set of optimal courses (with the maximum credits in total) that you can pass and prove the correctness of your algorithm, assuming the following:

1. *You can only prepare for the exams after the exam season starts (it starts on day 1).*
2. *On each day, you can only study one course the whole day (no multitasking).*
3. *The exams take place at the end of each day. Assume that they do not take any time, i.e., if you have any number of exams on any day, you can still study on that day.*
4. *If you do not study for a course before its exam, you are guaranteed to fail that course. For example, to pass an exam that is on day d_j, you need to study for that exam on a day i, where $1 \leq i \leq d_j$.*
5. *It is possible to have more than one exam on a day.*

Proposed Solution

We reduce ourselves to applying the greedy algorithm on a well chosen matroid. From now on, we will refer to a specific course by a number in $[n] = \{1, \dots, n\}$. Also let us suppose that they are ordered increasingly by the date of their exams, i.e. we have

$$d_1 \leq \dots \leq d_n.$$

For a subset of courses $X \subseteq [n]$, we say that X is *feasible* if X is a set of course that a student can prepare for according to the rules above. From this definition, we derive the following claim:

Claim 0.1. *Let $X \subseteq [n]$ be a subset of courses. Then if we write $X = \{i_1, \dots, i_k\}$ with $i_1 < \dots < i_k$, we have that X is feasible if and only if $\mu \leq d_{i_\mu}$ for each $\mu = 1, \dots, k$.*

Proof. Suppose that $\mu \leq d_{i_\mu}$ for each $\mu = 1, \dots, k$. Then revising for the course i_μ on the day μ for each $\mu = 1, \dots, k$ gives the student a strategy that respects the conditions. Let us check that this strategy checks the conditions given above:

1. The first day of preparation is on $(\mu = 1)$'st day and not before.
2. On day μ, the student prepares the course i_μ, which is only one course for each $\mu = 1, \dots, k$.
3. That is why we have $\mu \leq d_{i_\mu}$ and not $\mu < d_{i_\mu}$ for each $\mu = 1, \dots, k$.
4. Since $\mu \leq d_{i_\mu}$ for each $\mu = 1, \dots, k$, we have that the student will have prepared the examination i_μ the day μ which is before the day of its exam d_{i_μ} for each $\mu = 1, \dots, k$.
5. We do not assume that exams can not take place the same day. This proves that X is feasible. Now suppose that X is feasible and suppose that $\mu > d_{i_\mu}$ for some $\mu \in \{1, \dots, k\}$. Then trying to prepare for the courses $Y = \{i_1, \dots, i_\mu\}$ is impossible. Indeed suppose we have a preparation strategy for Y. Then this means we are trying to prepare for μ examinations in d_{i_μ} days. By the pigeonhole principle, this means that the student needs to work on 2 courses the same day at least one day, which violates one of the rules. Noting that a preparation strategy for X can be restricted to a preparation strategy for Y (by forgetting the preparation days for the courses not contained in Y for example), we see that we reach a contradiction. Thus such a μ can not exist and $\mu \leq d_{i_\mu}$ for each $\mu = 1, \dots, k$.

□

Now let us define the so-called "well chosen matroid". We choose $E = [n]$ as the ground set and $I = \{X \subseteq [n] | \ X \text{ is feasible}\}$ as the independent sets to form the matroid $M = (E, I)$. Now we need to prove that M is indeed a matroid by checking the 2 conditions:

1. Suppose we $X \in I$ and $Y \subseteq X$. Then, as said before, a preparation strategy for X gives a preparation strategy for Y by forgetting some of the preparation day. So $Y \in I$.

2. Now suppose we have $X = \{i_1, \dots, i_k\}, Y = \{j_1, \dots, j_l\} \in I$ and $l > k$ with

$$i_1 < \dots < i_k,$$
$$j_1 < \dots < j_l.$$

   and $j_l \notin X$. We distinguish 3 cases.
   If $j_l > i_k$, then the set $X \cup \{j_l\} = \{i_1, \dots, i_k, j_l\} = \{h_1, \dots, h_{k+1}\}$ is feasible. Indeed we have

$$\mu \le d_{i_\mu} = d_{h_\mu} \text{ if } \mu = 1, \dots, k,$$
$$d_{h_{k+1}} = d_{j_l} \ge l \ge k + 1,$$

   which implies that $X \cup \{j_l\}$ is feasible by the claim above.
   If $j_l < i_1$, then the set $X \cup \{j_l\} = \{j_l, i_1, \dots, i_k\} = \{h_1, \dots, h_{k+1}\}$ is feasible. Indeed we have

$$1 \le d_{j_l} = d_{h_1},$$
$$\mu \le l \le d_{j_l} \le d_{i_{\mu-1}} = d_{h_\mu},$$

   which implies that $X \cup \{j_l\}$ is feasible by the claim above.
   If $i_1 < j_l < i_k$. Let $\mu \in \{1, \dots, k-1\}$ be such that $i_\mu < j_l < i_{\mu+1}$. Then $X \cup \{j_l\} = \{i_1, \dots, i_\mu, j_l, i_{\mu+1}, \dots, i_k\} = \{h_1, \dots, h_{k+1}\}$ is feasible. Indeed we have that

$$\nu \le d_{i_\nu} = d_{h_\nu}, \text{ if } \nu = 1, \dots, \mu,$$
$$\mu + 1 \le l \le d_{j_l} = d_{h_{\mu+1}},$$
$$\nu \le l \le d_{j_l} \le d_{i_\nu} = d_{h_\nu}, \text{ if } \nu = \mu + 2, \dots, k + 1,$$

   which implies that $X \cup \{j_l\}$ is feasible by the claim above.

Now let us set the set of weights $w = (c_i)_{i \in E}$ and run the greedy algorithm on M with weights C. Let R be the resulting set of courses. Suppose that we have R^* be an optimal solution. Note that since R^* respect the preparation rules, then R^* is a feasible set and thus $R^* \in I$. In fact the optimality of R^* tells us that it is a base of maximal weight of the matroid M. By theorem 4 of the lecture notes 1, we get that

$$\sum_{r \in R} c_r \ge \sum_{r \in R^*} c_r$$

which translates to an equality with taking the optimality of R^* into account. Thus R is indeed an optimal solution to our problem.

2 - Safe Travels

0.1 3 - Implementation

Just do max-flow/min-cut ????????