10802EE 398000 Algorithms

Homework 8. Selecting Courses

● Introduction

In this homework, I will design an algorithm to help a student selecting courses to maximize his learning. The available courses have been list in the **courses.dat** file. The first line is the number of courses available, followed by the courses. Each line contains a single course: the course number, number of credits, capacity (number of students), time of the course and, finally, the name of the course.

Since we can use only greedy method, we have to identify four important concepts in greedy algorithm of this problem: Objective function, Constraints, Feasible solutions, and Optimal solution.

There are **n** available courses in the course.dat. Let credit of the course i equal to $\mathbf{c_i}$. $\mathbf{x_i \in \{0, 1\}}$ such that $x_i = 1$ if course i is picked.	
Objective function	The goal is to get the maximum number credits as possible. Thus, the objective function will be: $\sum_{i=1}^{n} x_i c_i$ **(1)**
Constraints	**There is no two courses selected at the same time. (2)**
Feasible solutions	Any solutions that satisfy constraints **(2)**.
Optimal solution	The solutions that satisfy constraints **(2)** and **maximize formula (1)**.

Table 1 Some important concepts in greedy algorithm of this problem

The purpose of this question is now much clearer: We have to help a student selecting courses to **maximize** $\sum_{i=1}^{n} x_i c_i$. Please be careful that **there is no two courses selected at the same time.**

In **Approach** part, I will first describe how I deal with the input data, some structures and methods will be shown. Then I will describe my greedy algorithm, and answer three questions: Is this system Matroid? Can this algorithm get the optimal result? If so, is the optimal results unique?

In **Results** part, I will show the weekly schedule, total credits, and number of courses selected using my algorithm. Also, I will make the conclusion of this homework.

● Approach

First, I will introduce how I deal with the input data. The structure I use is shown below:

```
typedef struct _course {  // stucture for course data
    char* courseNum;      // course number
    int credit;           // number of credits
    int capacity;         // number of students
    char* time;           // time of the course
    char* name;           // the name of the course
    int numOfTime;        // number of lessons per a week
    int** timeSch;        // the indices in the schedule of the course
    double ratio;         // credit / numOfTime
    short choose;         // chosen(1) or not(0)
} course;                 // the new data type
```

Algorithm 1 The structure to store the input data.

In **Algorithm 1**, I store the course number, number of credits, capacity (number of students), time of the course and the name of the course. Besides, I calculate number of lessons per a week and get the indices in the weekly schedule of the course. The weekly schedule and its corresponding

indices are shown below:

[Index] **number**	[0] **1**	[1] **2**	[2] **3**	[3] **4**	[4] **n**	[5] **5**	[6] **6**	[7] **7**	[8] **8**	[9] **9**	[10] **a**	[11] **b**	[12] **c**
[0] **M**													
[1] **T**													
[2] **W**													
[3] **R**													
[4] **F**													

Table 2 The weekly schedule and its corresponding indices. M, T, W, R, F are Monday to Friday.

1, 2, 3, 4, n, 5, 6, 7, 8, 9, a, b, c are lessons in a day.

To get the indices, I use two tables to map **number** to **row [Index]** and **column [index]** respectively. Also, there are two more variables in the structure: ratio and choose. Ratio is the value of credit over the number of lessons per a week, and choose is whether I choose this course or not.

We can start to design our greedy algorithm. I have tried three kinds of greedy algorithms. The general approach is shown in the next page.

Three kinds of greedy algorithms are only different in **Line 3: lis[1 : n] := lis sorted in certain order.** I have tried three kinds of sorting criteria: lis sorted by decreasing credit, lis sorted by decreasing ratio, and lis sorted by increasing number of lessons per a week. Since we want to make the total credit be maximized, it is really intuitive that we pick the course with more credits first. However, if we can fill the schedule as full as possible, it might result in more credits. Thus, I try to pick the course with less number of lessons first too. Last, I try to pick the course with larger ratio first. The ratio is equal to (credit / number of lessons). This is the combination of the previous two. The results are shown in **Results** part.

```
// Greedy method of this problem.
// Input: lis[1 : n], n
// Output: weekly schedule, _ans (total credits), _num (the number of courses selected),
//          updated lis[1:n] (at lis[1:n].choose part)
1 Algorithm greedy(lis, n, weekly schedule, _ans, _num)
2 {
3       lis[1 : n] := lis sorted in certain order;
4       for i := 1 to n do lis[i].choose := 0;
5       _ans = 0;
6       _num = 0;
7       for i := 1 to n do {
8             if (no course is at that time) then {
9                   lis[i].choose = 1;
10                  _ans = _ans + lis[i].credit;
11                  _num = _num + 1;
12                  fill the weekly time schedule;
13            }
14      }
15      return _ans, _num;
16 }
```

Algorithm 2 The general pseudocode of greedy.

The overall space complexity of **Algorithm greedy** is O(n) because it is characterized by n, the number of elements in the list.

The time complexity of **Algorithm greedy** is determined by the sorting part where I use the merge sort. Thus, the time complexity is O(n lg n).

Now, we have to answer some questions:

1. Is this system Matroid?

To answer this question, we first have to check whether the system is an independence system. The definition of an independence system is shown in the next page:

Let S be a finite set and I = {X : X ⊆ S}, then the set system (S, I) is an independence system if **(M1) ∅ ∈ I;** **(M2) If Y ∈ I and X ⊆ Y then X ∈ I.** The elements of I are called **independent**, the elements of $2^S \setminus I$ dependent. Minimal dependent sets are called **circuits**, maximal independent sets are called **bases**. For X ⊆ S, the maximal independent subsets of X are called bases of X.
Definition 1 Independence System.

Let **S** be a set of all courses available, **I** is the set of courses selected and no two courses at the same time. **The system (S, I) is an independent system** because **∅ ∈ I** (That is, we can select zero course), and **If Y ∈ I and X ⊆ Y then X ∈ I** (That is, the subset of courses we have selected also belongs to **I** since it must be no two courses at the same time.)

Then we have to check whether an independence system (S, I) is a matroid. The definition is shown below:

An independence system (S, I) is a matroid if **(M3) If X, Y ∈ I and |X| > |Y|, then there is an x ∈ X \ Y such that Y ∪ {x}∈ I.**
Definition 2 From Independence System to a matroid.

Unfortunately, **the system (S, I) in this problem is not a matroid**. We can find a counter example: X is a set that I choose EE380000 R7R8R9 and EE345000 T7T8T9 and Y is a set that I choose EE323500 T7T8R7. X, Y ∈ I because there is no courses at the same time in two sets

respectively. Also, |X| = 2 > 1 = |Y|. However, **there is no x ∈ X \ Y such that Y ∪ {x} ∈ I.** If x = EE380000 R7R8R9, **Y ∪ {x} ∉ I** because there is an overlap at R7. Also, if x = EE345000 T7T8T9, **Y ∪ {x} ∉ I** because there are overlaps at T7 and T8.

Although the system is not Matroid, greedy method may still works for this problem because Matroid is a necessary but not a sufficient condition for greedy method. So, can greedy method works in this problem? That is, can we use the **Algorithm greedy** (Algorithm 2) to find the optimal solution?

2. Can this algorithm get the optimal result?

Among three greedy algorithms I have designed (That is, pick the course with **larger credit** first, pick the course with **larger ratio** first, and pick the course with **less number of lessons** first), we can just argue whether pick the course with **larger credit** first will result in the optimal result since the other two have the less total credits (In **Results** part, **Figure1, Figure2, and Figure3**). So, can we get the optimal result using **Algorithm greedy** (Algorithm 2) by picking the course with larger credit first? It seems to me that the answer is false: **we can't get the optimal solution** based on **Algorithm greedy.** Here is a counter example:

```
3
EE001 4 100 M1M2W1W2 Course1
EE002 3 100 M1M2T1 Course2
EE003 3 100 W1W2F1 Course3
```

Example 1 The example input file.

If there are only three courses available, we will pick EE001 if we use **Algorithm greedy**

(Algorithm 2) by picking the course with larger credit first. However, the optimal solution will be choosing EE002 and EE003, which can make total credits equal to 6 instead of 4. Thus, the greedy algorithm **may not always work to find the optimal solution in this problem**.

- **Results**

```
Total credits: 37
Number of courses selected: 12
1: MATH102006 4 T3T4R3R4 Calculus (II)
2: MATH202001 4 T1T2F1F2 Advanced Calculus II
3: EE211000 3 T7T8R7 Modern Physics
4: EE214001 3 M3M4W2 Electromagnetism
5: EE231000 3 M1M2R1R2 Introduction to Programming
6: EE313000 3 W5W6R8R9 Optics and Photonics
7: EE335000 3 W3W4F4 Introduction to Solid-State Electronic Devices
8: EE336000 3 M7M8M9 Opto-electronic Devices
9: EE364000 3 M5M6RnR5 Communication Systems (I)
10: EE413500 3 T5T6F3 Principle of Lasers
11: EECS340000 3 RaRbRc Satellite Electrical System Design
12: EE240500 2 W7W8W9 Embedded System Laboratory
Weekly schedule
  1 2 3 4 n 5 6 7 8 9 a b c
M V V V V . V V V V V . . .
T V V V V . V V V V . . . .
W . V V V . V V V V V . . .
R V V V V V V . V V V V V V
F V V V V . . . . . . . . .
```

Figure 1 the results of picking the course with **larger credit** first.

In this report, we have proven that using Best-In Greedy Algorithm based on credits, number of lessons, and ratio cannot work in this problem. The Best-In Greedy Algorithm is a kind of greedy algorithm that try all elements of the set in certain order. If adding it to the final set can maintain independence, then add it to the final set. **Algorithm greedy** (Algorithm 2) is a Best-In Greedy Algorithm. However, it doesn't work in this problem. Briefly speaking, the reason is that the time of courses are too flexible (**Example 1**).

```
Total credits: 36
Number of courses selected: 12
1: EE211000 3 T7T8R7 Modern Physics
2: EE214001 3 M3M4W2 Electromagnetism
3: EE335000 3 W3W4F4 Introduction to Solid-State Electronic Devices
4: EE336000 3 M7M8M9 Opto-electronic Devices
5: EE351000 3 T3T4R3 Feedback Control Systems
6: EE366000 3 W5W6R8 Introduction to Digital Signal Processing
7: EE413500 3 T5T6F3 Principle of Lasers
8: EECS340000 3 RaRbRc Satellite Electrical System Design
9: MATH202001 4 T1T2F1F2 Advanced Calculus II
10: EE231000 3 M1M2R1R2 Introduction to Programming
11: EE364000 3 M5M6RnR5 Communication Systems (I)
12: EE240500 2 W7W8W9 Embedded System Laboratory
Weekly schedule
  1 2 3 4 n 5 6 7 8 9 a b c
M V V V V . V V V V V . . .
T V V V V . V V V V . . . .
W . V V V . V V V V V . . .
R V V V . V V . V V . V V V
F V V V V . . . . . . . . .
```

Figure 2 the results of picking the course with **larger ratio** first.

```
Total credits: 34
Number of courses selected: 12
1: YZ998601 2 W3W4 English
2: PME235002 3 T3R3R4 Mechanics of Materials
3: MATH242000 3 M3M4W2 Algebra II
4: LSC110200 3 R7R8R9 Life Science II
5: ENE553000 3 T7T8T9 Terahertz Science and Technology
6: EECS340000 3 RaRbRc Satellite Electrical System Design
7: EECS302000 3 M7M8R6 Introduction to Computer Networks
8: EE465000 2 W7W8W9 Communications System Laboratory
9: EE413500 3 T5T6F3 Principle of Lasers
10: YZ971000 3 T1T2F1F2 General Physics B (II)
11: EE364000 3 M5M6RnR5 Communication Systems (I)
12: EE231000 3 M1M2R1R2 Introduction to Programming
Weekly schedule
  1 2 3 4 n 5 6 7 8 9 a b c
M V V V V . V V V V . . . .
T V V V . . V V V V V . . .
W . V V V . . . V V V . . .
R V V V V V V V V V V V V V
F V V V . . . . . . . . . .
```

Figure 3 the results of picking the course with **less number of lessons** first.

In my opinion, the greedy algorithm cannot work because it doesn't consider the distribution of time of courses. Maybe other methods such as backtracking or dynamic programming work instead.