

Due Apr 24, 2:45pm

**Instructions:** You may work in groups of up to 4 people (but make sure to list your study partners on your homework). You do not need to provide pseudocode on this homework if you explain your algorithm clearly, in enough detail to enable someone else to implement your algorithm, given your written solution. Unless stated otherwise, you do not need to provide a proof of correctness for your algorithms on this homework. You may assume that linear programming problems can be solved in polynomial time, i.e., in time polynomial in the number of variables and the number of constraints.

**1. (25 pts.) Learn linear programming**

You are the manager of a large post office, and you need to ensure that on each day you have enough employees to handle the demand. You've noticed that there is a very strong weekly pattern to the demand, so that the number of employees needed on any particular day depends only upon which day of the week it is (e.g., Mon, Tue, etc.). In particular, the minimum number of employees who will be needed on each day of the week is as follows:

Mon	Tue	Wed	Thu	Fri	Sat	Sun
21	17	16	19	22	15	8

You need to hire enough employees to ensure that you will meet these minimums on each day of the week. Due to federal regulations, each employee must work 5 days in a row, then take two days off; each employee's 5-day work segment can start on any day of the week you choose, but they must strictly alternate 5 days on duty, followed by 2 days off duty. For instance, if you hire an employee whose work week starts on Wednesday, they will be working on Wed, Thu, Fri, Sat, Sun, and will be off-duty on Mon and Tue. Due to the recession, there's an effectively unlimited supply of potential workers willing to start on any particular day of the week. Due to federal pay scales, every employee is paid the same amount, so your total costs are proportional to the number of employees you hire.

Your goal is to hire the minimum number of employees sufficient to ensure that you meet the minimums shown above on each of the 7 days of the week. How many employees do you need?

**How to solve it.** Solve this problem with linear programming. You should write down a linear programming problem instance. Then, solve it with a linear programming solver. I've included a tutorial on how to use one linear programming solver, below. This problem will give you practice with translating a problem statement into a linear programming problem, and it will give you practice with using a linear programming solver.

**How to use `lp_solve`.** The instructional staff have installed a linear solver named `lp_solve` on the instructional Unix machines. It is easy to use. Here is a brief tutorial.

Suppose we had the following simple linear program:

```
maximize  $x_2 + 2x_3$ 
subject to:
   $0 \leq x_1 \leq 7$ 
   $5 \leq x_2 \leq 10$ 
   $0 \leq x_3 \leq 6$ 
   $x_1 + x_2 \leq x_3$ 
```

We can translate this into the `lp_solve` input format by storing the following into a file called `input.lp`:

```
maximize: x2 + 2*x3;
0 <= x1 <= 7;
5 <= x2 <= 10;
0 <= x3 <= 6;
x1 + x2 <= x3;
```

(Any other filename would be fine, too, as long as it ends with the `.lp` extension.) Then we execute the command

```
lp_solve input.lp
```

and the solver gives us the following output:

```
Value of objective function: 18.00000000

Actual values of the variables:
x2          6
x3          6
x1          0
```

which tells us that the maximum attainable value of the objective function is 18, and one way to achieve this is by setting  $x_1 = 0$ ,  $x_2 = 6$ , and  $x_3 = 6$ . Pretty easy, right?

A few extensions are also supported. If you want to find the minimum possible value of the objective function, rather than the maximum, use `minimize:` on the first line rather than `maximize:`. If you want to force a variable `x2` to be an integer, add the line

```
int x2;
```

to the input file.

The `lp_solve` solver is installed on the Unix instructional machines, so you may want to log into your Unix instructional account and use it there. (It is an open source package, so in principle you could download it and run it on your own machine, but it will probably be easier to just use the installation that the instructional staff have already kindly prepared for you.)

**What you should turn in.** Please turn in: (1) a description of the main idea of your solution, including a specification of the variables and what each variable intuitively corresponds to; (2) a print-out of the `lp_solve` input file that you used; (3) the final answer (the minimum number of employees needed to meet these constraints—we're looking for a number).

## 2. (30 pts.) Henron

You've been hired by Henron, a new startup with this great idea to make a killing by creating a market for chicken futures. Henron has relationships with a set of  $n$  suppliers and a set of  $m$  purchasers. The  $i$ -th supplier can supply up to  $s[i]$  chickens this year, and the  $j$ -th purchaser would like to buy up to  $b[j]$  chickens this year. Henron is the middleman and makes \$1 off each chicken that is sold. Thus, the more chickens that are sold, the more money you make!

However, there's a complication. Due to federal restrictions on interstate trafficking in avian life forms, supplier  $i$  can only sell chickens to a purchaser  $j$  if they are situated at most 100 miles apart. Assume that you're given a list  $L$  of all the pairs  $(i, j)$  such that supplier  $i$  is within 100 miles of purchaser  $j$ . You will be given  $n, m, s[1..n], b[1..m], L$  as input. Your job will be to compute the maximum number of chickens that can be sold this year. The running time of your algorithm should be polynomial in  $n$  and  $m$ .

- Formulate this as a network flow problem. In other words, show how to solve this problem, using a network flow algorithm as a subroutine. Show or describe the graph you have in mind, and explain why the output from the network flow algorithm gives a valid solution to this problem.
- Formulate this as a linear programming problem. In other words, show how to solve this problem, using a linear programming solver as a subroutine. Explain why this correctly solves the problem.
- Let's assume you don't care about the running time of your algorithm. Which formulation would be better, network flow or linear programming? Explain your answer.

HINT: You can't sell fractional chickens. (Can you imagine the mess?)

## 3. (25 pts.) Optimal gerrymandering

It will soon be election time in Upper Moldavia, and things don't look good for the Republican party: the majority of Upper Moldavians plan to vote Democan next year. However, the Republican party has a secret weapon up their sleeve: they control the committee that will be drawing up the map of voting districts. The election will be winner-take-all: in each district, whichever party gets the most votes receives one representative elected to the Moldavian Senate. Your job will be to find an algorithm the Republicans could use to maximize the number of representatives elected for their party next election.

There are  $n$  counties in Upper Moldavia. Exactly  $r_i$  of the residents in county  $i$  are Republican, and the other  $d_i$  residents are Democan. No one votes for a third party. In your power as mapmaker,

you must build  $m$  voting districts by amalgamating pieces from the various counties. A district can be made up of many portions of many counties, and each portion can be obtained by drawing off any fraction  $f_{i,j}$  of the  $i$ -th county towards the  $j$ -th district (thereby adding  $f_{i,j} \cdot r_i$  Republicanrats and  $f_{i,j} \cdot d_i$  Democans to the  $j$ -th district). For instance, district 1 might be composed of  $\frac{1}{2}$  of county A and  $\frac{1}{3}$  of county B; district 2 might include  $\frac{1}{3}$  of county A and all of county C; and district 3 might include  $\frac{1}{6}$  of county A and  $\frac{2}{3}$  of county B.

You are subject only to the restrictions that each of the  $m$  districts you draw up must contain at least  $10^6$  voters, and that every voter is in exactly one district. Your algorithm will receive as input the values  $n, m, r_i, d_i$ . You must find the districting assignment (given by the  $f_{i,j}$ 's) that maximizes  $E$ , the number of Republicanrat representatives that will be elected under your assignment. Your algorithm must run in time polynomial in  $n$  and  $m$ .

HINT: binary search, linear programming.

#### 4. (20 pts.) Spreading the work fairly

We're setting up a carpool for a group of  $n$  people, over a period of  $m$  days. Each day, a subset of the people want to participate in the carpool. Let  $S_i$  denote the set of people participating in the carpool on day  $i$ , for each  $i \in \{1, 2, \dots, m\}$ . We want to assign one person to drive on each day. The driver on day  $i$  has to be one of the people in  $S_i$ , and the rest of the people in  $S_i$  will be passengers. (We'll assume the car has plenty of room for everyone who wants to participate on any given day.)

The thing is, no one wants to drive if they don't have to. So the goal is to come up with a fair schedule, where no one drives more than their fair share, defined as follows:

Suppose Alice participates in the carpool on  $k$  days, and the number of people carpooling on those days is given by  $a_1, \dots, a_k$ . To be fair, she should not have to drive more than  $1/a_1 + 1/a_2 + \dots + 1/a_k$  days (rounding this expression up to an integer, if it is not already one), since that's her fair share if we divide up the driving equally. Call a schedule fair if no person has to drive more than his/her fair share.

Given the sets  $S_i$ , devise a polynomial-time algorithm to check whether there exists a fair schedule, and if so, find one. (Actually, there's a theorem that says that a fair schedule always exists, so you just need to find one; but you don't need to prove this theorem.)

Your answer should explain the idea of your algorithm; explain informally why it is correct; and justify why its running time is polynomial in  $n$  and  $m$ . Please label each of these categories clearly. As long as the idea is clear and presented in enough detail to enable someone to implement your algorithm from your written solution, you do not need to present pseudocode. As long as your explanation of correctness is convincing, you do not need to provide a formal proof.

HINT: Network flow. (If you follow this hint, make sure to describe or show the graph you use.)

*Revised 4/19 to state that Alice should not have to drive more than  $1/a_1 + 1/a_2 + \dots + 1/a_k$  days (previously, it erroneously said she should not have to carpool more than  $1/a_1 + 1/a_2 + \dots + 1/a_k$  days, which was not the intent).*