Section 8: Solutions

This section has two parts. The first involves getting a bit of practice with thinking about linear programming. You have only had one lecture on it so far but this should help you get used to it. The second part will involve a series of short problems in which we practice how to decide on a technique when faced with a completely new problem.

1. Cost-effective eating

You are given a list of foods indexed $1, \ldots, n$, as well as the calories c_i, sugars (g) s_i, and vitamin D (mcg) d_i in each food. You're trying to maintain a healthy diet by eating exactly 2000 calories per day. You also heard that the American Heart Association recommends at most 30 grams of sugar per day. And because you just moved to Seattle from LA this year, it's your first winter and you need to eat at least 15 mcg of vitamin D to avoid SAD.

Along with the nutrition information, you also know that one serving of food i costs m_i money. Find a way to compute a healthy diet that is as cheap as possible.

(a) Write a summary of the problem.

(b) To use linear programming:

  (i) What should the variables x_i represent?

  (ii) What is the objective function?

  (iii) What are the constraints (directly translated from the problem)?

  (iv) How can you transform the problem into standard form?

(c) Sketch the correctness of your solution.

Solution:

If we pick x_i units of food i, then the money we pay is $m_i x_i$, so our goal is to minimize $m_1 x_1 + \cdots + m_n x_n$. Similarly, we are given per-unit values for the amount of calories, sugar, and vitamin D, so our constraints are $c_1 x_1 + \cdots + c_n x_n = 2000$, $s_1 x_1 + \cdots + s_n x_n \leq 30$, and $d_1 x_1 + \cdots + d_n x_n \geq 15$. We transform the non-standard form constraints into standard form using algebra to arrive at the following:

$$\begin{aligned}
\textbf{maximize } & -m_1 x_1 - \cdots - m_n x_n \\
\textbf{subject to } & c_1 x_1 + \cdots + c_n x_n \leq 2000 \\
& -c_1 x_1 - \cdots - c_n x_n \leq -2000 \\
& s_1 x_1 + \cdots + s_n x_n \leq 30 \\
& -d_1 x_1 - \cdots - d_n x_n \leq -15 \\
& x \geq 0
\end{aligned}$$

Relying on an LP algorithm, we output the best $x_1, \ldots, x_n$ as desired.

Generally, the technique is to:

- Multiply a "minimize" objective by -1 to convert it into a "maximize objective".
- Convert $a = b$ into $a \geq b$ and $a \leq b$.
- Multiply any $a \geq b$ constraints by -1 to obtain $-a \leq -b$.

2. Technique toolbox

We have learned many useful algorithms in this class, including:

- Stable matching
- Graph traversal algorithms (B/DFS, topological sort, etc.)
- Weighted (greedy) graph algorithms (Dijkstra's, various MST algorithms)
- Network flows
- Linear programming

We have also learned several techniques to develop algorithms from scratch:

- Greedy algorithms
- Divide and conquer
- Dynamic programming

When faced with a new problem, how do we choose a technique? Generally, it would be good to follow the following steps:

- Read and summarize the problem.
- Ask yourself: does the problem remind me of something I already know?
  - If so, call that algorithm as a subroutine and do the necessary pre/post-processing.
  - If not,
    - ⋆ Visualize the problem with examples.
    - ⋆ Try a greedy idea against your examples first.
    - ⋆ If greedy doesn't work, try to identify subproblems. Are they half-sized or just slightly smaller?

Now, give it a try with the 16 problems below.

(a) There are a total of n courses you have to take, labeled from 1 to n. You are given a list `prerequisites` where $\texttt{prerequisites}[i] = (a_i, b_i)$ indicates that you must take course b_i first if you want to take a_i. Decide if you can take all courses.

Solution:

Graph traversal algorithms. The list `prerequisites` contains pairs of courses that can be represented as edges and the course numbers can represent vertices in a graph. The problem is equivalent to finding cycles in this graph.

(b) The landlord of an apartment building with n tenants is trying to maximize its profits. Tenant i has starting happiness $h_i \geq 0$ and will move it out if this score drops below zero. Raising their rent by $\$d$ will cause their happiness to decrease by $r_i d$, and spending $\$d$ on amenities for the entire building will cause their happiness to increase by $a_i d$ (that is, each tenant may react differently to the same changes). If the landlord applies the same rent change and amenities to all residents, what is the maximum profit increase that can be attained without any tenants moving out?

Solution:

Linear programming. Making several real-valued choices with linear constraints is typically a clear sign that linear programming may be helpful.

(c) In a university, there are n students who are applying to be TAs for m courses. Course j requires exactly t_j TAs, and each student can only TA one course. It is also a requirement to have previous taken the course that you TA for, so some students are ineligible for certain courses.

The students rank the courses that they would like to TA for, and the instructors of those courses rank the students as well. A student-course pair (i, j) will be *unhappy* if they are not matched and student i is eligible to TA for course j, but student i is either currently unmatched or prefers course j to their current match, and the instructor for course j prefers student i over at least one of the TAs currently assigned to their course. Find a valid assignment in which no student-course pair is unhappy, or return "not possible".

Solution:

Stable matching. The existence of preferences makes stable matching a natural choice.

(d) (KT) Suppose you are given a connected graph $G = (V, E)$, with distinct edge costs $c(e)$ for all $e \in E$. Given an edge e, decide whether e is contained in a minimum spanning tree of G in $O(m + n)$ time (where $m = |E|$ and $n = |V|$).

Solution:

Graph traversal algorithms. The MST algorithms studied in this class do not take $O(m + n)$ time. Using the "cut" and "cycle" properties of MSTs, $e = (u, v)$ does not belong to an MST iff there is a path from u to v using only edges cheaper than e. Thus, the algorithm is: construct a graph G' by deleting all edges with weight at least $c(e)$, and use B/DFS to determine if there is a path from u to v.

(e) You are a thief stealing bulk items from Whole Foods and want to maximize the value of the items being packed. Your bag has a maximum volume v, and you are very strong and don't care about weight. For goods $1, \ldots, n$, there are ℓ_i liters of bulk good i in the dispenser, which has density d_i kilograms per liter and is being sold at price p_i dollars per kilogram. Determine the best choice of items to pack.

Solution:

Greedy. There is a natural way to sort the "value" of the items (by price per liter), and picking items now does not prevent us from picking better items later, so greedy seems like a good choice.

(f) Let A be an $m \times n$ integer matrix, where each row and each column is sorted in ascending order. Given an integer `target`, determine whether or not `target` is in A.

Solution:

Divide and conquer. When the input is sorted, divide and conquer should be a first thing to try.

(g) (CLRS) There are m people playing Mario Kart on $n \times n$ grid, starting at distinct locations $(x_1, y_1), \ldots, (x_m, y_m)$. This is a strange map where the finish line is the edge of grid. Players can finish at any point on the edge. However, all players leave a trail of banana peels, so no two players' routes can overlap along any roads or intersections. Determine if it is possible for every player to reach the finish line.

Solution:

Network flow. Travel planning is often a network flow problem, and here the overlap restrictions can be phrased as vertex/edge capacities.

(h) (CLRS) You are a politician running for local office, and you want to appeal to a wide voter base. There are k

groups of voters, let n_i be the number of voters in the ith group, and you want at least half of each group to vote for you. Without any campaigning, a_i voters from group i will vote for you ($0 \le a_i \le n_i$).

Your campaign staff have determined that there are m issues that voters care about, and they will react differently depending on their group. In particular, for every $1000 you spend on advertising for issue j, d_{ij} is the number of additional voters in group i who will now vote for you. (If d_{ij} is negative, it means you lost voters in group i.) Determine the minimum advertising cost so that at least half of each group votes for you.

Solution:

Linear programming. Again, making real-valued choices with linear constraints.

(i) (KT) Suppose you are hiking the Appalachian Trail, and you can hike d miles every day before it gets dark. Along the trail, there are good resting sites at locations $0 = x_1 < x_2 < \cdots < x_{n-1} < x_n$. Find a minimum set of a resting sites that allow you to rest at least once every d miles, or return "not possible" if not possible.

Solution:

Greedy. Choosing the farthest resting site within d miles does not prevent us from being able to choose a better sequence of sites later on.

(j) Given a positive integer n, return the least number of perfect square numbers that sum to n.

Solution:

Dynamic programming. If a perfect square s is chosen to be part of the solution, then finding the least number of perfect square numbers that create $n - s$ is another sub-problem that should be memoized.

(k) A city is planning a network of snow routes. The city has n destinations $1, \ldots, n$, and bidirectional roads connect certain destination pairs. The road from destination i to j has length $d(i, j)$ in feet. The city will salt the roads with w pounds per foot in order to prevent ice. Regulations require all roads with an endpoint at city hall (located at c) must be clear of ice. Determine the minimum quantity of salt needed to ensure that this requirement is met, and additionally, every two destinations has an ice-free route between them.

Solution:

Weighted graph algorithms. The problem appears to be a variant of MST.

(l) Given an integer array `nums`, return an integer array `smaller` where `smaller`$[i]$ is the number of elements to the right of `nums`$[i]$ that are smaller than `nums`$[i]$.

Solution:

Divide and conquer. Since we want to find the number of smaller elements to the right of an element in an unsorted array, there should probably be some sorting involved, so a modified divide and conquer sorting algorithm can be used.

(m) Given two strings `source` and `target`, compute the minimum number k such that `target` is a subsequence of `source`^{k}, where `source`^{k} denotes concatenating k copies of `source`. (Subsequences may be non-contiguous.) If the task is impossible, say "not possible".

Solution:

Greedy. Looping through `source` and taking the first character that is still not taken in `target` does not block any future good choices from being taken.

(n) A network of one-way highways connects n cities. For some city pairs, traveling along the direct highway from city i to city j will cost a toll of $c(i, j) \geq 0$. For other city pairs, the government is trying to increase economic activity and will actually pay you $b(i, j) > 0$ to drive on the highway. Return the minimum total cost to go from city s to city t (which may be negative, if you get a net reward), or "not possible" if it is not possible to make the trip at all.

Solution:

Dynamic programming on graphs. This is an extension of Bellman–Ford, with an additional parameter for how many coupons we have access to.

(o) (KT) A hospital is trying to schedule its doctors during vacation periods. There are k vacation periods $D_1, \ldots, D_k$, and denote $D = D_1 \cup \cdots \cup D_k$ the set of all vacation days. There are n doctors, and each doctor provides a set of days $S_i \subseteq D$ when they are available to work if needed. There is a government-mandated limit c on the total number of days any doctor can be asked to work during vacation periods. Determine if there exists an assignment such that:

- Every vacation day in D has at least 1 doctor at the hospital.
- No doctor works more than 1 day per vacation period D_j.
- No doctor works more than c days total.
- No doctor works when they indicate they are unavailable (with S_i).

If so, output the assignment.

Solution:

Network flow. Variations of bipartite matching are often a good place to use network flow algorithms. In this case, we are matching doctors with days in each vacation period.

(p) You are given a list of integers `coins` representing coins of different denominations and an integer `amount` representing a total amount of money. Return the fewest number of coins you need to make up that amount. If that amount of money cannot be made up by any combination of coins, return "impossible".

Solution:

Dynamic programming. By using a coin, the total amount of money remaining is reduced, which becomes the next subproblem.

The following problems will not be covered in section, but may be useful to think about.
We recommend trying them by yourself first. Solutions will be posted in the evening.

3. Diet happiness

You need to get a certain amount of nutrition: protein, calories, fat, vitamins, but you also shouldn't get too much of certain things. Each quantity of food you could chose has certain amounts of each but you are also happier eating amounts of some foods versus others. The general problem here is to figure out how to choose a combination of amounts of food in your diet to maximize your happiness while meeting your nutritional needs.

Suppose that someone has no dietary restrictions but has food options per 100 grams:
Bread: 7g protein, fat 2 g, calories 260, happiness 5
Cheese: 27g protein, fat 17g, calories 250, happiness 4
Meat: 30g protein, fat 9g, calories 200 , happiness 2
Lentils: 10g protein, fat 1g, calories 120, happiness 3
Daily values: At least 50g protein, 80g fat, 2000 calories. No more than 100g fat, no more than 2300 calories.

Write this optimization problem as a linear program in standard form.

Solution:

We define 4 variables b, c, m, ℓ to express the amount of each food in 100g units to eat each day.

Total happiness is $5b + 4c + 2m + 3\ell$. For protein we need: $7b + 27c + 30m + 10\ell \geq 50$. For fat we need $80 \leq 2b + 17c + 9m + \ell \leq 100$. For calories we need: $2000 \leq 260b + 250c + 200m + 120\ell \leq 2300$.

In standard form we get:

$$\begin{aligned} \textbf{maximize} \quad & 5b + 4c + 2m + 3\ell \\ \textbf{subject to} \quad & -7b - 27c - 30m - 10\ell \leq -50 \\ & -2b - 17c - 9m - \ell \leq -80 \\ & 2b + 17c + 9m + \ell \leq 100 \\ & -260b - 250c - 200m - 120\ell \leq -2000 \\ & 260b + 250c + 200m + 120\ell \leq 2300 \\ & b, c, m, \ell \geq 0. \end{aligned}$$