CSE 421 Section 8

Linear Programming and Technique Toolbox

Administrivia

Announcements & Reminders

- **HW6** was due yesterday, 2/26
  - Late submissions open until tomorrow, 2/28 @ 11:59pm

- **HW7**
  - Due **Wednesday** 3/5@ 11:59pm
  - Late submissions will be open until Friday, 3/7 @ 11:59pm

Linear programming

Review of linear programming

Linear programming is the following problem:

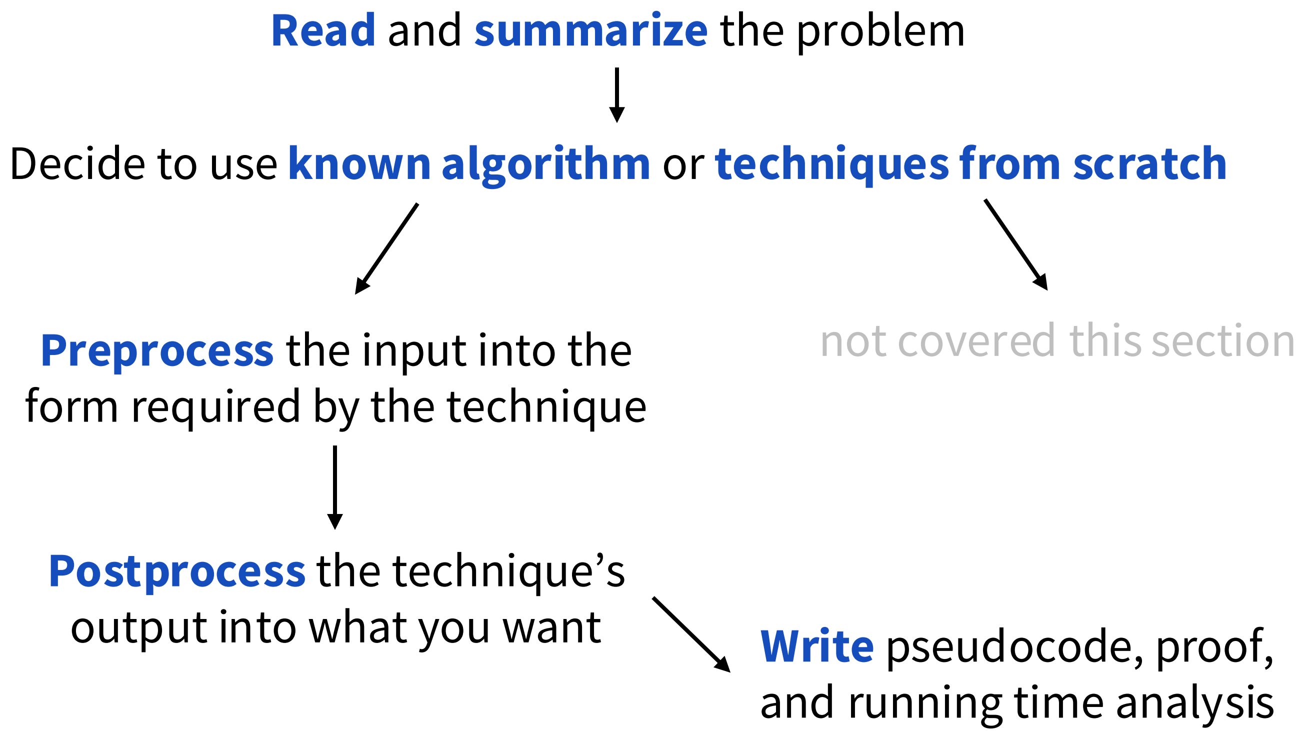
$$\textbf{maximize } c^T x$$
$$\textbf{subject to } Ax \le b, x \ge 0$$

In other words,

$$\begin{aligned}
&\textbf{maximize } c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
&\textbf{subject to } \quad a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \le b_1 \\
&\qquad a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \le b_2 \\
&\qquad \cdots \\
&\qquad a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \le b_m \\
&\qquad x_i \ge 0 \qquad \text{for all } i
\end{aligned}$$

Standard form is maximization with less than or equal to constraints

Problem solving strategy overview



Read and **summarize** the problem

Decide to use **known algorithm** or **techniques from scratch**

Preprocess the input into the form required by the technique

not covered this section

Postprocess the technique’s output into what you want

Write pseudocode, proof, and running time analysis

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 per serving of 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.

Work on this with the people around you, then we'll check!

Solution

Problem 1 – Cost-effective eating

a) Write a summary of the problem.

Input: Calories c_i, sugar s_i, vitamin D d_i, and money m_i per serving for each food.

Expected output: Minimum amount of money to meet nutrition standards.

Problem 1 – Cost-effective eating

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)?

Work on this with the people around you, then we'll check!

Solution

Problem 1 – Cost-effective eating

b) To use linear programming:

i. What should the variables x_i represent?

The number of servings of food i to eat.

ii. What is the objective function?

$$\textbf{minimize}\ m_1x_1 + \cdots + m_nx_n$$

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

$$\begin{aligned} \textbf{subject to}\quad c_1x_1 + \cdots + c_nx_n &= 2000 \\ s_1x_1 + \cdots + s_nx_n &\le 30 \\ d_1x_1 + \cdots + d_nx_n &\ge 15 \\ x_i &\ge 0 \qquad \text{for all } i \end{aligned}$$

Problem 1 – Cost-effective eating

iv. How can you transform the problem into standard form?

Work on this with the people around you, then we’ll check!

Solution

Problem 1 – Cost-effective eating

iv. How can you transform the problem into standard form?

$$\begin{aligned} \textbf{maximize} \quad & -m_1x_1 - \cdots - m_nx_n \\ \\ \textbf{subject to} \quad & c_1x_1 + \cdots + c_nx_n \le 2000 \\ & -c_1x_1 - \cdots - c_nx_n \le -2000 \\ & s_1x_1 + \cdots + s_nx_n \le 30 \\ & -d_1x_1 - \cdots - d_nx_n \le -15 \\ & x_i \ge 0 \qquad \text{for all } i \end{aligned}$$

1. Multiply “minimize” objective by -1.
2. Convert equalities into two inequalities.
3. Multiply $\geq$ inequalities by -1.

Problem 1 – Cost-effective eating

For basic LP problems, we will only be looking for a brief sketch of correctness, unless there is something nontrivial beyond directly translating the constraints.

c) Sketch the correctness of your solution.

Work on this with the people around you, then we’ll check!

Solution

Problem 1 – Cost-effective eating

For basic LP problems, we will only be looking for a brief sketch of correctness, unless there is something nontrivial beyond directly translating the constraints.

c) Sketch the correctness of your solution.

Because h_i is the price per serving, we pay $h_i x_i$ if we pick x_i servings of food i, so our goal is to minimize $h_1 x_1 + \cdots + h_n x_n$. Similarly, we are given per-serving values for the calories, sugar, and vitamin D of each food, so we have $c_1 x_1 + \cdots + c_n x_n = 2000$, $s_1 x_1 + \cdots + s_n x_n \leq 30$, and $d_1 x_1 + \cdots + d_n x_n \geq 15$. We transformed them to standard form using basic algebra. Relying on an LP algorithm, we output the best $x_1, \ldots, x_n$ as desired.

Technique toolbox

Which technique should I try?

We have covered many techniques for algorithms so far.

Using known algorithms	**Developing from scratch**
• Stable matching	• Greedy algorithms
• Graph traversal algorithms	• Divide and conquer
• Weighted graph algorithms	• Dynamic programming
• Network flows	
• Linear programming	

How should you pick which method to try?

Problem solving strategy overview

Read and **summarize** the problem

↓

Does the problem remind me of something I already know?

↙ Yes

No ↘

Call the known algorithm as a subroutine

1. **Visualize** the problem with **examples**.
2. Try a **greedy** idea against your examples.
3. Identify **subproblems**, are they halves or just slightly smaller?

Problem 2 - Technique toolbox activity

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
- Greedy algorithms
- Divide and conquer
- Dynamic programming

When faced with a new problem, how do we choose a technique?

Problem 2 - Technique toolbox activity

Each group will take 4 problems from the section worksheet. For each problem, (1) identify the class of algorithm type that can be used to solve it and (2) give a brief justification as to why that choice is appropriate.

- Group #1: (a) - (d)
- Group #2: (e) - (h)
- Group #3: (i) - (l)
- Group #4: (m) - (p)

There may be multiple algorithms that work (ex. Network flow vs Linear Programming). In that case, pick the one that feels more natural.

Problem 2 - Technique toolbox activity

(a) There are a total of n courses you have to take, labeled from 1 to n. You are given a list `prerequisites` where `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.

- ❑ Stable matching
- ❑ Graph traversal algorithms
- ❑ Weighted (greedy) graph algorithms
- ❑ Greedy algorithms
- ❑ Divide and conquer
- ❑ Dynamic programming
- ❑ Network flows
- ❑ Linear programming

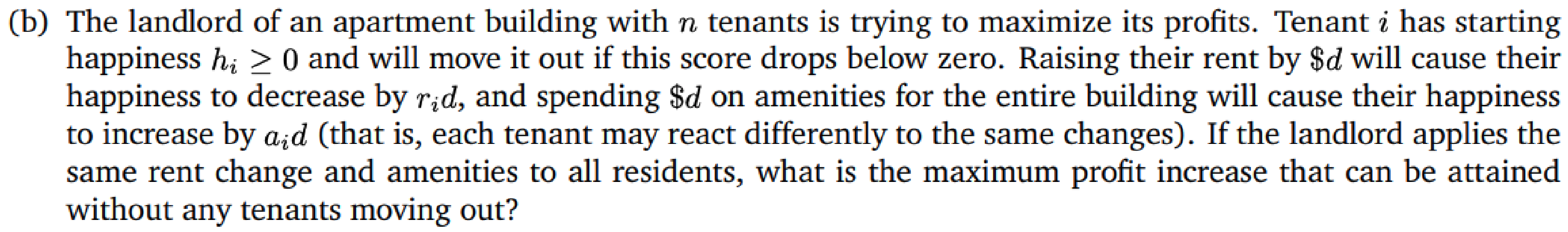
Solution

Problem 2 - Technique toolbox activity

(a) There are a total of n courses you have to take, labeled from 1 to n. You are given a list `prerequisites` where `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.

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.

Problem 2 - Technique toolbox activity



(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?

- ☐ Stable matching
- ☐ Graph traversal algorithms
- ☐ Weighted (greedy) graph algorithms
- ☐ Greedy algorithms
- ☐ Divide and conquer
- ☐ Dynamic programming
- ☐ Network flows
- ☐ Linear programming

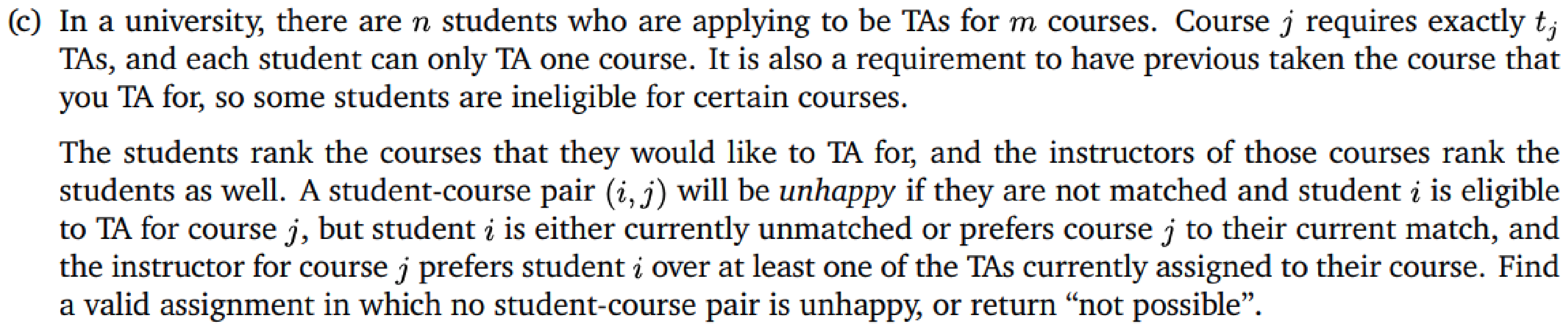
Solution

Problem 2 - Technique toolbox activity

(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?

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

Problem 2 - Technique toolbox activity



(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".

- ❑ Stable matching
- ❑ Graph traversal algorithms
- ❑ Weighted (greedy) graph algorithms
- ❑ Greedy algorithms
- ❑ Divide and conquer
- ❑ Dynamic programming
- ❑ Network flows
- ❑ Linear programming

Solution

Problem 2 - Technique toolbox activity

(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".

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

Problem 2 - Technique toolbox activity

(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 $n = |V|$).

- ☐ Stable matching
- ☐ Graph traversal algorithms
- ☐ Weighted (greedy) graph algorithms
- ☐ Greedy algorithms
- ☐ Divide and conquer
- ☐ Dynamic programming
- ☐ Network flows
- ☐ Linear programming

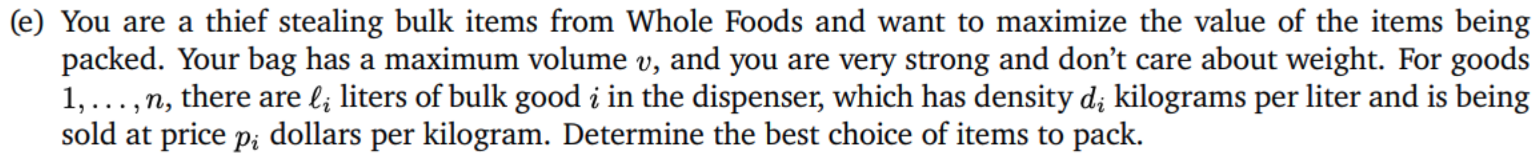
Solution

Problem 2 - Technique toolbox activity

(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 $n = |V|$).

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

Problem 2 - Technique toolbox activity

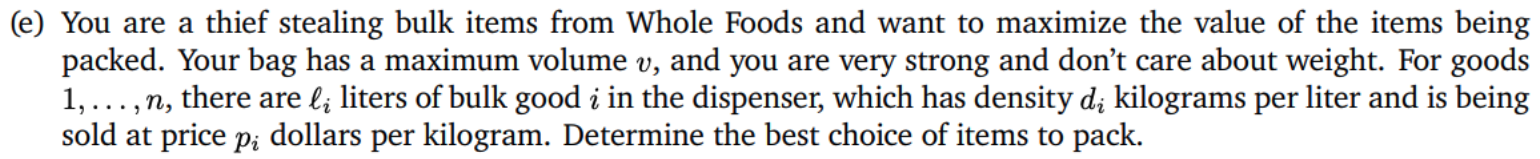


(e) You are a thief stealing bulk items from Whole Foods and want to maximize the value of the items being packed. Your bag has a maximum volume v, and you are very strong and don't care about weight. For goods $1, \dots, n$, there are ℓ_i liters of bulk good i in the dispenser, which has density d_i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

- ☐ Stable matching
- ☐ Graph traversal algorithms
- ☐ Weighted (greedy) graph algorithms
- ☐ Greedy algorithms
- ☐ Divide and conquer
- ☐ Dynamic programming
- ☐ Network flows
- ☐ Linear programming

Solution

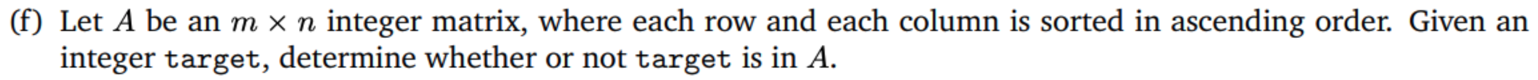
Problem 2 - Technique toolbox activity



(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.

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.

Problem 2 - Technique toolbox activity



(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.

- ❑ Stable matching
- ❑ Graph traversal algorithms
- ❑ Weighted (greedy) graph algorithms
- ❑ Greedy algorithms
- ❑ Divide and conquer
- ❑ Dynamic programming
- ❑ Network flows
- ❑ Linear programming

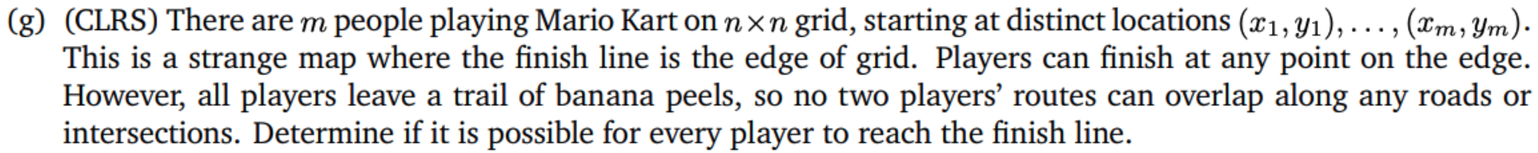
Solution

Problem 2 - Technique toolbox activity

(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.

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

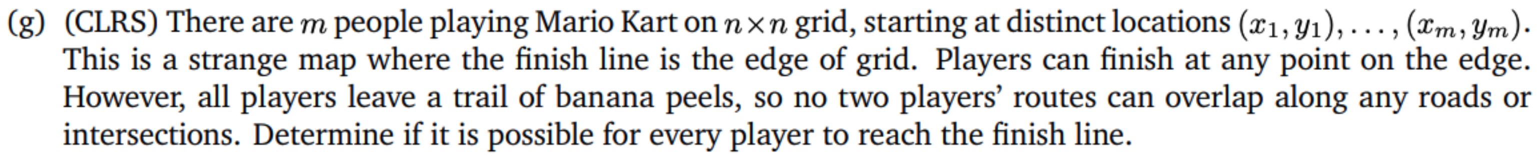
Problem 2 - Technique toolbox activity


(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.

- ❑ Stable matching
- ❑ Graph traversal algorithms
- ❑ Weighted (greedy) graph algorithms
- ❑ Greedy algorithms
- ❑ Divide and conquer
- ❑ Dynamic programming
- ❑ Network flows
- ❑ Linear programming

Solution

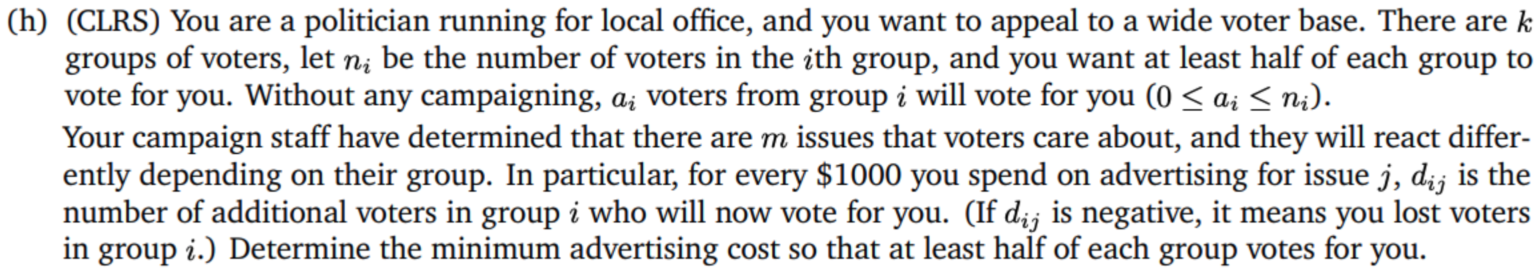
Problem 2 - Technique toolbox activity



(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.

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

Problem 2 - Technique toolbox activity

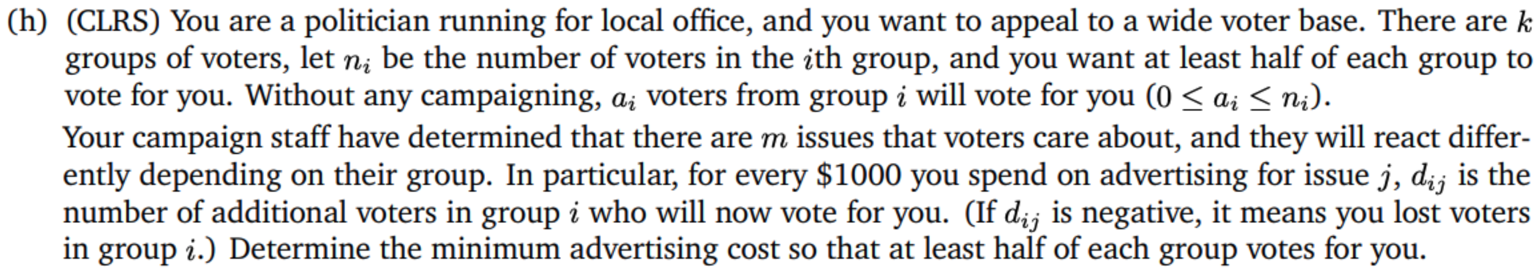

(h) (CLRS) You are a politician running for local office, and you want to appeal to a wide voter base. There are k groups of voters, let n_i be the number of voters in the ith group, and you want at least half of each group to vote for you. Without any campaigning, a_i voters from group i will vote for you $(0 \leq a_i \leq n_i)$.

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.

- ❑ Stable matching
- ❑ Graph traversal algorithms
- ❑ Weighted (greedy) graph algorithms
- ❑ Greedy algorithms
- ❑ Divide and conquer
- ❑ Dynamic programming
- ❑ Network flows
- ❑ Linear programming

Solution

Problem 2 - Technique toolbox activity



(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 ($0 \leq a_i \leq n_i$).

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.

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

Problem 2 - Technique toolbox activity

(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.

- ❑ Stable matching
- ❑ Graph traversal algorithms
- ❑ Weighted (greedy) graph algorithms
- ❑ Greedy algorithms
- ❑ Divide and conquer
- ❑ Dynamic programming
- ❑ Network flows
- ❑ Linear programming

Solution

Problem 2 - Technique toolbox activity

(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.

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.

Problem 2 - Technique toolbox activity

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

- ❑ Stable matching
- ❑ Graph traversal algorithms
- ❑ Weighted (greedy) graph algorithms
- ❑ Greedy algorithms
- ❑ Divide and conquer
- ❑ Dynamic programming
- ❑ Network flows
- ❑ Linear programming

Solution

Problem 2 - Technique toolbox activity

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

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.

Problem 2 - Technique toolbox activity

(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.

- ☐ Stable matching
- ☐ Graph traversal algorithms
- ☐ Weighted (greedy) graph algorithms
- ☐ Greedy algorithms
- ☐ Divide and conquer
- ☐ Dynamic programming
- ☐ Network flows
- ☐ Linear programming

Solution

Problem 2 - Technique toolbox activity

(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.

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

Problem 2 - Technique toolbox activity

(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`$[i]$.

- ☐ Stable matching
- ☐ Graph traversal algorithms
- ☐ Weighted (greedy) graph algorithms
- ☐ Greedy algorithms
- ☐ Divide and conquer
- ☐ Dynamic programming
- ☐ Network flows
- ☐ Linear programming

Solution

Problem 2 - Technique toolbox activity

(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`$[i]$.

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.

Problem 2 - Technique toolbox activity

(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".

- ☐ Stable matching
- ☐ Graph traversal algorithms
- ☐ Weighted (greedy) graph algorithms
- ☐ Greedy algorithms
- ☐ Divide and conquer
- ☐ Dynamic programming
- ☐ Network flows
- ☐ Linear programming

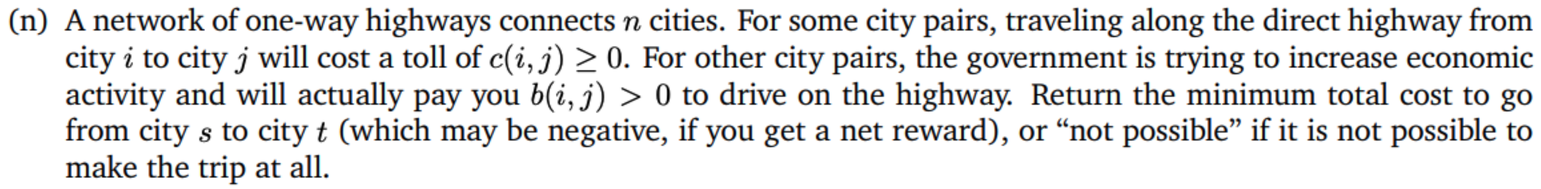
Solution

Problem 2 - Technique toolbox activity

(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".

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.

Problem 2 - Technique toolbox activity


(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.

- ❑ Stable matching
- ❑ Graph traversal algorithms
- ❑ Weighted (greedy) graph algorithms
- ❑ Greedy algorithms
- ❑ Divide and conquer
- ❑ Dynamic programming
- ❑ Network flows
- ❑ Linear programming

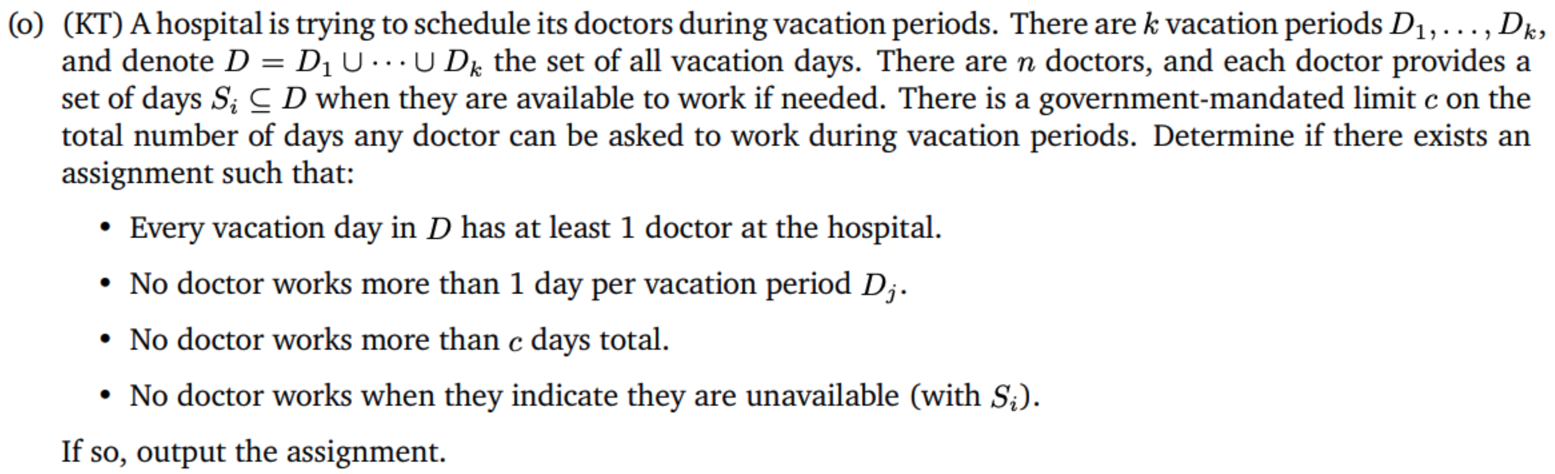
Solution

Problem 2 - Technique toolbox activity

(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.

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

Problem 2 - Technique toolbox activity



(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:

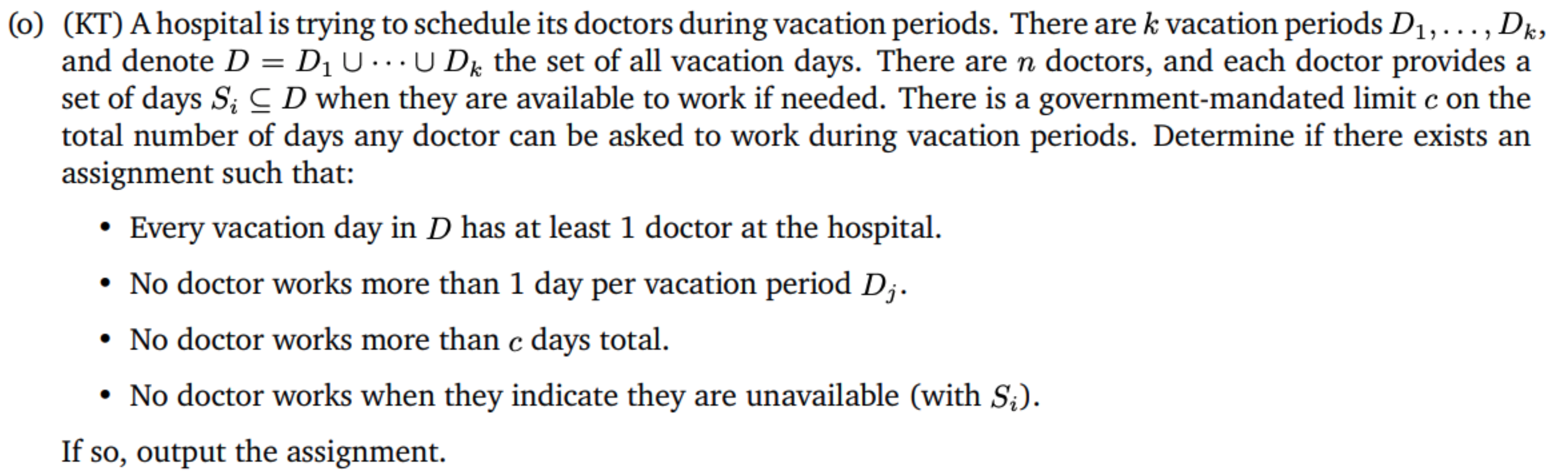
- Every vacation day in D has at least 1 doctor at the hospital.
- No doctor works more than 1 day per vacation period D_j.
- No doctor works more than c days total.
- No doctor works when they indicate they are unavailable (with S_i).

If so, output the assignment.

❑ Stable matching
❑ Graph traversal algorithms
❑ Weighted (greedy) graph algorithms
❑ Greedy algorithms
❑ Divide and conquer
❑ Dynamic programming
❑ Network flows
❑ Linear programming

Solution

Problem 2 - Technique toolbox activity



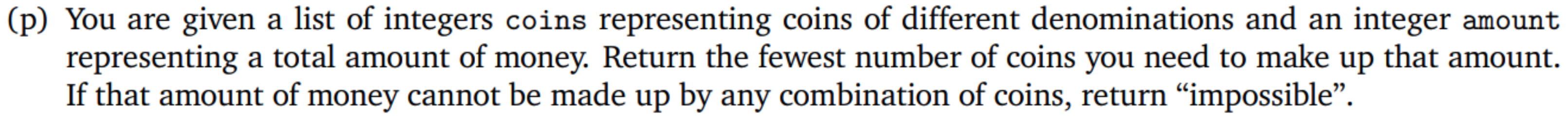
(o) (KT) A hospital is trying to schedule its doctors during vacation periods. There are k vacation periods $D_1, \ldots, D_k$, and denote $D = D_1 \cup \cdots \cup D_k$ the set of all vacation days. There are n doctors, and each doctor provides a set of days $S_i \subseteq D$ when they are available 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.

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.

Problem 2 - Technique toolbox activity



(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”.

- ❑ Stable matching
- ❑ Graph traversal algorithms
- ❑ Weighted (greedy) graph algorithms
- ❑ Greedy algorithms
- ❑ Divide and conquer
- ❑ Dynamic programming
- ❑ Network flows
- ❑ Linear programming

Solution

Problem 2 - Technique toolbox activity

(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".

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