1.1 Asymptotics

Below is a list of pairs of functions. For each pair f(•) and g(•), decide whether f is just O(g), f is justhttps://canvas.northwestern.edu/equation_images/%255COmega(g), or if f ishttps://canvas.northwestern.edu/equation_images/%255CTheta%2520(g). You do not need to prove your answer.

a. f(n) = n - 134, g(n) = n + 12

b. f(n) = n1.6, g(n) = n2

c. f(n) = n + log n2, g(n) = n + (log n)2

d. f(n) = n log n, g(n) = 3n log 19n1.1

e. f(n) = n log n, g(n) = 3n1.1 log 19n1.1

f. f(n) = n1.001, g(n) = n (log n)1000

g. f(n) = n3 / log n, g(n) = n2 (log n)1000

h. f(n) =https://canvas.northwestern.edu/equation_images/%255Csqrt%257Bn%257D, g(n) = 4log7n

i. f(n) = n2n, g(n) = 5n

j. f(n) = 2n, g(n) = 2n+1

1.2 More Asymptotics

Prove the following claims, or disprove them with a counterexample:

a. If https://canvas.northwestern.edu/equation_images/f%255Cleft%2528n%255Cright%2529 is https://canvas.northwestern.edu/equation_images/%255COmega%255Cleft%2528g%255Cleft%2528n%255Cright%2529%255Cright%2529 then \log\:f\left(n\right) is \Omega\left(\log\:g\left(n\right)\right).

b. If https://canvas.northwestern.edu/equation_images/f%255Cleft%2528n%255Cright%2529 is https://canvas.northwestern.edu/equation_images/%255CTheta%255Cleft%2528g%255Cright%2529 then https://canvas.northwestern.edu/equation_images/2%255E%257Bf%255Cleft%2528n%255Cright%2529%257D is https://canvas.northwestern.edu/equation_images/%255CTheta%255Cleft%25282%255E%257Bg%255Cleft%2528n%255Cright%2529%257D%255Cright%2529.

c. If https://canvas.northwestern.edu/equation_images/f%255Cleft%2528n%255Cright%2529 is https://canvas.northwestern.edu/equation_images/%255CTheta%255Cleft%2528g%255Cleft%2528n%255Cright%2529%255Cright%2529then https://canvas.northwestern.edu/equation_images/%255Csqrt%257Bf%255Cleft%2528n%255Cright%2529%257D is https://canvas.northwestern.edu/equation_images/%255CTheta%255Cleft%2528%255Csqrt%257Bg%255Cleft%2528n%255Cright%2529%257D%255Cright%2529.

d. Ifhttps://canvas.northwestern.edu/equation_images/f%255Cleft%2528n%255Cright%2529 is https://canvas.northwestern.edu/equation_images/%255CTheta%255Cleft%2528g%255Cright%2529then https://canvas.northwestern.edu/equation_images/f%255Cleft%2528n%255Cright%2529-%255Clog%2520n is https://canvas.northwestern.edu/equation_images/%255CTheta%255Cleft%2528g%255Cleft%2528n%255Cright%2529-%255Clog%2520n%255Cright%2529.

1.3 ARM WRESTLINGI

Little-known fact about Northwestern academic policy: when class sizes are too large, the professor is allowed to make students arm-wrestle for grades. Instead of assigning and grading homeworks, the professor of a certain large class of yours has, over the course of the quarter, made every pair of students in the class arm-wrestle. Every arm-wrestling match has a winner and a loser, so the outcomes of the quarter's hard work can be expressed as a directed graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529 with a single directed edge between each pair of vertices, pointing into the winner from that pair.

At the end of the quarter, there's the matter of grades. Given an outcome graph https://canvas.northwestern.edu/equation_images/G from a quarter of arm wrestling, how do you decide who gets what grade? The professor decides on the following method: find an ordering of the entire set of students such that Student 1 beats Student 2, Student 2 beats Student 3, etc., and then assign a curve to that ordering. We will call such an ordering "total." Prove that given an arbitrary outcome graph https://canvas.northwestern.edu/equation_images/G, there always exists a total ordering of the students.

1.4 Modern Tourism

You work for the popular ridesharing company Ubor. You've been put on one of their new projects - a tour service (which will be called TUbor, naturally). Because the people who use Ubor are all Very Busy, the gimmick for this service is that all Ubor tours go from scenic start location https://canvas.northwestern.edu/equation_images/s to the scenic dropoff location https://canvas.northwestern.edu/equation_images/t with as short a route as possible.

You've been asked to solve problems of the following type for each city where TUbor will launch: given the road network for the city (a directed graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529), a scenic start location https://canvas.northwestern.edu/equation_images/s%255Cin%2520V, and a scenic dropoff location https://canvas.northwestern.edu/equation_images/t%255Cin%2520V, give an algorithm which computes the number of distinct TUbor tours starting at https://canvas.northwestern.edu/equation_images/s and ending at https://canvas.northwestern.edu/equation_images/t. Assume all roads are the same length. Your algorithm should run in linear time in the number of edges and vertices in https://canvas.northwestern.edu/equation_images/G.

1.5 The Maize Runner

Your friend owns a farm, and every autumn, sets up a corn maze. The problem is, while your friend really loves designing mazes, they aren't so smart. You're acting as quality control - specifically, you need to check if their maze has any loops, as such a maze would be too difficult. Before your friend actually builds the maze, they've drawn it out as an undirected graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529. You need to figure out if https://canvas.northwestern.edu/equation_images/G has any cycles.

Fortunately, you took a great algorithms course at Northwestern, and already know how to do a breadth-first search of https://canvas.northwestern.edu/equation_images/G. In fact, you have an implementation of BFS which takes a vertex https://canvas.northwestern.edu/equation_images/r%255Cin%2520V  and returns the search tree rooted at https://canvas.northwestern.edu/equation_images/r for the graph, which includes each vertex's depth in the tree, along with a pointer to its parent, all in O\left(\left|V'\right|+\left|E'\right|\right) time, where V' and E' are the vertex and edge sets of https://canvas.northwestern.edu/equation_images/r's component in https://canvas.northwestern.edu/equation_images/G. Using this implementation as a black box, design an algorithm to solve this problem which runs in time https://canvas.northwestern.edu/equation_images/O%255Cleft%2528%255Cleft%257CV%255Cright%257C%2B%255Cleft%257CE%255Cright%257C%255Cright%2529.

2.1 Myth Boosters

You’ve been hired as an intern for the Discovery Channel hit series Myth Boosters. The premise of the show is that every week, alleged “experts” offer evidence that some completely implausible urban legend is actually true. Your boss comes to you one day and says that she’d like an algorithm that finds secret messages hidden in texts.  
  
Specifically, you are given as input a text, which is a sequence https://canvas.northwestern.edu/equation_images/W of words https://canvas.northwestern.edu/equation_images/w_1%252C%255Cldots%252Cw_n, and a target message, which is also a sequence https://canvas.northwestern.edu/equation_images/T of words https://canvas.northwestern.edu/equation_images/t_1%252C%255Cldots%252Ct_m, with https://canvas.northwestern.edu/equation_images/m%255Cleq%2520n. The target message https://canvas.northwestern.edu/equation_images/T is a valid secret message for the input text https://canvas.northwestern.edu/equation_images/W if you can delete words from the https://canvas.northwestern.edu/equation_images/W to get https://canvas.northwestern.edu/equation_images/T, without changing the ordering of any words. Your job is to design an algorithm which identifies whether or not https://canvas.northwestern.edu/equation_images/T is a valid secret message for https://canvas.northwestern.edu/equation_images/W. Your algorithm should run in https://canvas.northwestern.edu/equation_images/O%2528n%2529 time.

2.2 Snow, Men

You manage the company in charge of shoveling the sidewalks of various midwestern universities. Each university is represented as a graph https://canvas.northwestern.edu/equation_images/G%253D%2528V%252CE%2529 where each vertex is a location on campus and each edge represents a sidewalk between locations. When it snows, your job is to shovel some subset https://canvas.northwestern.edu/equation_images/S%255Csubseteq%2520E of the sidewalks, such that https://canvas.northwestern.edu/equation_images/%2528V%252CS%2529 is connected.  
  
To help you, you have a bunch (i.e. a number greater than https://canvas.northwestern.edu/equation_images/%257CE%257C) of strapping young male employees. Each man can shovel one sidewalk per day (union regulations). Each sidewalk https://canvas.northwestern.edu/equation_images/e takes an associated amount of time https://canvas.northwestern.edu/equation_images/t%2528e%2529 to shovel. You want to be fair to your employees in the following sense: you want the set https://canvas.northwestern.edu/equation_images/S you have them shovel to minimize the maximum time any employee spends shoveling, subject to the connectivity constraint already mentioned.  
  
You spend a while thinking about this, before your righthand snow man suggests that any minimum spanning tree of https://canvas.northwestern.edu/equation_images/G will be fair and connected. Is your assistant crazy? Prove or disprove their claim.

2.3 MST Trivia

Prove or disprove the following claims about MSTs on an arbitrary connected graph with arbitrary positive edge weights.

a. For any two vertices, the shortest path between them must be part of some minimum spanning tree.

b. If a graph has a unique heaviest edge, it cannot be part of any minimum spanning tree.

c. If any cycle in the graph contains a unique lightest edge, then this edge must be part of every MST.

d. Every (possibly non-unique) lightest edge of a graph is a member of some MST.

e. If a cycle in the graph contains a unique heaviest edge, then this edge cannot be part of any MST.

3.1 Matroids

In class we showed that for any set systemhttps://canvas.northwestern.edu/equation_images/%255Cleft%2528E%252CI%255Cright%2529 that is a matroid, for any values of the elements, the greedy-by-value algorithm finds a feasible set https://canvas.northwestern.edu/equation_images/S in I\subseteq2^E with the maximum total value.  Show that this is also "only if", i.e., that if the set system is not a matroid then there exists values of the elements such that greedy by value does not find the feasible set with the maximum total value.  Note: a set system is not a matroid because it fails either the subset property, the augmentation property, or both.

3.2 Travel Time

Consider the problem of finding the shortest path from your house to your place of work. Here, a useful notion of "shortest" is not the distance, but the travel time. For instance, traveling 10 miles on the freeway is probably faster than traveling 7 miles on residential streets. Unfortunately, the time it takes to traverse an edge in a road network could depend on the time of day - driving a mile on Chicago Avenue at 1am and 5pm are very different. The latter will take far longer.

To model this, we will use a directed graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529. For each edge https://canvas.northwestern.edu/equation_images/e%253D%255Cleft%2528u%252Cv%255Cright%2529 let https://canvas.northwestern.edu/equation_images/f_e%255Cleft%2528t%255Cright%2529 be the time you reach https://canvas.northwestern.edu/equation_images/vgiven you leave from https://canvas.northwestern.edu/equation_images/u at time https://canvas.northwestern.edu/equation_images/t. For example, if it takes you one unit of time to get from https://canvas.northwestern.edu/equation_images/u to https://canvas.northwestern.edu/equation_images/v if you leave at time 2, then https://canvas.northwestern.edu/equation_images/f_e%255Cleft%25282%255Cright%2529%253D2%2B1%253D3. We will make two assumptions about https://canvas.northwestern.edu/equation_images/f_e:

1. https://canvas.northwestern.edu/equation_images/f_e%255Cleft%2528t%255Cright%2529%255Cge%2520t. In other words, no traveling backwards in time.
2. If t'>t, then f_e\left(t'\right)\ge f_e\left(t\right). In other words, you won't arrive earlier by waiting.

The algorithmic problem is as follows: Given a graph https://canvas.northwestern.edu/equation_images/G, a start vertex https://canvas.northwestern.edu/equation_images/s%255Cin%2520Vs&Element;V, and travel time functions https://canvas.northwestern.edu/equation_images/f_e%255Cleft%2528%255Ccdot%255Cright%2529, compute the minimum travel time from https://canvas.northwestern.edu/equation_images/s to every other vertex, starting at time 0.

4.1 Peak Interest

You have a friend who is obsessed with designing weird new data structures. She needs your help writing a method for her latest concoction, which she is calling a "peak tree." A peak tree consists of https://canvas.northwestern.edu/equation_images/n%253D2%255Ed-1 nodes, which she promises are arranged in a complete binary tree of depth https://canvas.northwestern.edu/equation_images/d (with a root node, labeled r). Each node v has three internal variables: v.left is the left child of v, v.right is the right child of v (each of these is null if v is a leaf), and v.value, which is an integer, and which is distinct for each node in the tree.

The method she needs is findPeak(), and should identify a vertex v in the peak tree such that v has at least the value of its parent (if any) and both its children. Give a divide-and-conquer (ie recursive) implementation of findPeak(). It should run in https://canvas.northwestern.edu/equation_images/O%255Cleft%2528%255Clog%2520n%255Cright%2529 time.

4.2 Twinzies

Your data structure-obsessed buddy has another request. She's putting together what she's calling a "twin array." It is two arrays. The two arrays, labeled A and B, are each of length https://canvas.northwestern.edu/equation_images/n, and store integers. Your friend promises that all the integers are distinct, both within and across the two arrays, and that both of the arrays will be sorted lowest to highest.

She needs your help implementing the median() function, which returns the https://canvas.northwestern.edu/equation_images/nth highest value from among all the elements in both lists. Give a divide-and-conquer implementation of median(). It should run in https://canvas.northwestern.edu/equation_images/O%255Cleft%2528%255Clog%2520n%255Cright%2529 time.

4.3 How Many Sums?

Let https://canvas.northwestern.edu/equation_images/A and https://canvas.northwestern.edu/equation_images/B be subsets of https://canvas.northwestern.edu/equation_images/%255Cleft%255C%257B1%252C%255Cldots%252Cn%255Cright%255C%257D. Consider the set https://canvas.northwestern.edu/equation_images/C%255Csubset%255Cleft%255C%257B1%252C%255Cldots%252C2n%255Cright%255C%257D defined as C=\left\{a+b\:\:\mid\:a\in A\wedge b\in B\right\}. We would like to count for each https://canvas.northwestern.edu/equation_images/c%255Cin%2520C the number of ways it is obtained from elements https://canvas.northwestern.edu/equation_images/a%255Cin%2520A and https://canvas.northwestern.edu/equation_images/b%255Cin%2520B. E.g., for https://canvas.northwestern.edu/equation_images/A%253D%255Cleft%255C%257B1%252C3%255Cright%255C%257D and https://canvas.northwestern.edu/equation_images/B%253D%255Cleft%255C%257B2%252C4%255Cright%255C%257D the set https://canvas.northwestern.edu/equation_images/C%253D%255Cleft%255C%257B3%252C5%252C7%255Cright%255C%257D and https://canvas.northwestern.edu/equation_images/3 and https://canvas.northwestern.edu/equation_images/7 are obtained one way, while https://canvas.northwestern.edu/equation_images/5 is obtained two ways.

a. Give and analyze a natural brute-force algorithm. Your runtime should be in terms of https://canvas.northwestern.edu/equation_images/n.

b. Give and analyze an algorithm with asymptotically better runtime than brute force, in terms of https://canvas.northwestern.edu/equation_images/n.

In both cases your algorithm should output a list of elements of https://canvas.northwestern.edu/equation_images/C with their respective counts.

5.1 Boxing

You're the newly elected president of the Northwestern University Boxing Club, and as part of your presidential duties, you're tasked with bringing boxes to the first meeting. You have https://canvas.northwestern.edu/equation_images/n boxes that you want to bring, but unfortunately, you have only one set of hands with which to carry them all. Your solution is to nest as many of them inside each other as possible, and then carry the nested boxes to the meeting. Each box https://canvas.northwestern.edu/equation_images/i has a height https://canvas.northwestern.edu/equation_images/h_i, a width https://canvas.northwestern.edu/equation_images/w_i, and a length https://canvas.northwestern.edu/equation_images/l_i, and you can only nest box https://canvas.northwestern.edu/equation_images/jinside box https://canvas.northwestern.edu/equation_images/i if https://canvas.northwestern.edu/equation_images/h_i%253Eh_j, https://canvas.northwestern.edu/equation_images/w_i%253Ew_j, and https://canvas.northwestern.edu/equation_images/l_i%253El_j. (In other words, no rotating boxes.) Give an algorithm which determines the maximum number of boxes you can bring to the meeting. Your algorithm's runtime should be polynomial in https://canvas.northwestern.edu/equation_images/n. Put your algorithm in the four-part format discussed in class. To prove correctness, you need only argue the correctness of your recurrence.

Additionally, code up the iterative version of your DP algorithm, and verify that it is correct on several test cases. Print the input/output for your test cases and upload it along with your writeup.

5.2 Tiny Times Tables

Your precocious 6-year-old cousin just learned how multiplication works, and couldn't be more excited. She's so excited, in fact, that she's decided to try to multiply letters. You asked your cousin "Why stop there? Why not use any symbol you want?" Your cousin's eyes get wide and she runs off to her room. Several hours later, she runs downstairs and presents you with a list of https://canvas.northwestern.edu/equation_images/n symbols https://canvas.northwestern.edu/equation_images/S%253D%255Cleft%255C%257Bs_1%252C%255Cldots%252Cs_n%255Cright%255C%257D, along with an https://canvas.northwestern.edu/equation_images/n%255Ctimes%2520n table https://canvas.northwestern.edu/equation_images/T, where entry https://canvas.northwestern.edu/equation_images/%255Cleft%2528i%252Cj%255Cright%2529 is the product of https://canvas.northwestern.edu/equation_images/s_i and https://canvas.northwestern.edu/equation_images/s_j, which is also a symbol in https://canvas.northwestern.edu/equation_images/S.

You pat her on the head, and then send her off to play with her new system. She starts stringing together symbols, and realizes that in her new system, the order of multiplications matters. You explain to her how to use parentheses to specify the order of operations. Her eyes go wide again, and off she runs once more. Again she comes downstairs several hours later, this time with a puzzle. She gives you a string of symbols https://canvas.northwestern.edu/equation_images/p%253Dp_1%255Cldots%2520p_m, all elements of https://canvas.northwestern.edu/equation_images/S, and a target symbol https://canvas.northwestern.edu/equation_images/t%255Cin%2520S. She wants to know: is there a way of putting parentheses into https://canvas.northwestern.edu/equation_images/p such that when the multiplications are performed according to https://canvas.northwestern.edu/equation_images/T, in the parenthesized order, you get https://canvas.northwestern.edu/equation_images/t? Give an algorithm to solve your cousin's puzzle which is polynomial in https://canvas.northwestern.edu/equation_images/m and https://canvas.northwestern.edu/equation_images/n. Put your algorithm in the four-part format discussed in class. To prove correctness, you need only argue the correctness of your recurrence.

Additionally, code up the iterative version of your DP algorithm, and verify that it is correct on several test cases. Print the input/output for your test cases and upload it along with your writeup.

6.1 Reading And/Or Weeping

You've decided to spend your free time this month reading Kleinberg and Tardos cover to cover. You've got https://canvas.northwestern.edu/equation_images/D days this month, and on day https://canvas.northwestern.edu/equation_images/i, you have https://canvas.northwestern.edu/equation_images/m_i free minutes to read. There are https://canvas.northwestern.edu/equation_images/n sections in the book, and section https://canvas.northwestern.edu/equation_images/j takes https://canvas.northwestern.edu/equation_images/t_j minutes to read. You want to read all the sections in order, and want to have read all the sections by the end of the month. Your task is to schedule your reading for the month.

If you read sections https://canvas.northwestern.edu/equation_images/j%252C%255Cldots%252Cj%2Bk on day https://canvas.northwestern.edu/equation_images/i, there are two problems you might face. First, you might leave yourself with free time to do other things. Let https://canvas.northwestern.edu/equation_images/T_i%253D%255Csum_%257B%255Cell%253Dj%257D%255E%257Bj%2Bk%257Dt_%257B%255Cell%257D be the time spent reading on day https://canvas.northwestern.edu/equation_images/i. If you have free time, then https://canvas.northwestern.edu/equation_images/T_i%253Cm_i. Let https://canvas.northwestern.edu/equation_images/F_i%253D%255Cmax%255Cleft%2528m_i-T_i%252C0%255Cright%2529. That is, https://canvas.northwestern.edu/equation_images/F_i is your free time on day https://canvas.northwestern.edu/equation_images/i, or https://canvas.northwestern.edu/equation_images/0 if you have no free time. Having free time is a disaster, so you decide that part of your goal should be to minimize https://canvas.northwestern.edu/equation_images/%255Csum_%257Bi%253D1%257D%255EDF_i%255E4. The second, less bad thing that could happen is that you could decide to read too much on day https://canvas.northwestern.edu/equation_images/i and have to skip a few hours of sleep. That is, https://canvas.northwestern.edu/equation_images/T_i%253Em_i. Let https://canvas.northwestern.edu/equation_images/S_i be the total sleep you lose from reading. Then https://canvas.northwestern.edu/equation_images/S_i%253D%255Cmax%255Cleft%2528T_i-m_i%252C0%255Cright%2529. You don't want to lose too much sleep, so you decide that the other part of your objective should be to minimize https://canvas.northwestern.edu/equation_images/%255Csum_%257Bi%253D1%257D%255EDS_i. Your full objective, then, is to schedule your reading to minimize your total unhappiness, https://canvas.northwestern.edu/equation_images/%255Csum_%257Bi%253D1%257D%255EDF_i%255E4%2B%255Csum_%257Bi%253D1%257D%255EDS_i.

As an example, consider three days with https://canvas.northwestern.edu/equation_images/m_1%253D5m1=5, https://canvas.northwestern.edu/equation_images/m_2%253D4m2=4, and https://canvas.northwestern.edu/equation_images/m_3%253D6m3=6 and four sections, with https://canvas.northwestern.edu/equation_images/t_1%253D3t1=3, https://canvas.northwestern.edu/equation_images/t_2%253D1t2=1, https://canvas.northwestern.edu/equation_images/t_3%253D5t3=5, and https://canvas.northwestern.edu/equation_images/t_4%253D4t4=4. An optimal solution will read sections 1 and 2 on day 1, section 3 on day 2, and section 4 on day 3. For this solution, https://canvas.northwestern.edu/equation_images/T_1%253D4T1=4, https://canvas.northwestern.edu/equation_images/T_2%253D5T2=5, and https://canvas.northwestern.edu/equation_images/T_3%253D4T3=4, https://canvas.northwestern.edu/equation_images/F_1%253D1F1=1, https://canvas.northwestern.edu/equation_images/F_2%253D0F2=0, and https://canvas.northwestern.edu/equation_images/F_3%253D2F3=2, and https://canvas.northwestern.edu/equation_images/S_1%253D0S1=0, https://canvas.northwestern.edu/equation_images/S_2%253D1S2=1, and https://canvas.northwestern.edu/equation_images/S_3%253D0S3=0. It follows that your total unhappiness is https://canvas.northwestern.edu/equation_images/1%255E4%2B2%255E4%2B1%253D1814+24+1=18.

Give a dynamic programming algorithm which computes the total unhappiness of the optimal reading schedule. Include all four parts of a good writeup. You do not need to prove correctness, except in justifying your recurrence. Additionally, program your iterative algorithm, and run several test cases. Include a printout of your inputs and outputs with your writeup.

6.2 FiEncournstters

You've graduated from Northwestern, and done what any respectable NU grad would do: become a conspiracy theorist/amateur xenobiologist. The truth you're currently trying to reveal to the world is simple: there are two kinds of aliens out there, and they're both trying to get in touch with us. These aliens' preferred means of communication are radio signals, which they have been transmitting to us for decades. We've just been too blind to see! Or rather, we haven't had the right algorithmic tools.

You know from hundreds of hours huddled in your basement wearing a tinfoil hat that the two alien races each have a signal that they consistently transmit. Race A has a string https://canvas.northwestern.edu/equation_images/a_1%255Cldots%2520a_p of bits that they transmit, and race B has a different string, https://canvas.northwestern.edu/equation_images/b_1%255Cldots%2520b_q, and they've both been transmitting these non-stop. The reason we haven't been able to make out these signals is because they've gotten mixed. Instead of receiving https://canvas.northwestern.edu/equation_images/a_1%255Cldots%2520a_pa_1%255Cldots%2520a_pa_1%255Cldots%2520a_p%255Cldots and https://canvas.northwestern.edu/equation_images/b_1%255Cldots%2520b_qb_1%255Cldots%2520b_qb_1%255Cldots%2520b_q%255Cldots, we receive jumbles of the two strings' bits, which might look something like:https://canvas.northwestern.edu/equation_images/a_1a_2b_1b_2a_3a_4a_5b_3a_6b_4%255Cldots. The bits are all there, and they're in order - the two strings are just mixed up.

Your task is to develop a dynamic programming algorithm to show the world the truth about aliens. As input, you take the two binary message strings https://canvas.northwestern.edu/equation_images/a%253Da_1%255Cldots%2520a_p and https://canvas.northwestern.edu/equation_images/b%253Db_1%255Cldots%2520b_q, and a candidate string https://canvas.northwestern.edu/equation_images/c%253Dc_1%255Cldots%2520c_n. You must determine if there are natural numbers https://canvas.northwestern.edu/equation_images/k_1 and https://canvas.northwestern.edu/equation_images/k_2 such that if you repeat https://canvas.northwestern.edu/equation_images/a https://canvas.northwestern.edu/equation_images/k_1 times (denoted https://canvas.northwestern.edu/equation_images/a%255E%257Bk_1%257D) and https://canvas.northwestern.edu/equation_images/b https://canvas.northwestern.edu/equation_images/k_2 times (https://canvas.northwestern.edu/equation_images/b%255E%257Bk_2%257D), that there is a way of combining the bits of https://canvas.northwestern.edu/equation_images/a%255E%257Bk_1%257D and https://canvas.northwestern.edu/equation_images/b%255E%257Bk_2%257D, but preserving the order of the bits, which yields https://canvas.northwestern.edu/equation_images/c. Your algorithm's runtime should be polynomial in https://canvas.northwestern.edu/equation_images/p,https://canvas.northwestern.edu/equation_images/q, and https://canvas.northwestern.edu/equation_images/n.

As an example, assume https://canvas.northwestern.edu/equation_images/a%253D0011a=0011 and https://canvas.northwestern.edu/equation_images/b%253D1010b=1010. The string https://canvas.northwestern.edu/equation_images/c%253D010110100011c=010110100011 would be a "Yes" instance to this problem, as if we set https://canvas.northwestern.edu/equation_images/k_1%253D2k1=2 and https://canvas.northwestern.edu/equation_images/k_2%253D1k2=1, then https://canvas.northwestern.edu/equation_images/c_1c_3c_4c_5c_8c_9c_%257B11%257Dc_%257B12%257D%253Da%255E%257Bk_1%257D%253D00110011c1c3c4c5c8c9c11c12=ak1=00110011 and https://canvas.northwestern.edu/equation_images/c_2c_6c_7c_%257B10%257D%253Db%255E%257Bk_2%257D%253D1010c2c6c7c10=bk2=1010. Meanwhile, the https://canvas.northwestern.edu/equation_images/c%253D000101101011c=000101101011 is a "No" instance. To convince yourself of this fact, notice that https://canvas.northwestern.edu/equation_images/cc starts with three 0s, which cannot be the case for any jumbling of https://canvas.northwestern.edu/equation_images/aa and https://canvas.northwestern.edu/equation_images/bb.

Include all four parts of a good writeup. You do not need to prove correctness, except in justifying your recurrence. Additionally, program your iterative algorithm, and run several test cases. Include a printout of your inputs and outputs with your writeup.

6.3 Cat People

It's election season for the National Association of Students With Too Many Cats. The Association is comprised of the students at https://canvas.northwestern.edu/equation_images/n universities. There are two competing factions, the Ferals and Domestics, each of which have a candidate vying for president of the organization. University https://canvas.northwestern.edu/equation_images/i's membership is comprised of https://canvas.northwestern.edu/equation_images/f_i Ferals and https://canvas.northwestern.edu/equation_images/d_i Domestics, for a total of https://canvas.northwestern.edu/equation_images/m_i students overall. Every election, the bylaws require the https://canvas.northwestern.edu/equation_images/n universities to be split into an A and B division of https://canvas.northwestern.edu/equation_images/%255Cfrac%257Bn%257D%257B2%257D universities each. Each division will pool its membership and select a candidate by majority-rules (with ties not counting as a win). If one of the two candidates wins both divisions, then that candidate is president for the year. Otherwise, the previous president retains office for another the year.

The stakes are high, and you're a rabid Domestic. Fortunately, you're also the one in charge of dividing up the universities into the two divisions. Your algorithmic task is to take as input the list of https://canvas.northwestern.edu/equation_images/f_is and https://canvas.northwestern.edu/equation_images/d_is, and output whether or not it is possible to divide the universities evenly into two groups so that the Domestic candidate wins both divisions. Give a dynamic programming algorithm for the task which runs in time polynomial in https://canvas.northwestern.edu/equation_images/n and https://canvas.northwestern.edu/equation_images/M%253D%255Csum_im_i.

As an example, assume there are four universities, with https://canvas.northwestern.edu/equation_images/f_1%253D7f1=7, https://canvas.northwestern.edu/equation_images/f_2%253D2f2=2, https://canvas.northwestern.edu/equation_images/f_3%253D3f3=3, and https://canvas.northwestern.edu/equation_images/f_4%253D8f4=8, and https://canvas.northwestern.edu/equation_images/d_1%253D3d1=3, https://canvas.northwestern.edu/equation_images/d_2%253D5d2=5, https://canvas.northwestern.edu/equation_images/d_3%253D8d3=8, and https://canvas.northwestern.edu/equation_images/d_4%253D6d4=6. In this case, it is possible for the Domestics to win, if we select universities 1 and 3 to be in the A division and universities 2 and 4 to be in the B division. In each division, the Domstics win out by a narrow margin of 1. Clearly, if we decrease the number of domestics at any university by 1, then the Domestics will no longer be able to win both divisions, no matter how the divisions are selected.

Include all four parts of a good writeup. You do not need to prove correctness, except in justifying your recurrence. Additionally, program your iterative algorithm, and run several test cases. Include a printout of your inputs and outputs with your writeup.

7.1 Zoo Tycoon

You're in charge of feeding your zoo's large herbivorous mammals. In particular, you have https://canvas.northwestern.edu/equation_images/n different animals, and https://canvas.northwestern.edu/equation_images/m different kinds of food. Animal https://canvas.northwestern.edu/equation_images/i must eat https://canvas.northwestern.edu/equation_images/F_i tons of food each month. To complicate things, your animals are a little picky about what food they'll eat. Animal https://canvas.northwestern.edu/equation_images/i has a set https://canvas.northwestern.edu/equation_images/S_i%255Csubseteq%255Cleft%255C%257B1%252C%255Cldots%252Cm%255Cright%255C%257D of different foods that it likes to eat, and each month, will only be willing to eat https://canvas.northwestern.edu/equation_images/D_i tons of food not in https://canvas.northwestern.edu/equation_images/S_i (in total across all foods not in https://canvas.northwestern.edu/equation_images/S_i). For each food https://canvas.northwestern.edu/equation_images/j, you have https://canvas.northwestern.edu/equation_images/T_j tons of that food. You want to know: do you have enough food (of the right kinds) to feed all the animals for the month? Give a polynomial-time algorithm to answer this question. (Note: If you use a network flow algorithm as a black box, you do not need to re-explain how to solve network flow.)

7.2 Disrupting Shipping

One intuitive story for the max flow problem is that the input graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529G=(V,E) is a rail network, the capacities are shipping capacities along rail lines, and the algorithm designer is interested in maximizing the volume of shipments from some source city https://canvas.northwestern.edu/equation_images/ss to the sink city https://canvas.northwestern.edu/equation_images/tt.

You're interested in ruining this particular algorithm designer's plans. Specifically, you're given a rail network https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529G=(V,E) with*unit capacities*, a source https://canvas.northwestern.edu/equation_images/ss, and a sink https://canvas.northwestern.edu/equation_images/tt. You have enough explosives to destroy https://canvas.northwestern.edu/equation_images/dd rails. Your goal is to reduce the volume of shipping between https://canvas.northwestern.edu/equation_images/ss and https://canvas.northwestern.edu/equation_images/tt (which will be computed via max flow after you're done blowing stuff up) by as much as possible. The question is: which rails should you destroy? Give a polynomial-time algorithm to answer this question.

7.3 No Child Left Behind

You're in charge of a youth mentorship program. The program matches young orphans to high-school-aged mentors, so that they have some good influences to look up to in life. A noble cause. You have https://canvas.northwestern.edu/equation_images/n orphans to assign mentors this year, and this pool will be the same through the entire year. The problem you face is that the pool of mentors doesn't stay stable over the course of the year. Each week, the set of available mentors changes, and you need to update the mentorship assignments to keep up with these fluctuations.

Every week, you need to solve the following problem: as inputs, you're given:

* https://canvas.northwestern.edu/equation_images/n orphans.
* https://canvas.northwestern.edu/equation_images/m potential mentors. Each potential mentor https://canvas.northwestern.edu/equation_images/j has a list https://canvas.northwestern.edu/equation_images/L_j of orphans whose interests are compatible with theirs. Mentor https://canvas.northwestern.edu/equation_images/j can only mentor orphans in https://canvas.northwestern.edu/equation_images/L_j. Drawing edges between mentors and compatible orphans forms a *compatibility graph*.
* A pre-existing matching https://canvas.northwestern.edu/equation_images/M of orphans to mentors which respects the compatibility graph. This is the set of currently existing mentorship pairings. Note that https://canvas.northwestern.edu/equation_images/M need not be a maximum matching.

You want see if there is a larger matching than https://canvas.northwestern.edu/equation_images/M in this week's compatibility graph. You're not heartless, though, so any orphan who is currently matched by https://canvas.northwestern.edu/equation_images/M needs to stay matched but not necessarily to the same mentor.

a. Give a polynomial-time algorithm which computes the maximum matching in the compatibility graph, subject to the constraint that each orphan currently matched by https://canvas.northwestern.edu/equation_images/M stays matched to a mentor (but maybe not to the same mentor).

b. Will the matching computed by your algorithm be a maximum matching (that is, without the constraint that currently matched orphans stay matched)? Prove your answer.

8.1 Secret Santa Cycles

Until this last Thursday, you were super excited to run your department's secret Santa gift exchange. You had a brilliant plan: you'd take the friendship graph for your department, compute a Hamiltonian cycle, and then each department member would buy a gift for their successor in the cycle.

But then you went to algorithms and were told that Hamiltonian Cycle is NP-Hard.

You've thought some more, and come up with a new idea. Rather than finding one cycle containing everyone in the department, you're going to allow there to be multiple cycles in your gift exchange. Each member of the department will be in exactly one of these cycles. The gift exchange will then work as before - each member will buy a gift for their successor. First, you need to figure out if this is possible. Formally, the decision problem you need to solve is:

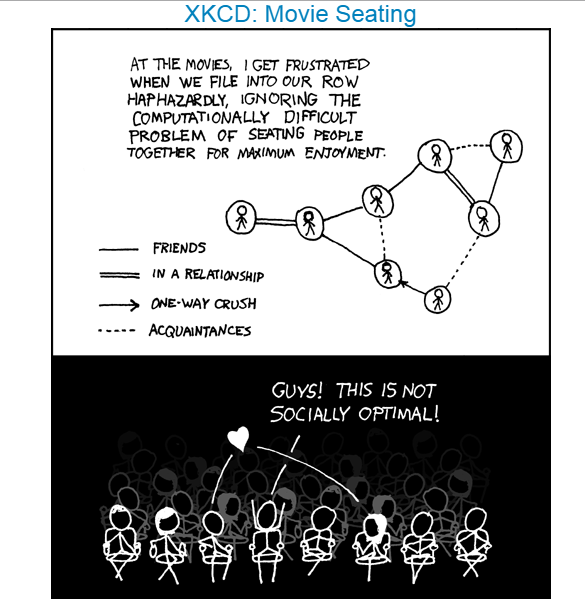
**Input:** A directed graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529 representing friendships in your department.

**Output:**Whether there exist cycles https://canvas.northwestern.edu/equation_images/C_1%252C%255Cldots%252CC_k in https://canvas.northwestern.edu/equation_images/G, each of length at least 2, such that each vertex https://canvas.northwestern.edu/equation_images/v%255Cin%2520V is a member of exactly one https://canvas.northwestern.edu/equation_images/C_j.

Give a polynomial-time algorithm to solve this problem or show that it is NP-complete.  (This week all NP-hardness reductions should be from 3-SAT or Hamiltonian Cycle.)

8.2 Movie Seating

http://xkcd.com/173/



The new Star Wars movie has finally come out! To celebrate, you and https://canvas.northwestern.edu/equation_images/n friends are all going to see the film together. You know each of these https://canvas.northwestern.edu/equation_images/n people, but they have a variety of relationships to each other. Some are enemies (and don't want to sit together), some are in love (and do want to sit together), some don't even know each other (and don't care about sitting together), and so on. This would be complicated enough if seating was completely unconstrained. To make things harder, though, at the theater, everyone must sit in a row. Your job is to untangle this complicated social web, and put it in a row of movie theater seats.

More formally, you're faced with a*complete* directed graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529, with https://canvas.northwestern.edu/equation_images/n vertices, each representing a friend. Their relationships are encoded in labels on the edges. There are five kinds of directed relationships:

* Love (L)
* Friendship (F)
* Acquaintance (A)
* Strangers (S)
* Hatred (H)

All of the above relationships are asymmetric - for example, it is possible for person https://canvas.northwestern.edu/equation_images/u to love person https://canvas.northwestern.edu/equation_images/v while person https://canvas.northwestern.edu/equation_images/v hates person https://canvas.northwestern.edu/equation_images/u.

A seating arrangement is an ordering of the vertices in https://canvas.northwestern.edu/equation_images/V. The relationships above determine the social enjoyment of an arrangement, which we'll measure in "points." The points of an arrangement are given by:

* 3 points for each person sitting next to a person they love.
* 2 points for each person sitting next to a friend.
* 1 point for each person sitting next to an acquaintance.
* -3 points for each person sitting next to someone they hate.

As a concrete example, say you have 3 people, https://canvas.northwestern.edu/equation_images/a, https://canvas.northwestern.edu/equation_images/b, and https://canvas.northwestern.edu/equation_images/c. The relationships are:

* https://canvas.northwestern.edu/equation_images/a hates https://canvas.northwestern.edu/equation_images/b and loves https://canvas.northwestern.edu/equation_images/c.
* https://canvas.northwestern.edu/equation_images/b is strangers with https://canvas.northwestern.edu/equation_images/a and acquaintances with https://canvas.northwestern.edu/equation_images/c.
* https://canvas.northwestern.edu/equation_images/c is friends with both https://canvas.northwestern.edu/equation_images/a and https://canvas.northwestern.edu/equation_images/b.

Given these relationships, the ordering https://canvas.northwestern.edu/equation_images/abc would have point value 0, since https://canvas.northwestern.edu/equation_images/a hates https://canvas.northwestern.edu/equation_images/b (-3), https://canvas.northwestern.edu/equation_images/b is acquaintances with https://canvas.northwestern.edu/equation_images/c(+1), and https://canvas.northwestern.edu/equation_images/c considers  https://canvas.northwestern.edu/equation_images/b a friend (+2) (with all the other relationships being stranger relationships).

The decision version of the problem you face is the following:

**Input:** A complete directed graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529 with relationship labels on each edge, and an integer https://canvas.northwestern.edu/equation_images/k.

**Output:** Whether or not there is a seating arrangement with point value at least https://canvas.northwestern.edu/equation_images/k.

Give a polynomial-time algorithm to solve this problem or show that it is NP-complete.  (This week all NP-hardness reductions should be from 3-SAT or Hamiltonian Cycle.)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hints:

1. The problem is NP-hard. Those of who who came up with an algorithm will either win a Clay Millenium Prize for resolving P vs. NP or should rethink your answer.

2. The most reasonable reductions we know of for the problem are multi-stage. My recommendations:

• Start with directed Hamiltonian Cycle rather than 3-SAT.

• You’re probably best served with the following chained reductions:

-Directed Hamiltonian Cycle -> Undirected Hamiltonian Cycle -> Movie Seating

-Directed Hamiltonian Cycle -> Directed Hamiltonian Path -> Undirected Hamiltonian Path -> Movie Seating.

There might be other ways, and pat on the back if you find one, but the above are the easiest I can come up with.

3. Regarding grading: whatever you do, the very last reduction (e.g. Undirected HC/P to Movie Seating) will receive the largest emphasis, pointswise. If you want partial credit, nailing this reduction down is your best bet, even if you don’t get the other, intermediate reductions in the chain.  *Extra credit: if you turn in the full reduction by the original due date.*

8.3 More Cat People

You did it! The Domestics won the last election, and you've been appointed coordinator of the annual conference of the National Association of Students With Too Many Cats. There are https://canvas.northwestern.edu/equation_images/n universities in the association, and for the conference, you need to select a delegate from each one. Each university https://canvas.northwestern.edu/equation_images/i has a list of potential delegates https://canvas.northwestern.edu/equation_images/D_i for you to choose from. To complicate things, the Association's members are notoriously catty, and many pairs of potential delegates are in long-standing fights so bitter that inviting both members to the conference would be a grave mistake. You have a list https://canvas.northwestern.edu/equation_images/L containing pairs of delegates who are fighting in such a way. Formally, your problem is:

**Input:** https://canvas.northwestern.edu/equation_images/n delegate lists https://canvas.northwestern.edu/equation_images/D_1%252C%255Cldots%252CD_n, and a list https://canvas.northwestern.edu/equation_images/L containing https://canvas.northwestern.edu/equation_images/m pairs of delegates in https://canvas.northwestern.edu/equation_images/%255Cbigcup_%257Bi%253D1%257D%255EnD_i.

**Output:** Whether it is possible to invite exactly one delegate from each of the https://canvas.northwestern.edu/equation_images/n delegate lists in such a way that you don't invite any pair of delegates from https://canvas.northwestern.edu/equation_images/L. (Note: it is permissible to invite just one member of a pair in https://canvas.northwestern.edu/equation_images/L. Just not both.)

Give a polynomial-time algorithm to solve this problem or show that it is NP-complete.  (This week all NP-hardness reductions should be from 3-SAT or Hamiltonian Cycle.)

9.1 No Free Lunch

You just took a part-time job planning lunches for a daycare for children with serious food allergies. Your boss explains the daily lunch routine to you: the lunch planner (you) chooses the foods (among the https://canvas.northwestern.edu/equation_images/m types in the pantry) to serve and puts them out on a table. Each of the https://canvas.northwestern.edu/equation_images/n children then takes some of the food they like most and eats it. The challenge you face is that the food a child will choose need not be the food that particular child is able to eat safely. You need to choose a menu which minimizes the overall medical damage the children do to themselves. Formally, the problem is the following:

**Input:**

* For each child in https://canvas.northwestern.edu/equation_images/i%255Cin%255Cleft%255C%257B1%252C%255Cldots%252Cn%255Cright%255C%257D, a strict preference ordering https://canvas.northwestern.edu/equation_images/%253E_i over the foods https://canvas.northwestern.edu/equation_images/j%255Cin%255Cleft%255C%257B1%252C%255Cldots%252Cm%255Cright%255C%257D. [In other words, for two types of food https://canvas.northwestern.edu/equation_images/j and j', j>_ij' means "child https://canvas.northwestern.edu/equation_images/i prefers food https://canvas.northwestern.edu/equation_images/j to food j'.]
* For each child https://canvas.northwestern.edu/equation_images/i and each food https://canvas.northwestern.edu/equation_images/j, a nonnegative cost https://canvas.northwestern.edu/equation_images/c_%257Bi%252Cj%257D. This is the expected medical bill if child https://canvas.northwestern.edu/equation_images/i eats food https://canvas.northwestern.edu/equation_images/j.
* A nonnegative number https://canvas.northwestern.edu/equation_images/k.

**Process:** You choose a nonempty subset https://canvas.northwestern.edu/equation_images/S%255Csubseteq%255Cleft%255C%257B1%252C%255Cldots%252Cm%255Cright%255C%257D of foods to serve. Once you have chosen https://canvas.northwestern.edu/equation_images/S:

* Each child https://canvas.northwestern.edu/equation_images/i eats the food from https://canvas.northwestern.edu/equation_images/S which they most prefer according to https://canvas.northwestern.edu/equation_images/%253E_i. Denote this food by https://canvas.northwestern.edu/equation_images/%255Cmax%255Cleft%2528S%252Ci%255Cright%2529. [Note: multiple children can eat the same type of food.]
* Each child incurs an expected medical bill of https://canvas.northwestern.edu/equation_images/c_%257Bi%252C%255Cmax%255Cleft%2528S%252Ci%255Cright%2529%257D.

**Question:** Is there a nonempty subset https://canvas.northwestern.edu/equation_images/S%255Csubseteq%255Cleft%255C%257B1%252C%255Cldots%252Cm%255Cright%255C%257D such that https://canvas.northwestern.edu/equation_images/%255Csum_ic_%257Bi%252C%255Cmax%255Cleft%2528S%252Ci%255Cright%2529%257D%255Cle%2520k?

As an example, assume there are two foods and two children. Assume child 1 prefers food 1 to food 2, and child 2 prefers food 2 to food 1. If we had https://canvas.northwestern.edu/equation_images/c_%257B11%257D%255Cle%2520c_%257B12%257D and https://canvas.northwestern.edu/equation_images/c_%257B22%257D%255Cle%2520c_%257B21%257D, then it would be optimal to choose https://canvas.northwestern.edu/equation_images/S%253D%255Cleft%255C%257B1%252C2%255Cright%255C%257D - the children will each choose the safest food of their own volition. If either https://canvas.northwestern.edu/equation_images/c_%257B11%257D%253Ec_%257B12%257D or https://canvas.northwestern.edu/equation_images/c_%257B22%257D%253Ec_%257B21%257D, though, then we would need to compare the costs of our three options: https://canvas.northwestern.edu/equation_images/c_%257B11%257D%2Bc_%257B22%257D from choosing https://canvas.northwestern.edu/equation_images/S%253D%255Cleft%255C%257B1%252C2%255Cright%255C%257D, https://canvas.northwestern.edu/equation_images/c_%257B11%257D%2Bc_%257B21%257D from choosing https://canvas.northwestern.edu/equation_images/S%253D%255Cleft%255C%257B1%255Cright%255C%257D, https://canvas.northwestern.edu/equation_images/c_%257B12%257D%2Bc_%257B22%257D from choosing https://canvas.northwestern.edu/equation_images/S%253D%255Cleft%255C%257B2%255Cright%255C%257D.

Give a polynomial-time algorithm to solve this problem, or prove that it is NP-complete. If you use dynamic programming, include all four parts of a good write-up. You do not need to prove correctness for DP, except in justifying your recurrence, and you do not need to program your iterative algorithm. If you prove that it is NP-complete, you may reduce from any of the following problems: Independent Set, Vertex/Set Cover, 3-D Matching, Graph Coloring, Hamiltonian Cycle/Path, TSP, Subset-Sum, 3-SAT.

[Note: if you find yourself actually running such a daycare, please do not structure it like this. It's obviously a horrible system.]

9.2 A Perfect Party

You're organizing a party at your house to celebrate famous comedian Richard Pryor's birthday (December 1, but you already knew that). You want this party to be the best one possible. You've bought all sorts of tasty snacks and fancy decorations - you just need to decide who to invite. Because this is supposed to be the best party ever, you decide that everyone you invite must be friends with every other person you invite. The decision version of the algorithmic problem you want to solve is therefore the following:

**Input:** The graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529 representing your social network, and a positive integer https://canvas.northwestern.edu/equation_images/k. Each vertex is a friend of yours, and https://canvas.northwestern.edu/equation_images/%255Cleft%255C%257Bu%252Cv%255Cright%255C%257D%255Cin%2520E if and only if https://canvas.northwestern.edu/equation_images/u and https://canvas.northwestern.edu/equation_images/v are friends.

**Output:**Whether or not there is a guest list https://canvas.northwestern.edu/equation_images/S%255Csubseteq%2520V of size at least https://canvas.northwestern.edu/equation_images/k such that every member of the guest list is friends with every other member.

Give a polynomial-time algorithm to solve this problem, or prove that it is NP-complete. If you use dynamic programming, include all four parts of a good write-up. You do not need to prove correctness for DP, except in justifying your recurrence, and you do not need to program your iterative algorithm. If you prove that it is NP-complete, you may reduce from any of the following problems: Independent Set, Vertex/Set Cover, 3-D Matching, Graph Coloring, Hamiltonian Cycle/Path, TSP, Subset-Sum, 3-SAT.

9.3 The Future of Trade

Upon graduating from Northwestern University, you've taken a job at an ecommerce startup called Bartr. They run a market for - you guessed it - the trade of goods and services without the use of money. People list the stuff they have and a numerical value for both their stuff and other peoples' stuff. The Bartr trading platform then looks for pairwise trades that produce a net increase in total value. Your job is to design the trade-finding algorithm. The formal details of your problem:

**Input:**

* A set https://canvas.northwestern.edu/equation_images/%255Cleft%255C%257B1%252C%255Cldots%252Cn%255Cright%255C%257D{1,&ltdot;,n} of users.
* For each user https://canvas.northwestern.edu/equation_images/ii, a set https://canvas.northwestern.edu/equation_images/A_iAi of items which are available for trade. Let https://canvas.northwestern.edu/equation_images/A%253D%255Ccup_iA_iA=∪iAi be the set of all items, with https://canvas.northwestern.edu/equation_images/%255Cleft%257CA%255Cright%257C%253Dm|A|=m.
* For each user https://canvas.northwestern.edu/equation_images/ii and item https://canvas.northwestern.edu/equation_images/jj, a value https://canvas.northwestern.edu/equation_images/v_%257Bij%257Dvij of user https://canvas.northwestern.edu/equation_images/ii for item https://canvas.northwestern.edu/equation_images/jj. Assume all value are positive integers.

**Question:** Does there exist a pair of users who would gain from trading? Formally, is there some https://canvas.northwestern.edu/equation_images/ii and i' and https://canvas.northwestern.edu/equation_images/S_i%255Csubseteq%2520A_iSi&subseteq;Ai and S_{i'}\subseteq A_{i'} such that \sum_{j\in S_i}v_{i'j}>\sum_{j\in S_{i'}}v_{i'j} and \sum_{j\in S_{i'}}v_{ij}>\sum_{j\in S_i}v_{ij}?

Give a polynomial-time algorithm [in not just https://canvas.northwestern.edu/equation_images/nn and https://canvas.northwestern.edu/equation_images/mm but the number of bits needed to represent thehttps://canvas.northwestern.edu/equation_images/v_%257Bij%257Dvijs] to solve this problem, or prove that this task is NP-complete. If you use dynamic programming, include all four parts of a good write-up. You do not need to prove correctness for DP, except in justifying your recurrence, and you do not need to program your iterative algorithm. If you prove that it is NP-complete, you may reduce from any of the following problems: Independent Set, Vertex/Set Cover, 3-D Matching, Graph Coloring, Hamiltonian Cycle/Path, TSP, Subset-Sum, 3-SAT.

10.1 Trees

You've bought a plot of land which is 20 meters wide and https://canvas.northwestern.edu/equation_images/5n meters long, and you've decided to plant an orchard. To help you do this, you've divided the land up into 5\times5 squares, and tested the soil quality in each square. For each https://canvas.northwestern.edu/equation_images/i%255Cin%255Cleft%255C%257B1%252C%255Cldots%252C4%255Cright%255C%257D and https://canvas.northwestern.edu/equation_images/j%255Cin%255Cleft%255C%257B1%252C%255Cldots%252Cn%255Cright%255C%257D, https://canvas.northwestern.edu/equation_images/q_%257Bij%257D is a positive real number indicating the soil quality in square https://canvas.northwestern.edu/equation_images/%255Cleft%2528i%252Cj%255Cright%2529. Note that the https://canvas.northwestern.edu/equation_images/q_%257Bij%257Ds form a https://canvas.northwestern.edu/equation_images/4%255Ctimes%2520n matrix. Given this data, you now need to decide where to plant trees.

You may plant a tree in any location, but it must be that no two trees can be neighbors in the horizontal direction or vertical direction. In other words, if you plant a tree at location https://canvas.northwestern.edu/equation_images/%255Cleft%2528i%252Cj%255Cright%2529, you cannot plant a tree at \left(i\pm1,j\right) or \left(i,j\pm1\right) (assuming those indices exist). Your goal is to plant trees to maximize the total value of https://canvas.northwestern.edu/equation_images/q_%257Bij%257D, summed over the squares you choose. Formally:

**Input:**

* A https://canvas.northwestern.edu/equation_images/4%255Ctimes%2520n matrix https://canvas.northwestern.edu/equation_images/Q, with entries https://canvas.northwestern.edu/equation_images/q_%257Bij%257D.
* A nonnegative number https://canvas.northwestern.edu/equation_images/k.

**Output:** True if and only if there is a subset https://canvas.northwestern.edu/equation_images/Sof the entries of https://canvas.northwestern.edu/equation_images/q_%257Bij%257D such that:

* if https://canvas.northwestern.edu/equation_images/%255Cleft%2528i%252Cj%255Cright%2529%255Cin%2520S, then  \left(i\pm1,j\right)\notin S and \left(i,j\pm1\right)\notin S.
* https://canvas.northwestern.edu/equation_images/%255Csum_%257Bi%255Cin%2520S%257Dq_%257Bij%257D%255Cge%2520k.

Give a polynomial time algorithm to solve this problem using dynamic programming.  Include all four parts of a good write-up. You do not need to prove correctness for DP, except in justifying your recurrence, and you do not need to program your iterative algorithm.

This assignment must be done individually.  Students submitting this extra credit assignment will also be required to do peer reviews as detailed on the page: [Extra Credit Assignments](https://canvas.northwestern.edu/courses/29575/pages/extra-credit-assignments).

There will be two extra credit assignments, one for *Dynamic Programming* and one for *NP-completeness* reductions.  The problems must be done individually.  This grade is out of 20 points.  Your grade for this problem and be used to replace your grade for a similar problem on a previous homework (dynamic programming for dynamic programming and NP-completeness for NP-completeness).  To receive full credit you must both complete the exercise and four peer reviews.

**Grading details:**  You will be graded on both the correctness of your solution, the effectiveness by which you communicate it (as always), and the accuracy of your peer reviews.  You will be assigned four peer reviews.  Each peer review will have a rubric and you are expected to mark the rubric as the graders would mark it.  Your peer review grade will be based on how close your assessment is to the grade the assignment receives for each item of the rubric.  Small differences are ok and will not lose any points.  Your least accurate peer review will be dropped.

10.2 Flood warning, revisited

We saw the following problem in lab: you're trying to design a flood evacuation plan for a county. You have a map of the county, modeled by a directed graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529, where vertices are locations and edges are roads. A subset https://canvas.northwestern.edu/equation_images/X%255Csubseteq%2520V of the locations are towns, full of people who will need to be evacuated. Another subset https://canvas.northwestern.edu/equation_images/S%255Csubseteq%2520V are safe zones - areas of high elevation where civilians can hide from flood waters. Your goal was for each https://canvas.northwestern.edu/equation_images/x_i%255Cin%2520X, find a safe zone https://canvas.northwestern.edu/equation_images/s_i%255Cin%2520S and a path https://canvas.northwestern.edu/equation_images/P_i from https://canvas.northwestern.edu/equation_images/x_i to https://canvas.northwestern.edu/equation_images/s_i in https://canvas.northwestern.edu/equation_images/G such that no two such paths share an edge.

You took your model for the situation, along with the algorithm we found in lab, to an expert in disaster planning, and they told you that you're thinking of things wrong. First of all, every city https://canvas.northwestern.edu/equation_images/x_i%255Cin%2520X already has a legally prescribed flood evacuation zone https://canvas.northwestern.edu/equation_images/s_i%255Cin%2520S. This is law, and you can't pick how the cities are matched to evacuation zones like you thought. Second, the expert tells you, you don't need to be worried about congestion on roads - the real problem is that at intersections, you'll need to worry about gridlock. In other words, the real problems aren't the roads (edges), its the locations (vertices). You think over the expert's advice, and come up with the following, new formulation:

**Input:** A graph https://canvas.northwestern.edu/equation_images/G%253D%255Cleft%2528V%252CE%255Cright%2529 representing the map for the county, a subset https://canvas.northwestern.edu/equation_images/X%255Csubseteq%2520V of vertices representing towns, and for each town https://canvas.northwestern.edu/equation_images/x_i%255Cin%2520X, an assigned safe zone https://canvas.northwestern.edu/equation_images/s_isi.

**Output:** Whether or not it is possible to find a path https://canvas.northwestern.edu/equation_images/P_i from https://canvas.northwestern.edu/equation_images/x_i to https://canvas.northwestern.edu/equation_images/s_i for each https://canvas.northwestern.edu/equation_images/i such that no two paths https://canvas.northwestern.edu/equation_images/P_i and https://canvas.northwestern.edu/equation_images/P_j share a vertex.

Prove that this problem is NP-complete.  You may reduce from any of the following problems: Independent Set, Vertex/Set Cover, 3-D Matching, Graph Coloring, Hamiltonian Cycle/Path, TSP, Subset-Sum, 3-SAT.

This assignment must be done individually.  It can be used to replace the grade for any previously assigned NP-completeness reduction.  Students submitting this extra credit assignment will also be required to do peer reviews as detailed on the page: [Extra Credit Assignments](https://canvas.northwestern.edu/courses/29575/pages/extra-credit-assignments).

10.3 spirit of giving

You are in charge of optimizing your dorm's holiday toy drive.  You’ve collected https://canvas.northwestern.edu/equation_images/m toys which you plan to give out to the https://canvas.northwestern.edu/equation_images/n children at your local children's hospital.  There is good news here and bad news.  The good news is that you really have collected a lot of toys: there are more than ten toys per child on average, i.e., https://canvas.northwestern.edu/equation_images/m%253E10n. The bad news is that none of the toys are really that exceptional: the toys have values https://canvas.northwestern.edu/equation_images/v_1%252C%255Cldots%252Cv_m  and none of the toys' values are more than twice the average, i.e., for all https://canvas.northwestern.edu/equation_images/i, https://canvas.northwestern.edu/equation_images/v_i%255Cle%255Cfrac%257B2%257D%257Bm%257D%255Csum_%257Bj%253D1%257D%255Emv_j.  [Note: values don't depend on the child - each kid values the toys the same way.]

Your job is to figure out which children get which toys.  In other words, you choose a partitioning S_1,\ldots,\:S_n of the toys where child https://canvas.northwestern.edu/equation_images/i gets set https://canvas.northwestern.edu/equation_images/S_i. Given such a partition, there is inevitably going to be a least happy child who has the minimum value set, i.e., \min_i\:\sum_{j\in S_i}v_j. You would like to allocate the toys to make this least happy child as happy as possible.

There's more bad news, however. You thought for a while, and realized that this problem is NP-hard.  But! You’ve just learned all about approximation algorithms - this seems like a great opportunity to design one. You decide to try the standard approach:

a) Identify an upper bound on the total value of the toys given to the least happy child by the optimal partitioning.  
  
b) Give a polynomial time algorithm that does a pretty good job of partitioning the objects.  
  
c) Prove that your algorithm is a constant approximation by comparing its performance to the upper bound.  Be sure to identify the smallest constant you can.

This assignment must be done individually.  It can be used to replace the grade for any previously assigned problem.  Students submitting this extra credit assignment will also be required to do peer reviews as detailed on the page: [Extra Credit Assignments](https://canvas.northwestern.edu/courses/29575/pages/extra-credit-assignments).