Homework 1:

Question 1: (Goodrich and Tamassia) In the *art hall guarding* problem we are give a line *L*that represents a long hallway in an art gallery. We are also given a set $X = \{x_0, x_1, \ldots, x_{n-1}\}$ of real numbers that specify the positions of paintings in the hallway. Suppse that a single guard can protect all the paintings within distance of at most 1 of his or her position (on both sides). Design an algorithm for finding a placement of guards that uses the minimum number of guards to guard all the paintings with positions in *X*. Give the running time and prove that your algorithm is correct.

Question 2: (Kleinberg and Tardos) You are managing a team of expert hackers, and each week you have to choose a job for them to undertake. Suppose there are *n* weeks and for week *i*you have to choose between assigning a low stress job with payout l_i or a high stress job with payout h_i. The catch is that if you assign a high stress job in week *i* then you needed to assign no jobs in week *i-1* so that your team has time to adequately prepare for the job. Your task is: given a set of payouts l_1,l_2,...,l_n and h_1,h_2,...,h_n, find a schedule that where you choose between doing the low stress job, the high stress job, and no job in each week to get a payout of h_k, l_k, or 0, respectively, such that you produce the schedule with maximum possible payout. Provide your algorithm, prove your algorithm is correct, and state and justify its running time.

Question 3: (Goodrich and Tamassia) Given a set *P* of *n* teams, a ***round robin tournament*** is a collection of games where every team plays each other team exactly once. Design an efficient algorithm for constructing the schedule of games of a round robin tournament for the *n* teams. Prove your algorithm correctly creates the round robin tournament, and state and justify its running time.

Question 4: (Kleinberg and Tardos) You are interested in analyzing some hard to obtain data from two separate databases. Each database contains *n* numeric values - so there are *2n* values total - and you may assume that no two values are the same. You'd like to determine the median of this set of *2n* values, which we will define here to ve the *n*th smallest value. However, the only way you can access these values is thorugh queries to the database. In a single query, you can specify the value *k* to one of the two databases, and the chosen database will return the *k*th smallest value that it contains. Since the queries are expensive, you would like to compute the median using as few queries as possible. Give your algorithm, prove your algorithm correct, and state and justify its running time.

Question 5: (Based on Kleinberg and Tardos) You are scheduling a telescope to view astronomical events. Suppose the telescope can move between -85$_o$; and +85$_o$ and the telescope can move 1$_o$ in either direction per minute. You are given a list of *n* celestial events. Each event has a time (in minutes since the sunset) and a location in degrees in the sky. If the telescope is pointing at that location at that minute, it can take a picture of the event. Otherwise it misses the event. Assume that taking a picture takes 1 minute and the telescope does not move during that time. Given the *n* events, the location and time of each event, and assuming the telescope begins the evening at position 0, give an efficient algorithm that finds a schedule that photographs the maximum number of events such that the last event in the list is photographed. Prove your algorithm is correct. State and justify the running time.

Question 6: (Kleinberg and Tardos) You are consulting for a shipping company that does a large amount of business shipping packages between New York and Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit *W* on the maximum amount of weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package *i* has a weight w_i. The trucking station is quite small so only one truck can be in the station at a time. Company policy requires that the boxes be shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after theirs make it to Boston faster. At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order received and whenever the next box does not fit, they send the truck on its way. But they also wonder if they might be using too many trucks. Maybe if they should wait for all the boxes to arrive that day, and because they know exactly the boxes and their weights, they can use fewer trucks. They will still pack the boxes in the order received so that the boxes arrive in New York in the proper order, but they might send off a truck with less weight so that later trucks can be better packed and thus use fewer trucks. Either find an algorithm that produces fewer trucks or prove that the original algorithm used by the company is optimal.

Homework 2:

Question 1: (Cormen, et al) An array A[1,...,n] contains all the integers from 0 to n except one. You are to find the missing integer. Here is the catch, your process is limited in that it cannot access an entire integer in a single query. Rather, the elements of A are represented in binary, and the only queries allowed are "fetch the j_{th} bit of A[i]". Each query takes constant time. Give an algorithm that quickly finds the missing integer of A. Prove your algorithm correct, and state and justify the running time. (Hint: brute force is O(n log n) because it is O(n) to run through the array and each entry has O(log n) bits.)

Question 2: You decide to start a pizzaria buisiness in your dorm. Because there is no dining area, this will be a pick-up only buisiness. You create an app where customers can choose their toppings and give a time when the pizza should be ready. You have a "ready on time or it is free" guarantee, and you coded your app so that it would not let the customers enter unreasonable times. You finish class, and on the way back to the dorm, you check your buisiness page, and to your shock you are already flooded with orders. Too many to possibly make on time. Here is your dilemma. You have *n* pizza orders. Each order has a value v_i which is the amount you are charging for that pizza and a pickup time t_i when it has to be done by or you are giving it away for free. You can only make one pizza at a time (the dorm has a small oven), and it takes the same amount of time to make each pizza. Design an algorithm that tells you the order you should make the pizzas in order to minimize the total value of the pizzas you are giving away for free.

Question 3: (Kleinberg and Tardos) You have price data for *n* consecutive days for a stock. The days are numbered $i = 1, 2, \ldots, n$, and for each day *i*, you have a price *p(i)* per share for the stock on that day. A *k-shot strategy* is a collection of *m* pairs of days $(b_1, s_1), \ldots, (b_m, s_m)$, where $0 <= m <= k$
and $1 <= b_1 < s_1 < b_2 < s_2 < \ldots < b_m < s_m <= n$.
These are non-overlapping intervals where you will buy 1000 shares of the stock on day b_i and sell

Your goal is to design an efficient algorithm that takes the *n* prices and the value *k* as input and determines the *k-shot strategy* with the maximum return. Prove your algorithm, and state and justify the running time. The algorithm you design should be polynomial in both *n* and *k* (*k* should not appear in the exponent of the running time).

Question 4: (Based on a problem by Kleinberg and Tardos) Spies from the Rebel Alliance have stolen the plans to the Death Star and secretly transmitted them somewhere. Maybe you can use these transmissions to uncover secret supporters of the rebels. The rebels sent the plans in a series of *n* separate transmissions (cause the Death Star is, like, big), and because the transmissions were done secretly, you don't know exactly when each transmission was sent, but you have a good guess. For the *i*th transmission, you know it was sent at time
t i ± ε iti±εi where ε iεi is the error factor. For some transmissions, you had the spy under surveillance so the ε iεi is rather small, but for some transmissions you only discovered about it after the fact so the ε iεi value in those transmissions is quite large.

Darth Vader recently captured a ship of Princess Leia. Searching the computer on board turned up no Death Star plans, but the ship suspiciously received *n* transmissions, and from the log you can see the exact times those transmissions arrived: $x_1, x_2, \dots x_n$.

You would love to blow up Leia's home planet of Alderon as punishment, but there is due process in the Empire. That means, before you blow up the planet, you must give strong evidence that she received the stolen transmissions. You will do that by matching each transmission time x_i arrival on Leia's ship with some transmission time t_j of the spies. Formally, x_i matches to t_j if
t j − ε j ≤ x i ≤ t j + ε jtj−εj≤xi≤tj+εj.

Give an algorithm (with proof and running time justification) that finds such a matching if one exists. We want the punishment to come quickly so the algorithm should be fast: $O(n_2)$ at worst.

Question 5:(Kleinberg and Tardos) You are consulting for an intelligence agency whose jobs consist monitoring and analyzing electronic signals coming from ships in coastal Atlantic waters. They want a fast algorithm to "untangle" a superposition of two signals. Specifically, they are picturing a situation in which each of two ships is emitting a short sequence of 0's and 1's over and over, and they want to make sure that the signal they're hearing is an *interleaving* of these two emissions, with nothing extra added in.
Given a string *x* consisting of 0s and 1s, we write x_k to denote *k* copies of *x* concatenated together.
We say that a string x' is a *repetition* of *x* if it is a prefix of x_k for some number *k*.
So x'=10110110110 is a repetition of *x* = 101.
We say that a string *s* is an *interleaving* of *x* and *y* if its symbols can be partitioned into two (not necessarily contiguous) sequences s' and s'' so that s' is a repetition of *x* and s'' is a repetition of *y*.
For example, if *x* = 101 and *y*= 00, then *s*=100010101 with s' = 101101 and s''=000.
Give an efficient algorithm that takes strings *s*, *x*, and *y* and decides if *s* is an interleaving of *x* and *y*.

Question 6: (Goodrich and Tamassia) Suppose you have a geometric description of the buildings of Manhattan and you would like to build a representation of the New York skyline. That is, suppose you are given a description of a set of rectangles, all of which have one of their sides on the *x*-axis, and you would like to build a representation of the union of all these rectangles. Formally, since each rectangle has a side on the *x*-axis, you can assume that you are given a set, $S = \{[a_1,b_1],[a_2,b_2],\dots,[a_n,b_n]\}$ of sub-intervals in the interval [0,1], with $0 <= a_i < b_i <= 1$, for $i = 1, 2,\dots, n$, such that there is an associated height, h_i, for each inteval $[a_i,b_i]$ in *S*. The ***skyline*** of *S* is defined to be a list of pairs $[(x_0,c_0),(x_i,c_1),(x_2,c_2),\dots,(x_m,c_m),(x_{m+1},0)]$, with $x_0 = 0$and $x_{m+1}=1$, and ordered by x_i values, such that, each subinterval, $[x_i,x_{i+1}]$, is the maximal subinterval that has a single highest interval, which is at height c_i, in *S*, containing $[x_i,x_{i+1}]$, for $i = 0, 1,\dots, m$. Design an O(*n* log *n*)-time algorithm for computing the skyline of *S*. Prove your algorithm correct and justify the running time.

Homework 3:

Question 1: You have n airplanes arriving at an airport. Only one plane can land at a time, there must be at least 3 minutes separating each plane landing. To make matters worse, storms prevented you from landing plans, and now there is a window of exactly $3(n-1)$ minutes available for landing planes before the storms return. Each plane i is arriving at the airport at time t_i, and each plane has enough fuel to circle the airport for x_i minutes before running out. You need the order to land the planes so that you can get all the planes down in the window and so that no plane runs out of fuel. If it is impossible to do so, your algorithm should report that. Solve this problem by *reducing* the problem to a class problem you have already solved, prove your reduction correct, and give and justify the runtime of the reduction.

Question 2: The *Longest Path* problem takes a graph G and an integer K and asks if there exists a path in G of at least K edges such that no vertex or edge repeats on the path. The *Hamiltonian Cycle* problem takes a graph G and asks if there exists a path in G that starts at some vertex, visits every vertex on G and returns to the starting vertex visiting each vertex exactly once and never repeating an edge. Give an efficient reduction of the *Hamiltonian Cycle* problem to the *Longest Path* problem. Give the running time of the reduction, and prove the reduction is correct.

Question 3: The *Subset Sum* problem is as follows: you are given n integers and a target T. Does there exist a subset of the numbers that sums to exactly T? The *Knapsack* problem is as follows: you are given a set of m items, each item i has a value v_i and a weight w_i, and you are given two numbers B and C. Does there exist a subset of the items in which the sum of the values is at least B and the sum of the weights is at most C? Give a reduction from an instance of the *Subset Sum* problem to the *Knapsack* problem that runs in polynomial time (polynomial on the bit-size of the *Subset Sum* problem instance). Give the running time of your reduction, and prove the reduction is correct.

Question 4: (Based on a problem of Goodrich and Tamassia) There is a major calamity approaching, and you must evacuate people from their current locations to safe spaces. Suppose there are n regions with people, and region i has N_i people that must be evacuated. Suppose there are m safe spaces and safe space j can shelter M_j people safely. The regions and safe spaces are scattered across a large distance, and you will only evacuate people from a region to a safe space if they can be transported there in under 24 hours. Suppose you have a list, for each region, which safe spaces are within a 24 hour distance. Reduce this problem to the network flow algorithm so that an efficient algorithm that evacuates the maximum number of people to safe spaces such that nobody has to travel more than 24 hours, and no safe space gets overcrowded. Prove your reduction correct and give the entire algorithm running time (reduction plus the cost of flow algorithm).

Question 5: (Tardos and Kleinberg) You have n cell towers located at points in a plane. Tower i is located at point t_i. You have n cell phones, and each cell phone j is located at point p_j. Each cell phone j also has a range d_j, and phone j can connect to tower i if the straightline distance between t_i and p_j is at most d_j. Suppose we require each cell phone to connect to a different cell tower. Suppose the owner of phone 1 is going for a drive due east, travelling continuously for distance z. As this cell phone moves, we may have to change the assignments of phones to towers in order to keep all the phones connected. Use a reduction to give a $O(n^3)$ algorithm that finds a sequence of assignments of phones to base stations so that all phones stay connected for the duration of the travel of phone 1. If such an assignment is not possible, the algorithm should report the point on the travelling phone's path where it is no longer possible to maintain full connectivity. Prove your algorithm is correct and justify its running time.

Question 6: (based on a problem by Tardos and Kleinberg) Viral marketing has become big news recently given the accusations about fake stories being propagated on the internet. Suppose you are given a graph that models a social network. The nodes are the actors in the network, and we have a directed edge from node a to node b if actor b "follows" a. Suppose for each node a there is a fraction $p(a)$ such that if more than $p(a)$ of the actors that a follows all post a particular rumor, then a will post that rumor as well. This is an iterative process so if a follows only b and b follows only c, and $p(a) = p(b) = 1/2$ (or any value at less than 1), then if $c$posts a rumor, b will post the rumor, and once b posts the rumor, a does as well. You are given the social network, the probabilities for each node/actor in the network, and two numbers k and b. Does there exist as set of k initial actors such that if each actor posts the same rumor, then after some number of iterations, we will have b actors all posting that rumor? Prove that this problem is NP-hard by doing a reduction with some known NP-hard problem.

Homework 4:

Question 1: (Based on Goodrich and Tammasia) We saw in lecture how to compute the shortest path from a source to a destination in a network. Now, what if the lengths of the edges change with time? For example, you live far from work, and you would like to find the fastest route from home to work. You have the entire road network as a graph *G* and you have found a website that has a function $f_{i,j}$, defined for each edge $e = (i,j)$ in *G*, that maps a time *t* to the expected length of time it takes to travel the edge *e*. (For example: $f_{i,j}(t) = k$ means that it takes *k* minutes to go from *i* to *j* along edge *e* at time *t* minutes after midnight.) You can assume the functions obey the laws of physics so someone leaving *i* later than you will not arrive at *j* earlier than you if you take the same edges. That is: if $t_1 < t_2$, then $f_{i,j}(t_2) + t_2 - t_1 > f_{i,j}(t_1)$. We are given *G*, two vertices labelled *h* and *w*, the edge functions, a starting time t_0 and a goal time *T*. We want to find a path from *h* to *w* that starts at time t_0 and arrives at *w* at or before time *T.*

Question 2: We saw in lecture how to compute the shortest path from a source to a destination in a network. Now, what if we need to minimize two different distance metrics at the same time? For example, suppose we need to ship goods from a warehouse in Topeka, Kansas to a plant in Astana, Kazakhstan. There is no single shipper that makes the entire trip so we will be using lots of transportation networks: trains, ships, trucks. There are lots of possible choices. Should we send the shipment to New York and across Europe or to San Francisco and across Asia? Which countries do we transport through, what companies do we use? We can model this as a graph where each vertex represents a place where we can change transportation options, and each edge *(a,b)* has both a *length* (how long does take to transport the goods from point *a* to point *b*) and a *cost* (how much did it cost to transport the goods on this option). We want to minimize both the total cost of the shipment and the total time.

To be precise, consider a graph *G* where each edge has both a *length* and a *cost*, where the *length* and *cost* are non-negative numbers. Two vertices are labelled *s* and *t*, and we have two numbers *L* and *C*. We want to find a path in *G* from *s* to *t* where the sum of the *lengths* of each edge on the path is at most *L* and the sum of the *costs* of each edge of the path is at most *C*.

Question 3: (Goodrich and Tammasia) Suppose you are given a telephone network, which is a graph *G* whose vertices represent switching stations, and whose edges represent communication bandwidth of a path in *G* is the bandwidth of the lowest-bandwidth edge of the path. We want an algorithm that, given a diagram and two switching centers *a* and *b*, will output the maximum bandwidth of a path between *a* and *b*. (You are not given the path. You must find the maximum over all possible paths from *a* to *b* in *G*.)

Question 4: (Goodrich and Tammasia) Suppose that CONTROL, a secrect U.S. government counterintelligence agency has built a communications network that links *n* stations spread across the world using *m* communication channels between pairs of stations. Suppose that the evil spy agency KAOS is able to eavesdrop on *k* of these channels, and that CONTROL knows which *k* have been compromised. CONTROL as a message *M* that it wants to send from its headquarters station *s* to field station *t*, and it wants to use a communications path that touches the fewest number of compromised edges.

Question 5: You are in charge of battling smugglers. You have cultivated a number of spies inside the smuggling operation, and from them you have mapped out the entire smuggling network. The smugglers collect the contraband from a large number of sites, and then they transport it to its destination through a network of "safe houses". Through your spies, you have identifi ed all the sites where the contraband is produced, the places where it is distributed, and all the safe houses used by the smugglers. You have also discovered the different routes that smuggler couriers use to transport the contraband from one safe house to another. Your goal is to disrupt the entire network by attacking the safe houses. The smugglers are resourceful, and they are capable of adjusting the network if one safe house is taken down. However, you believe that if you attack enough safe houses simultaneously so that you completely break the source sites from the distribution sites, the smugglers will be unable to recover. Attacking a safe house is dangerous so you do not want to attack too many at once or you will spread your resources too thin. Can you determine the fewest safe houses you can attack such that you break all routes from the source sites to the distribution sites? As an equivalent decision problem, given a number K, can you determine if it is possible to attack just K safe houses in order to disconnect the source sites from the distribution sites?

Question 6: (Kleinberg and Tardos) There's a natural intuition that two nodes that are far apart in a communication network - separated by many hops - have a more tenuous connection than two nodes that are close together. Suppose that an *n*-node, *m*-edge, undirected graph *G* contains two nodes *s* and *t* such that the distance between *s* and *t* is strictly greater than *n/2* edges. Show that there must exist some node *v* that is not *s* or *t* such that deleting *v* from *G* destroys all *s*-*t* paths. Give an algorithm with running time *O(m+n)* to find such a node *v*.

Homework 5:

problem. Is there an efficient algorithm? If so, provide the algorithm, a proof, and the running time with justification. Is there no known efficient algorithm? If so, provide the proof.

California deregulated its energy market. This "free" market works by matching producers of energy (power plants) with consumers of energy (a public utility, a factory). Each producer makes a *sell order* indicating the price for which the producer will sell energy (in cents per kilowatt) and the number of kilowatts the producer will sell. Each consumer makes a *buy order* indicating the price for which the consumer is willing to purchase energy and the amount the consumer wishes to purchase. If the price of a buy order is higher than or equal to the price on a sell order, a sale takes place and the consumer then sells *x* kilowatts to the consumer where *x* is the minimum of the amount the consumer will buy or the producer will sell.

The California Energy Board approaches you with the following problem. Consumers and producers want the ability to create a "secret" order. A secret order indicates that the producer (or consumer) is willing to sell (buy) all of its energy at a lower (higher) price, but only if they are guaranteed to sell (buy) a minimum amount. For example, suppose Pacific Gas and Electric (PG&E) currently has a sell order on the market stating that it is willing to sell up to 2,000,000 kilowatts at 4 cents per kilowatt. However, PG&E is also willing to sell at 3 cents a kilowatt if it can sell at least 1,750,000 kilowatts.

The problem California Energy Board has is that it wants the program to automatically match secret orders. That is, if there are enough sellers of energy who are selling at or below the secret buy price of an order (including any secret sell prices) and the total amount is at least the minimum set by the secret buy, then we have a match. However, if the match includes some secret sell orders, then we may also need to include some other orders on the buy side so that the secret sell gets its minimum.

Consider the following example.

Seller	**Price**	**Amount**	**Secret Price**	**Secret Min**
Bill's Wind Farm	3	250,000	---	---
PG&E	4	2,000,000	3	1,750,000

Buyer	**Price**	**Amount**	**Secret Price**	**Secret Min**
Ford Motors	2.5	1,500,000	3	1,000,000
Santa Clara Electric	2	1,500,000	3	1,200,000

If we did not have secret orders, no sales would occur in this example because the producers are asking for a higer price than the consumers want to spend. However, if we look at the secret price, then we have a sale.

PG&E will sell at 3 cents if it can sell at least 1,750,000 kW. This minimum can be met if we sell to both Ford Motors and Santa Clara Electric. However, Ford Motors and Santa Clara Electric will only buy at 3 cents if they can get, combined, 2,200,000. PG&E is only selling 2,000,000, but if we include Bill's Wind Farm, then we have a total amount of 2,250,000 kW which satisfies all the minimum quantities. Thus, Bill's Wind Farm and PG&E will sell a total of 2,250,000 kW at 3 cents per kW to Ford Motors and Santa Clara Electric.

Here are the specifications of the algorithm that California wants you to design. The input to the algorithm will be a list of buy and sell orders, each with at most one secret price. The output is a list of valid sales. If no sale is possible, the output list should be empty.

Question 2: (Goodrich and Tamassia) A part of doing internationally involves the trading of different currencies, and the markets that facilitate such trades can fluctuate during a trading day in ways that create profit opportunities. For example, at a given moment during a trading day, 1 U.S. dollar might be worth 0.98 Canadian dollar, 1 Canadian dollar might be worth 0.81 euros, and 1 euro might be worth 1.32 U.S. dollars. Sometimes, as in this example, it is possible for us to perform a cyclic sequence of currency exchanges, all at the same time, and end up with more money that we started with, which is an action known as ***currency arbitrage***. For instance, with the above exchange rates, we could perform a cyclic sequence of trades from U.S. dollars to Canadian dollars to euros and back to U.S. dollars, which could turn \$1,000,000 into \$1,047,816. (Assume any overheads and commission costs are worked into the exchange rates.) Suppose you are given a complete directed graph G that represents the currency exchange rates that exist at a given moment in time on a trading day. Each vertex in G is a currency, and each directed edge (v,w) in G is labeled with an exchange rate $r(v,w)$ that is the amount of currency w that would be exchanged for 1 unit of currency v. In order to profit from this information, you need to find, as quickly as possible, a cycle $(v_1, v_2, \ldots, v_k, v_1)$ that maximizes the product, $r(v_1,v_2) \times r(v_2,v_3) \times \ldots \times r(v_k, v_1)$, such that this product is strictly greater than 1. Give and properly analyze an efficient dynamic programming algorithm for finding such a cycle, if it exists.

Question 3: (Kleinberg and Tardos) You have a set of n jobs and a single processor. Each job has a one or more time intervals that it needs to use the processor. For example, job i might need to processor from 1pm to 2pm and 5pm to 6pm. No other job that requires those times can be using the processor if job i is running. However, the processor is free at all other times to run other jobs. Each job has 1 or more such intervals when it requires the processor. You are given an input number k and you want to know if it is possible to find k jobs from the list of n that can all be completed on the processor (i.e. that no job in the k requires a slot of time also used by another of the k jobs) ?

Question 4: (Kleinberg and Tardos) You are given a directed graph G with a source s and a sink t. Each edge e has a value m_e, where m_e is the minimum amount of flow we want on edge e. There is no upper bound on how much flow an edge can have. Give an algorithm to to find a flow from s to t that is as small as possible and meets the minimum flow requirement for each edge.

Question 5: You decide to take a part-time job in the construction business. Much of your work is spent cutting wooden beams into appropriate length pieces. Being new to the team, you are not trusted with measuring the lengths for the cuts. Instead each beam comes with the cut locations already marked on it. Your job is to place the beam into the saw at the appropriate position for the cut, and then to run the saw to cut the beam in two. You soon notice that it is much easier to place a shorter beam into the saw than it is to place a longer beam. You decide to use your algorithm knowledge to decrease the time it takes to cut a beam into pieces.

Specifically, you must cut a beam into n pieces where $(l_1, l_2, \ldots , l_n)$ is the sequence of lengths for the cuts. That is, l_1 is the length from the front end of the beam to the first mark, l_i for $1 < i < n$ is the length from mark i-1 to mark i, and l_n is the length from the last mark to the back end of the beam. Assume that the cost of cutting a beam segment into two pieces is $c + kL$ where L is the length of the beam segment before you cut it in two and k, c are positive constants. Your algorithm should determine the optimal order for cutting the beam at the marks in order to minimize the total cost.

Question 6: (Goodrich and Tamassia) You need to ship n packages across the country using trucks. Each package i has a weight w_i, and each truck can hold a total weight of M. You want to make the shipment using as few trucks as possible. More formally, given n packages with weights $w_1, w_2, \ldots, w_n$, and numbers M and K, is there an efficient algorithm to let you to ship the packages using K trucks where the total weight carried by each truck is at most M, and K is at most twice the minimum possible?