GRAPH PROBLEMS

1. Suppose that you want to get from vertex s to vertex t in an unweighted graph G = (V, E), but you would like to stop by vertex u if it is possible to do so without increasing the length of your path by more than a factor of α. Describe an efficient algorithm that would determine an optimal s-t path given your preference for stopping at u along the way if doing so is not prohibitively costly. (It should either return the shortest path from s to t or the shortest path from s to t containing u, depending on the situation.) If it helps, imagine that there are burgers at u

2. Ted and Marshall are taking a road trip from Somerville to Vancouver (that's in Canada). Because it's a 52-hour drive, Ted and Marshall decide to switch off driving at each rest stop they visit; however, because Ted has a better sense of direction than Marshall, he should be driving both when they depart and when they arrive (to navigate the city streets). Given a route map represented as a weighted undirected graph G = (V, E, w) with positive edge weights, where vertices represent rest stops and edges represent routes between rest stops, devise an efficient algorithm to find a route (if possible) of minimum distance between Somerville and Vancouver such that Ted and Marshall alternate edges and Ted drives the first and last edge

3. Professor Kirk has managed to get himself lost in his brand new starship. Furthermore, while boldly going places and meeting strange new, oddly humanoid aliens, his starship's engines have developed a strange problem: he can only make "transwarp jump" to solar systems at distance exactly 5 from his location. Given a starmap represented as an unweighted undirected graph G = (V, E), where vertices represent glorious new solar systems to explore and edges represent transwarp routes, devise an efficient algorithm to find a route (if possible) of minimum distance from Kirk's current location s to the location t representing Earth, that Kirk's ship will be able to follow. Please hurry—Professor Kirk doesn't want to miss his hot stardate!

4. In the longest path problem, we're given a weighted directed graph G = (V, E, w), a source s ∈ V , and we're asked to find the longest simple path from s to every vertex in G. For a general graph, it's not known whether there exists a polynomial-time algorithm to solve this problem. If we restrict G to be acyclic, however, this problem can be solved in polynomial time. Give an efficient algorithm for finding the longest paths from s in a weighted directed acyclic graph G, give its runtime, and explain why your solution doesn't work when G is not acyclic.

5. You're playing the hit new platform video game, Mega Meat Man, and are having trouble getting through Level 6006. You've decided to model the level as a directed graph, where each vertex represents a platform you can reach, and each edge represents a jump you can try to make. After extensive experimentation, you've labeled each edge with the probability (a number in [0, 1]) that you can successfully make the jump. Unfortunately, if you fail to make any jump, you instantly die, and have to start over. Describe an efficient algorithm to find a path from the start platform s to the goal platform t that maximizes the probability of a successful traversal.
6. We are given a directed graph G = (V, E), and, for each edge (u, v) ∈ E, we are given a probability f(u, v) that the edge may fail. These probabilities are independent. The reliability π(p) of a path p = (u1, u2, . . . uk) is the probability that no edge fails in the path, i.e. π(p) = (1 − f(u1, u2)) · (1 − f(u2, u3)). . . · (1 − f(uk−1, uk)). Given a graph G, the edge failure probabilities, and two vertices s, t ∈ V , we are interested in finding a path from s to t of maximum reliability.
    a. Propose an efficient algorithm to solve this problem. Analyze its running time.
    b. You tend to be risk-averse and in addition to finding a most reliable simple path from s to t, you also want to find a next-most reliable simple path, and output these two paths. Propose an algorithm to solve the problem, argue its correctness, and give its asymptotic running time.

7. When an airline is compiling flight plans to all destinations from an airport it serves, the flight plans are plotted through the air over other

airports in case the plane needs to make an emergency landing. In other words, flights can be taken only along pre-defined edges between airports. Two airports are adjacent if there is an edge between them. The airline also likes to ensure that all the airports along a flight plan will be no more than three edges away from an airport that the airline regularly serves. Given a graph with V vertices representing all the airports, the subset W of V which are served by the airline, the distance w(u, v) for each pair of adjacent airports u, v, and a base airport s, give an algorithm which finds the shortest distance from s to all other airports, with the airports along the path never more than 3 edges from an airport in W.

8. Suppose you are given an adjacency-list representation of an N-vertex graph undirected G with non-negative edge weights in which every vertex has at most 5 incident edges. Give an algorithm that will find the K closest vertices to some vertex v in O(K log K) time.

9. The diameter of a weighted undirected graph G = (V, E) is the maximum distance between any two vertices in G, i.e. $\Delta(G) = \max_{u,v \in V} \delta(u, v)$ where $\Delta(G)$ is the diameter of G and $\delta(u, v)$ is the weight of a shortest path between vertices u and v in G. Assuming that all edge weights in G are non-negative, give an O(E + V log V)-time algorithm to find a value D that satisfies the following relation: $\Delta(G)/2 \leq D \leq \Delta(G)$. You must prove that the value of D output by your algorithm indeed satisfies the above relation. Hint: For any arbitrary vertex u, what can you say about $\max_{v \in V} \delta(u, v)$?

10. You are at an airport in a foreign city and would like to choose a hotel that has the maximum number of shortest paths from the airport (so that you reduce the risk of getting lost). Suppose you are given a city map with unit distance between each pair of directly connected locations. Design an O(V +E)-time algorithm that finds the number of shortest paths between the airport (the source vertex s) and the hotel (the target vertex t).

11. Consider a mixed unweighted graph G = (V, E) with both directed and undirected edges. Assume that initially there are no cycles in G which use only directed edges. Give an algorithm to assign direction to each of the undirected edges so that the completely directed graph so obtained has no cycles. Analyze the asymptotic complexity of the algorithm.

12. Modify Dijsktra's algorithm to find, out of all shortest paths, the one with fewest number of edges from a start vertex s to all other vertices.
    a. Propose an augmented data-structure for each node for solving this problem.
    b. Modify the RELAX function with your augmentation.

RELAX(u,v,w) if (____) // relax condition then d[v] = d[u] + w(u,v) __________________________________ parent[v] = u

13. Consider a road network modelled as a weighted undirected graph G with positive edge weights where edges represent roads connecting cities in G. However some roads are known to be very rough, and while traversing from city s to t we never want to take a route that takes more than a single rough road. Assume a boolean attribute r[e] for each edge e which indicates if e is rough or not. Give an efficient algorithm to compute the shortest distance between two cities s and t that doesn't traverse more than a single rough road. (Hint: Transform G and use a standard shortest path algorithm as a black-box.)

14. In the graph of 2x2x2 Rubik's cube positions (as in Problem Set 4), there are exactly 6 edges incident on each vertex. We want to use a bi-directional BFS to find a shortest path between two vertices s and t. If the shortest path from s to t contains d edges, approximately how many vertices must be explored to find a shortest path, in the worst case? How does this compare to the number of vertices required for a single BFS starting from s?

15. An efficiency-crazed (complexity minded) algorithms developer tries to reduce the amount of work done in the Bellman-Ford

algorithm. Her modified Bellman-Ford first runs a modified BFS starting from s that keeps track of the BFS step d(e) in which it discovers each edge e. Thus, $1 \leq d(e) < |V|$. In the Bellman-Ford algorithm itself, instead of relaxing all the edges in every iteration, at iteration i (i starts at 1 and goes up to $|V| - 1$), she only relaxes the edges with $d(e) \leq i$. Is this be guaranteed to give the shortest distance from the s to all the vertices or detect a negative cycle? Explain why or provide a small counterexample

16. Beverly owns a vacation home in Hawaii and wishes to rent the place out for n days beginning on May 1st (on the n + 1st day, she plans to take a vacation there herself). She has obtained m bids, each of which has a starting day si and ending day ei (between 1 and n), and the amount \$i that the bidder is willing to pay. Beverly can only rent the house to a single bidder on any given day. (That is, she may not accept two bids bi and bj such that the intervals (si , ei) and (sj , ej) overlap.) Beverly decides to model this problem as a directed graph with weighted edges so that she can use a standard graph algorithm (or a minor variation of a standard algorithm) to find the bids to accept which maximizes her revenue. Describe such a model, and give the asymptotic cost of finding the set of bids that maximizes the revenues. Assume that the bids are given as list of tuples, not necessarily sorted in any particular order. Try to find an algorithm that is as efficient as possible.

17. At Podunk State University, there are 100 courses, and n students. The number of students n varies from term to term and is virtually unbounded because Podunk State is required by law to admit all students who apply for admission (and even a few who don’t). Since there are few courses and many students, it is common to find groups of students who are all taking the same set of courses at the same time. In order to optimize its available resources in light of the current economic crisis, the administration wants to know how many clusters there are, where a cluster is defined as a maximal set of students who are all registered for the same set of courses. The administration hires you to solve this problem. You are given a table that lists all n students and the courses each is taking. Design an

efficient algorithm to determine the number of clusters. Analyze its running time as a function of n

18. The Enterprise is in orbit around Starbase 6006. An urgent medical situation requires that it travel as quickly as possible to the distant planet Vertex T. In fact, the situation is so urgent that the Federation requests that the Enterprise arrive yesterday. Your goal is to determine if it is possible to get the Enterprise from s to t in negative total time, so that it arrives before it leaves. You have a directed graph G representing the Universe. Each known location in Federation space is a vertex in the graph. There are a number of hyperspace bypasses between these points; a hyperspace bypass from point u to point v is represented by the edge (u, v). The weight of the edge w(u, v) is equal to the amount of time that passes while traversing it. Notice that edges may have negative weights! Some bypasses run through some sort of crazy time-warp thing, causing the ship to travel back in time.
    a. Ensign Dijkstra volunteers to lay in the course. In one sentence, explain why Dijkstra's algorithm can't be used to solve this problem.
    b. You ask Ensign Bitdiddle to plot the course instead. Supposing that G contains no negative-weight cycles, describe an algorithm that returns TRUE if there exists a path from s to t with negative total weight, and FALSE otherwise. Analyze the running time of your algorithm.
    c. Lieutenant Data tells you that there is exactly one negative-weight cycle present in G. Describe an algorithm that returns TRUE if there exists a path from s to t with negative total weight, and FALSE otherwise. (You may assume that you know which vertices are in the cycle.) Analyze the running time of your algorithm.

19. Suppose you want to get from s to t on weighted graph G with nonnegative edge weights, but you would like to stop by u if it isn't too inconvenient. (Here too inconvenient means that it increases the length of your travel by more than 10%.) Describe an efficient algorithm that would determine an optimal s to t path given your preference for stopping at u along the way if not too inconvenient. It

should either return the shortest path from s to t, or the shortest path from s to t containing u.

20. Suppose that you implement Dijkstra's algorithm using a priority queue algorithm that requires O(V) time to initialize, worst-case f(V, E) time for each EXTRACT-MIN operation and worst-case g(V, E) time for each DECREASE-KEY operation. How much (worst-case) time does it take to run Dijkstra's algorithm on an input graph G = (V, E)?.

21. Consider a network of computers represented by a directed graph G. Each vertex v ∈ V represents a computer, and each edge (u, v) ∈ E represents a network link from u to v.
    a. Let each edge (u, v) in G have a weight w(u, v) ∈ (0, 1], representing the probability that a packet going from u to v is successfully delivered. Give an efficient algorithm to find the path along which a packet has the highest probability of reaching its destination (i.e., the path for which the product of the edge weights is maximized). Hint: Transform the weights and use a shortest path algorithm.
    b. Now, instead of weighted edges, consider weighted vertices. In other words, network links never drop packets, but nodes might. Edges are not weighted, but each vertex v has a weight w(v) ∈ (0, 1], representing the probability that an incoming packet is not dropped by that node. Give an efficient algorithm to find the path along which a packet has the highest probability of reaching its destination (i.e., the path for which the product of the vertex weights is maximized). Hint: Transform the graph and use the algorithm from part (a)

22. You are given a list of all scheduled daily flights in the US, giving departure airports, departure times, destination airports, and arrival times. We want an algorithm to compute travel times between airports, including waiting times between connections. Assume that if one flight arrives at time t and another departs at time t 0 ≥ t, travelers can make the connection. Further assume that at a given

airport, no two events (arrivals or departures) occur at the same time, and that there are at most 100 events at any airport during a given day. All times are given in GMT; don't worry about time zones. Construct a weighted graph so that given a departure airport, a departure time T, and a destination airport, we can efficiently determine the earliest time T 0 that a traveler can be guaranteed (according to the schedules!) of arriving at her destination on that day (ignore overnight flights and overnight airport stays).

a. What do vertices represent? What do edges in your graph represent, and what is the weight of an edge?
b. Give an upper bound on the number of edges and vertices in your graph if there are n airports in the US and m daily flights. Justify your bound.
c. What algorithm would you use to compute the shortest travel times, and what is its running time in terms of the number of vertices, V , and the number of edges, E?

23. We define an articulation point as a vertex that when removed causes a connected graph to become disconnected. For this problem, we will try to find the articulation points in an undirected graph G.

a. How can we efficiently check whether or not a graph is disconnected? (Hint: think of a recent problem set question)
b. Describe an algorithm that uses a brute force approach to find all the articulation points in G in O(V (V + E)) time.
c. Suppose we run DFS on graph G. Consider the types of edges that can exist in a DFS tree produced from an undirected graph, recalling that cross edges can't happen in the DFS of an undirected graph. Argue that a non-root, non-leaf vertex u is an articulation point if and only if there exists a subtree rooted at a child of u that has no back edges to a proper ancestor of u.

24. Your new startup, Bird Tours, brings people around Boston in a new aviation car that can both drive and fly. You've constructed a weighted directed graph G = (V, E, w) representing the best time to drive or fly between various city sites (represented by vertices in V). You've also written a history of Boston, which would be best described by visiting sites v0, v1, . . . , vk in that order. Your goal is to

find the shortest path in G that visits v0, v1, . . . , vk in order, possibly visiting other vertices in between. (The path must have v0, v1, . . . , vk as a subsequence; the path is allowed to visit a vertex more than once. For example, v0, v2, v1, v2, . . . , vk is legal.) To do the computation, you've found an online service, Paths ' RUs, that will compute the shortest path from a given source s to a given target t in a given weighted graph, for the bargain price of $1. You see how to solve the problem by paying $k, calling Paths ' RUs with (v0, v1, G),(v1, v2, G), . . . ,(vk−1, vk, G) and piecing together the paths. Describe how to solve the problem with only $1 by calling Paths ' RUs with (s, t, G0) for a newly constructed graph G0 = (V 0 , E0 , w0), and converting the resulting path into a path in G.

25. You are traveling by car from one city to another city. Unfortunately, you have a hole in your gas tank, and you have to refill your gas tank to travel across more than two roads. In addition, there is a toll booth on every road that charges you for using that road. Your goal is to find the least-expensive path from your start to your destination. You represent the city network using a directed graph G = (V, E, w) with weights w defined on both edges and vertices. The vertices V represent the cities and the edges E represent the roads. The weight w(e) of an edge e represents the toll amount on that road. The weight w(v) of a vertex v is the price of filling your gas tank in that city (which is a fixed price independent of how much gas you have left, or ∞ if there is no gas available to purchase). You are allowed (but not obligated) to end your journey with an empty tank, and you may assume that you always start your journey with a full tank. Below is an example graph that we will use to answer part (a). One seemingly cheap path from s to t is (s, u1, u2, t) at a cost of $8. Unfortunately, this path is not valid, because our leaky gas tank won't permit moving across three edges without refilling our gas tank. One valid path is (s, u3, u2, t) at a cost of $22. (This is a valid path: we begin with a full tank, travel across one edge to a gas station where we refill our tank, and then travel two edges to the destination, arriving with an empty gas tank. Notice that we are unable to continue the journey to u5 if we wanted to, because even though there is a gas station there, we have to traverse a third edge to get to it.)

a. The valid path given in the description above is not the best path. Find a least-expensive path from s to t in the graph above.
b. Find the least-expensive path from s to u5 in the graph above
c. Give an O(V log V +E) algorithm to find, in a given graph G = (V, E, w), a least-expensive valid path from a city s to a city t, or report that no such path exists.