CS/ECE 374 A ✧ Fall 2019

Homework 8

Due Tuesday, November 5, 2019 at 8pm

This is the last homework before Midterm 2.

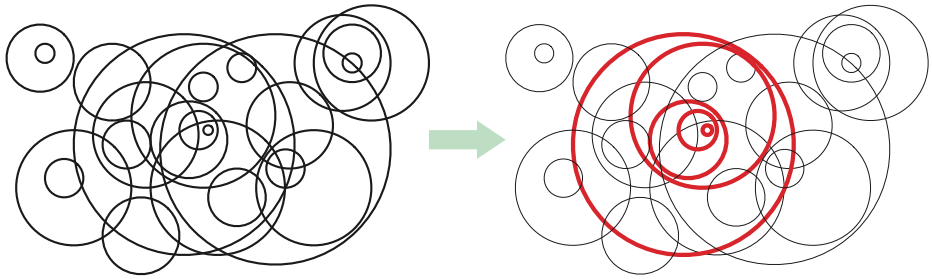
---

1. Over the summer, you pick up a part-time consulting gig at a gun range, designing targets for their customers to shoot at. A *target* consists of a properly nested set of circles, meaning for any two circles in the set, the smaller circle lies entirely inside the larger circle. The score for each shot is equal to the number of circles that contain the bullet hole.

   The shooters at the range aren't very good; their shots tend to be distributed uniformly at random on the target sheet. As a result, the expected score for any shot is proportional to the sum of the areas of the circles that make up the target. You'd like to make this expected score as large as possible.

   Now suppose your boss hands you a target sheet with n circles drawn on it. Describe and analyze an efficient algorithm to find a properly nested subset of these circles that maximizes the sum of the circle areas. You cannot *move* the circles; you must keep them exactly where your boss has drawn them.

   The input to your algorithm consists of three arrays $R[1..n]$, $X[1..n]$, and $Y[1..n]$, specifying the radius of each circle and the x- and y-coordinates of its center. The output is the sum of the circle areas in the best target.



2. The Cheery Hells neighborhood of Sham-Poobanana runs a popular and well-regulated Halloween celebration, attended by thousands of costumed children from all across Poobanana County. To regulate and protect the flood of costumed children, the Cheery Hells Neighborhood Association has designated a walking direction for each stretch of sidewalk.

   After paying the $25 entrance fee, each child receives a complete map of the neighborhood, in the form of a directed graph G, whose vertices represent houses. Each edge $v \to w$ indicates that one can walk directly from house v to house w following the designated sidewalk directions. (Anyone caught walking backward along a sidewalk is summarily ejected from Cheery Hells, without their candy. No refunds.) One special vertex s designates the entrance to Cheery Hells. Children can visit houses as many times

as they like, but biometric scanners at every house ensure that each child receives candy only at their *first* visit to each house.

Unknown to the Neighborhood Association, the children of Cheery Hells have published a secret web site, accessible only through a link embedded in yet another TikTok cover of "Spooky Scary Skeletons", listing the amount of candy that each house in Cheery Hells will give to each visitor. (The web site also asks visitors to say "Gimme some Skittles, but I don't wanna pay for them" instead of "Trick or treat", just to mess with the grownups.)

Describe and analyze an algorithm to compute the maximum amount of candy that a single child can obtain in a walk through Cheery Hells, starting at the entrance node s. The input to your algorithm is the directed graph G, along with a non-negative integer $c(v)$ for each vertex describing the amount of candy that house gives each first-time visitor.

[Hint: Think about two special cases first: (1) Cheery Hells is strongly connected, and (2) Cheery Hells is acyclic. Solving only these two special cases is worth half credit.]

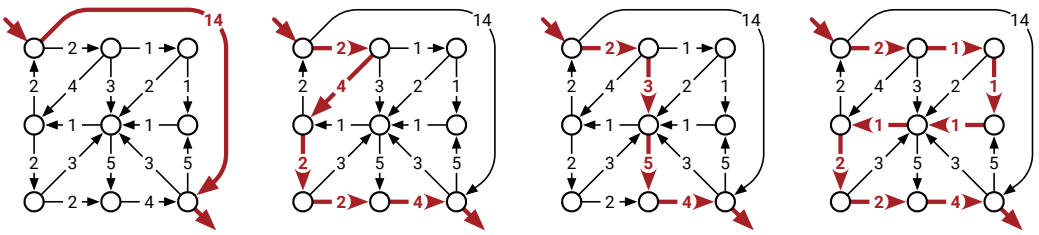
3. Morty needs to retrieve a stabilized plumbus from the Clackspire Labyrinth. He must enter the labyrinth using Rick's interdimensional portal gun, traverse the Labyrinth to a plumbus, then take that plumbus through the Labyrinth to a fleeb to be stabilized, and finally take the stabilized plumbus back to the original portal to return home. Plumbuses are stabilized by fleeb juice, which any fleeb will release immediately after being removed from its fleebhole. An unstabilized plumbus will explode if it is carried more than 137 flinks from its original storage unit. The Clackspire Labyrinth smells like farts, so Morty wants to spend as little time there as possible.

   Rick has given Morty a detailed map of the Clackspire Labyrinth, which consist of a directed graph $G = (V, E)$ with non-negative edge weights (indicating distance in flinks), along with two disjoint subsets $P \subset V$ and $F \subset V$, indicating the plumbus storage units and fleebholes, respectively. Morty needs to identify a start vertex s, a plumbus storage unit $p \in P$, and a fleebhole $f \in F$, such that the shortest-path distance from p to f is at most 137 flinks long, and the length of the shortest walk $s \rightsquigarrow p \rightsquigarrow f \rightsquigarrow s$ is as short as possible.

   Describe and analyze an algo(burp)rithm to so(burp)olve Morty's problem. You can assume that it is in fact possible for Morty to succeed.

Solved Problem

4. Although we typically speak of "the" shortest path from one vertex to another, a single graph could contain several minimum-length paths with the same endpoints.



Four (of many) equal-length shortest paths.

Describe and analyze an algorithm to compute the *number* of shortest paths from a source vertex s to a target vertex t in an arbitrary directed graph G with weighted edges. Assume that all edge weights are positive and that any necessary arithmetic operations can be performed in $O(1)$ time each.

[Hint: Compute shortest path distances from s to every other vertex. Throw away all edges that cannot be part of a shortest path from s to another vertex. What's left?]

Solution: We start by computing shortest-path distances $dist(v)$ from s to v, for every vertex v, using Dijkstra's algorithm. Call an edge $u \to v$ ***tight*** if $dist(u) + w(u \to v) = dist(v)$. Every edge in a shortest path from s to t must be tight. Conversely, every path from s to t that uses only tight edges has total length $dist(t)$ and is therefore a shortest path!

Let H be the subgraph of all tight edges in G. We can easily construct H in $O(V+E)$ time. Because all edge weights are positive, H is a directed acyclic graph. It remains only to count the number of paths from s to t in H.

For any vertex v, let $NumPaths(v)$ denote the number of paths in H from v to t; we need to compute $NumPaths(s)$. This function satisfies the following simple recurrence:

$$NumPaths(v) = \begin{cases} 1 & \text{if } v = t \\ \displaystyle\sum_{v \to w} NumPaths(w) & \text{otherwise} \end{cases}$$

In particular, if v is a sink but $v \neq t$ (and thus there are no paths from v to t), this recurrence correctly gives us $NumPaths(v) = \sum \varnothing = 0$.

We can memoize this function into the graph itself, storing each value $NumPaths(v)$ at the corresponding vertex v. Since each subproblem depends only on its successors in H, we can compute $NumPaths(v)$ for all vertices v by considering the vertices in reverse topological order, or equivalently, by performing a depth-first search of H starting at s. The resulting algorithm runs in $O(V+E)$ time.

The overall running time of the algorithm is dominated by Dijkstra's algorithm in the preprocessing phase, which runs in $\boldsymbol{O(E \log V)}$ ***time***. ■

Rubric: 10 points = 5 points for reduction to counting paths in a dag (standard graph reduction rubric) + 5 points for the path-counting algorithm (standard dynamic programming rubric)