

CS375 Weekly Assignment 9

Due April 29nd 10PM ET

A general note for CS375: When writing up your homework, please write neatly and explain your answers clearly, giving all details needed to make your answers easy to understand. Graders may not award full credit to incomplete or illegible solutions.

You are encouraged to discuss the problems with your classmates, **but your submission should be your own**. When writing your solutions up, you should not be looking at any other student's submission.

Clear communication **is the point**, on every assignment. In general in CS375, unless explicitly specified otherwise, answers should be accompanied by explanations. Answers without explanations may not receive full credit. Please feel free to ask me any questions about explanations that might come up!

Necessary acknowledgements: many of these questions are reproductions or adaptations of questions of Eric Aaron's devising. Others are from our course texts as listed on the syllabus.

Recall the MAXFLOW problem from class:

MAXFLOW

Input: A graph $G = (V, E)$ with vertices V and directed edges E with capacities $c(u, v) \geq 0$ for each edge (u, v) in E and two marked vertices s and t (we'll assume there are no edges into s or out of t).

Output: The maximum flow on the graph.

Constraints:

1. For each vertex v other than s and t it holds that $\sum_u f(u, v) = \sum_w f(v, w)$:
ie, the flow into all vertices equals the flow out.
2. For each edge (u, v) in E it holds that $f(u, v) \leq c(u, v)$.

In order to define the Ford-Fulkerson algorithm in class, we defined the residual graph of G with respect to f as $G_f = (V, E_f)$ where $c_f(u, v) = c(u, v) - f(u, v)$, $c_f(v, u) = f(u, v)$, and E_f is all the edges with positive residual capacity c_f (note that this may remove some edges from E and may include some reverse edges).

We defined the Ford-Fulkerson algorithm as follows:

Ford-Fulkerson($G = (V, E), s, t$):

 initialize $f(u, v) = 0$ for all (u, v) in E

 while there exists a path P from s to t in G_f :

 let (u, v) be the edge of minimal residual capacity in P

 Increase the flow f by $c_f(u, v)$ for every edge in P .

We also considered the following problem:

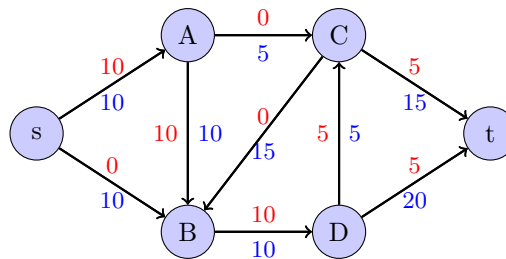
MINCUT

Input: A graph $G = (V, E)$ with vertices V and directed edges E with capacities $c(u, v) \geq 0$ for each edge (u, v) in E and two marked vertices s and t (we'll assume there are no edges into s or out of t).

Output: The minimum s - t cut in the graph, where an s - t cut is a partition of the vertices of G into two sets S and T such that s is in S and t is in T . The value of an s - t cut (S, T) is defined as $\sum_{u \in S} \sum_{v \in T} c(u, v)$, the sum of the capacities of edges going from S to T .

and showed that the maximum flow on a graph is equal to the minimum cut. In fact, we showed that technically the FORDFULKERSON algorithm was capable of determining the maximum flow and minimum cut, effectively designing an algorithm for both problems simultaneously.

- (a) Consider the following graph with capacities marked in **blue**, on which I've already pushed some flow (in **red**):



Using this as a starting point (so don't reset the flow to 0), enter the while loop in the FORDFULKERSON algorithm, trace through what happens, and determine the max flow.

- Identify an S-T cut with cut-capacity equal to the maximal flow.

- Consider the following problem:

CIRCULATION

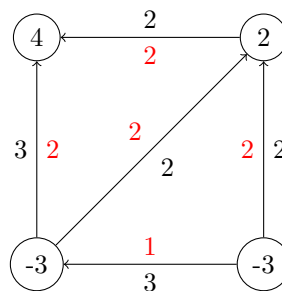
Input: A directed graph G where each edge $e = (u, v)$ has an integer capacity $c(u, v)$ and each vertex v has a demand d_v .

Output: TRUE if there is a way to set flow values on $f(u, v)$ to handle the demands of each vertex, otherwise FALSE.

Constraints:

- The flow on each edge is at most the capacity: $f(u, v) \leq c(u, v)$ for all edges (u, v) .
- The demands on each vertex v are satisfied: $\sum_u f(u, v) - \sum_w f(v, w) = d_v$.

As an example:



The bottom left node has a flow of 1 entering and a flow of 4 leaving, satisfying its demand of -3 .

Show that $\text{CIRCULATION} \leq_p \text{MAXFLOW}$.

3. Consider the following problem:

LOWERBOUNDEDCIRCULATION

Input: A directed graph G where each edge $e = (u, v)$ has an integer capacity $c(u, v)$ and an integer lowerbound $\ell(u, v)$, and each vertex v has a demand d_v .

Output: TRUE if there is a way to set flow values on $f(u, v)$ to handle the demands of each vertex, otherwise FALSE.

Constraints:

- The flow on each edge is at least the lowerbound and at most the capacity:
 $\ell(u, v) \leq f(u, v) \leq c(u, v)$ for all edges (u, v) .
- The demands on each vertex v are satisfied: $\sum_u f(u, v) - \sum_w f(v, w) = d_v$.

- (a) Show that $\text{LOWERBOUNDEDCIRCULATION} \leq_p \text{CIRCULATION}$, which in turn means that $\text{LOWERBOUNDEDCIRCULATION} \leq_p \text{MAXFLOW}$.
- (b) Suppose I have an algorithm for MAXFLOW with a runtime of $f(n, m)$ where n is the number of vertices and m is the number of edges in the graph given as input. Using the reductions you've determined so far, what would be the resulting runtime of the induced algorithm for $\text{LOWERBOUNDEDCIRCULATION}$?
4. Pick 2 of the following problems (I'll probably have you pick another 2 next week). For each problem, assume you have a blackbox algorithm for any of the problems above (MAXFLOW , MINCUT , CIRCULATION , $\text{LOWERBOUNDEDCIRCULATION}$) with a runtime of $f(n, m)$ where n is the number of vertices and m the number of edges in the input graph.
- (a) Consider the following problem:

DISJOINTPATHS

Input: A directed graph G and two marked vertices s and t

Output: The maximum number of edge-disjoint paths from s to t

Constraints: Here, two paths are called edge-disjoint if they have no over-lapping edges (though they are allowed to have over-lapping vertices)

Design and analyze an algorithm for this problem.

- (b) (*Algorithm Design*) The computational problems faced by the nation's large airline carriers are almost too complex to even imagine. They have to produce schedules for thousands of routes each day that are efficient in terms of equipment usage, crew allocation, customer satisfaction, and a host of other factors all in the face of unpredictable issues like weather and breakdowns. It's not surprising that they're among the largest consumers of high-powered algorithmic techniques.

Covering these computational problems in any realistic level of detail would take us much too far afield. Instead, we'll discuss a toy problem that captures, in a very clean way, some of the resource allocation issues that arise in a context such as this.

Suppose you're in charge of managing a fleet of airplanes and you'd like to create a flight schedule for them. Here's a very simple model for this. Your market research has identified a set of m particular flight segments that would be very lucrative if you could serve them; each flight segment f is specified by four parameters: its origin airport f_S , its destination airport f_T , its departure time f_D , and its arrival time f_A . Given two flight segments f and g where $f_T = g_S$ and $f_A \leq g_D$, a single plane can be used to handle the flights.

Design and analyze an algorithm to determine the minimum number of planes are sufficient to handle a given list of flight segments.

- (c) (*Algorithm Design*) As an input, we are given an underlying set of projects $P = \{p_1, \dots, p_n\}$ where each project p_i has associated revenue r_i (which may be positive or negative) and requires a (possibly empty) subset P_i of P of prerequisite projects to be completed first.

Design and analyze an algorithm to determine the maximal revenue obtainable from the projects.

- (d) (*Algorithm Design*) As an input, we are given:

- k vacation periods of varying lengths combining for a total of m vacation days
- n doctors, each of which has specified over all the vacation days exactly which days they could come to work
- a parameter c determining the maximal number of vacation days each doctor can be asked to come in.

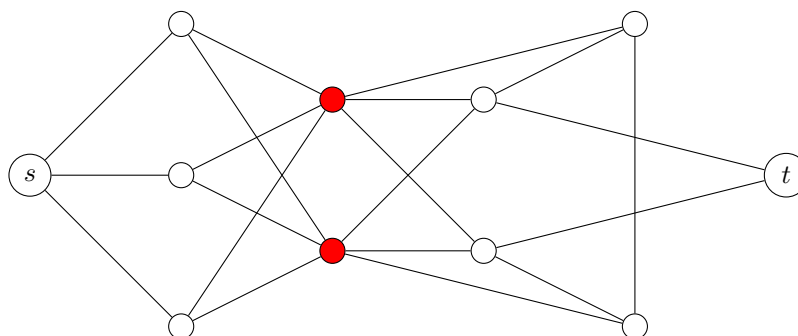
Design and analyze an algorithm which takes the above info and determines if we can assign doctors to work each vacation day so that

- each day as a doctor working (no day is missed),
- no doctor works more than c days total across all vacation days
- no doctor works more than 1 day per vacation period (so, a doctor could work 1 day over thanksgiving break and another day over winter break, but not 2 days of July 4th weekend, etc.)

- (e) (Erickson 11.4) The Island of Sodor is home to a large number of towns and villages, connected by an extensive rail network. Recently, several cases of a deadly contagious disease (either swine flu or zombies; reports are unclear) have been reported in the village of Skarloey. The controller of the Sodor railway plans to close down certain railway stations to prevent the disease from spreading to Tidmouth, his home town. No trains can pass through a closed station. To minimize expense (and public notice), he wants to close down as few stations as possible. However, he cannot close the Skarloey station, because that would expose him to the disease, and he cannot close the Tidmouth station, because then he couldn't visit his favorite pub.

Describe and analyze an algorithm to find the minimum number of stations that must be closed to block all rail travel from Skarloey to Tidmouth. The Sodor rail network is represented by an undirected graph, with a vertex for each station and an edge for each rail connection between two stations. Two special vertices s and t represent the stations in Skarloey and Tidmouth.

For example, given the following input graph, your algorithm should return the integer 2.



- (f) (Erickson 11.6) The SPU Commuter Silence Department is installing a mini-golf course in the basement of the See-Bull Center! The playing field is a closed polygon bounded by m horizontal and vertical line segments, meeting at right angles. The course has n starting points and n holes, in one-to-one correspondence. It is always possible hit the ball along a straight line directly from each starting point to the corresponding hole, without touching the boundary of the playing field. (Players are not allowed to bounce golf balls off the walls; too much glass.) The n starting points and n holes are all at distinct locations. Sadly, the architect's computer crashed just as construction was about to begin. Thanks to the herculean efforts of their sysadmins, they were

able to recover the locations of the starting points and the holes, but all information about which starting points correspond to which holes was lost!

Describe and analyze an algorithm to compute a pairing between the starting points and the holes that meets the straight-line requirement, or to report that no such correspondence exists. The input consists of the x - and y -coordinates of the m corners of the playing field, the n starting points, and the n holes. Assume you can determine in constant time whether two line segments intersect, given the x - and y -coordinates of their endpoints.

- (g) (Erickson 11.7) A **Cycle Cover** of a given directed graph is a set of vertex-disjoint cycles that cover every vertex in G .

Describe and analyze an algorithm to find a cycle cover for a given graph, or correctly report that no cycle exists. (**Hint:** this is actually similar to the products-buyers problem we did in class)

- (h) (*Algorithm Design* 7.11) We're given the following input:

- A set of students designing surveys they wish to distribute $A_1, \dots, A_m$
- A set of demographic groups $\mathcal{D} = \{D_1, \dots, D_k\}$ (CS Majors, Johnson Pond residents, etc.) which may overlap
- A set of n students $s_1, \dots, s_n$ at Colby, where for each student we know exactly which demographics they belong to
- For each surveyor A_i , the subset $\mathcal{D}_i$ of $\mathcal{D}$ that they want to give their survey to and the number r_i of students they need to complete their survey
- For each student s_i , the maximal number of surveys x_i they are willing to complete

With the information provided, design and analyze an algorithm to determine if there is an assignment of surveys to students so that each surveyor is happy (they get the number of students from relevant demographics they wanted) and the students aren't overwhelmed (no student has to complete more surveys than they are willing).

- (i) Let M be an $n \times n$ grid consisting only of 0s and 1s. We say that M is rearrangeable if it possible to perform a sequence of row swaps or column swaps and arrive at a matrix where every entry on the main diagonal is 1. For example, the grid:

$$\begin{array}{cccc} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}$$

is rearrangeable by swapping the first and second rows, then swapping the 2nd and 3rd rows, then swapping the 3rd and 4th rows.

- i. Give an example of a matrix that is not rearrangeable despite having a 1 in every row and every column.
- ii. Design and analyze an algorithm that determines whether a given 0-1 square matrix is rearrangeable.