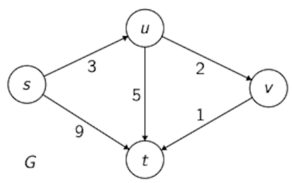
Lab 13: Shortest Paths

Module: Graphs

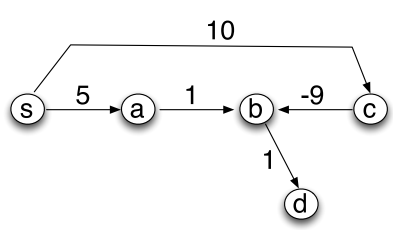
Collaboration level 0 (no restrictions). Open notes.

1. Step through `Dijkstra(G, s, t)` on the graph shown below. Complete the table below to show what the arrays d[] and p[] are at each step of the algorithm, and indicate what path is returned and what its cost is. Here D represents the set of vertices that have been removed from the PQ and their shortest paths found (in the notes we denoted it by S). .



	d[s]	d[u]	d[v]	d[t]	p[s]	p[u]	p[v]	p[t]
When entering the first while loop for the first time, the state is:	0	∞	∞	∞	None	None	None	None
Immediately after the first vertex is explored	0	3	∞	9	None	s	None	s
Immediately after the second vertex is explored								
Immediately after the third vertex is explored								
Immediately after the fourth vertex is explored								

2. Consider the directed graph below and assume you want to compute SSSP(s).



(a) Run Dijkstra's algorithm on the graph above step by step. Are there any vertices for which d[x] is correct? Are there any vertices for which d[x] is incorrect? Why?

3. Prove that the following claim is false by showing a counterexample:

   Claim: Let $G = (V, E)$ be a directed graph with negative-weight edges, but no negative-weight cycles. Let $w, w < 0$, be the smallest weight in G. Then one can compute SSSP in the following way: transform G into a graph with all positive weights by adding $-w$ to all edges, run Dijkstra, and subtract from each shortest path the corresponding number of edges times $-w$. Thus, SSSP can be solved by Dijkstra's algorithm even on graph with negative weights.

4. You are given an image as a two-dimensional array of size $m \times n$. Each cell of the array represents a pixel in the image, and contains a number that represents the color of that pixel (for e.g. using the RGB model).

   A segment in the image is a set of pixels that have the same color and are **connected**: each pixel in the segment an be reached from any other pixel in the segment by a sequence of moves up, down, left or right.

   Design an efficient algorithm to find the size of the largest segment in the image.

Additional problems: Optional

1. **Arbitrage:** Suppose the various economies of the world use a set of currencies $C_1, C_2, ..., C_n$ –think of these as dollars pounds, bitcoins, etc. Your bank allows you to trade each currency C_i for any other currency C_j and finds some way to charge you for this service (in a manner to be elaborated in the subparts below). We will devise algorithms to trade currencies to maximize the amount we end up with.

   (a) Suppose that for each ordered pair of currencies (C_i, C_j), the bank charges a flat fee of $f_{ij} > O$ dollars to exchange C_i for C_j (regardless of the quantity of currency being exchanged). Devise an efficient algorithm which, given a starting currency C_s, a target currency C_t, and a list of fees f_{ij} for all $i, j \in \{1, ..., n\}$ computes the cheapest way (that is, incurring the least in fees) to exchange all of our currency in C_s into currency C_t. Justity the correctness of your algorithm and its runtime.

   (b) Consider the more realistic setting where the bank does not charge flat tees, but instead uses exchange rates. In particular, for each ordered pair (C_i, C_j), the bank lets you trade one unit of C_i to r_{ij} units of C_j. Devise an efficient algorithm which, given starting currency C_s, target currency C_t, and a list of rates r_{ij}, computes a sequence of exchanges that results in the greatest amount of C_t. Justify the correctness of vour algorithm and its runtime.

   Assume all exchange rates $r_{ij} > 1$.

   Hint: How can you turn a product of terms into a sum?