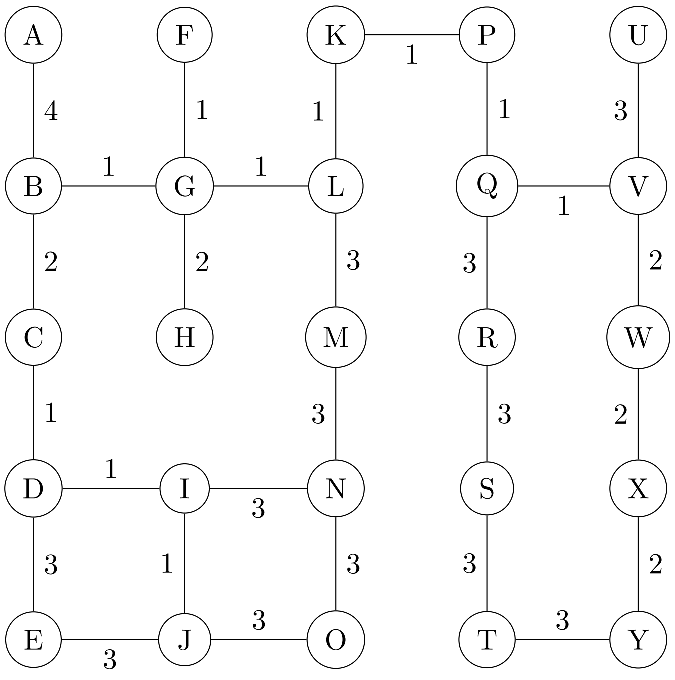
Algorithms Exercise Sheet: Computing orderings/shortcuts for CH

Instructions

Answer the following questions to deepen your understanding of contraction hierarchies. All questions will reference the following graph:



Setup for CH

Offline-vs-offline Ordering computation

Let's think about Contraction Hierarchies (CH) on this graph. The way I explained CH in class, I said that the first step is to choose an ordering of the nodes and then to perform the shortcut operation on the graph in the order chosen. This ordering methodology is called **offline**, which means that the entire ordering is picked first and then we do shortcutting second. However, there's no particular need to compute the ordering offline; it would be totally valid to choose the ordering in an **online** manner: pick a node, then shortcut it, and then pick a node, shortcut it, and so on. Computing the ordering online can be a nice way to reduce the computational burden of trying to "get the ordering right" all at once. The first 2 exercises will have you practice and evaluate online-vs-offline ordering computation.

How to choose the ordering?

There are lots of ways to think about picking an ordering. One simple "classic" heuristic is to compute the *edge difference* for each node, then order the nodes in increasing order of edge difference. The edge difference of node v is computed according to

$$\text{EdgeDiff}(v) = (\# \text{ of shortcuts added by shortcutting } v) - (\# \text{ of edges removed when } v \text{ is removed}).$$

If EdgeDiff(v) is low, this means that when v is removed, many edges are removed, but few shortcuts are added to the auxiliary graph. For example, in the graph at the top of the project, we have that EdgeDiff$A = -1$, and EdgeDiff$(E) = -2$ (in both cases, no shortcuts are added but some edges are removed), whereas EdgeDiff$(G) = 2$ (4 edges are removed, but all 6 shortcuts are added).

Exercises

1. Compute the auxiliary data for this graph in an *offline* manner using edge difference to drive your ordering. Choose an offline ordering based on edge difference (increasing order of edge difference). Note that there will be many ties; you can break the ties however you want. The main thing is that you should commit to the node ordering before you start shortcutting any nodes.

(a) What ordering was returned based on this offline approach?

(b) Based on this ordering, compute all shortcuts in order.

(c) How many total shortcuts were created?

2. Compute the auxiliary data for this graph in an *online* manner using edge difference to drive your ordering. Compute the edge difference values for the graph, and shortcut the node with the lowest edge difference. After each shortcut operation, re-compute the edge difference before you compute the next shortcut. Thus, your ordering and your set of shortcuts are determined together at the same time.

(a) What ordering was returned based on the online approach?

(b) Report the set of shortcuts which was returned using this online approach.

(c) How many total shortcuts were created?

3. **CS 5080 only:** We might try one more thing. If the goal is to minimize the size of the auxiliary data (the set of shortcuts), why not do that directly? So here, repeat the online approach, but this time don't use the whole edge difference; instead, just try to minimize the total number of shortcuts added. That is, pick the next node v to minimize based just on its shortcut score defined as:

$$\text{ShortcutScore}(v) = (\text{\# of shortcuts added by shortcutting } v).$$

Compute the auxiliary data for this graph in an *online* manner using shortcut score to drive your ordering. Compute the shortcut scores for the graph, and shortcut the node with the lowest shortcut score. After each shortcut operation, re-compute the shortcute score before you compute the next shortcut. Thus, your ordering and your set of shortcuts are determined together at the same time.

(a) What ordering was returned based on the shortcut-minimizing online approach?

(b) Report the set of shortcuts which was returned using the shortcut-minimizing online approach.

(c) How many total shortcuts were created?

(d) Speculate about whether one of these heuristics (edge difference vs. shortcut score) might be better than the other. Note there may not be a quick or universal answer, but do a bit of thinking about any kinds of differences/discrepancies you see between the two.