COMP 5204 - Assignment 1

Hussein Houdrouge (ID: 101 232 199)
Email: houdrouge.hussein@gmail.com

February 2022

Question 1: Triangulation's Hierarchy

Show via a set of illustrations and comments how Kirkpatrick's algorithm would work on the point set shown in class. (Note that the graph produced for the class notes, may not be the one you get.)

In particular, this means, producing the sequence of triangulation, the graph, and showing an execution of a point location query.

Solution. We will start by the following triangulation that is combinatorially equivalent (although the geometry matters here but I could not produce the exact figure) to the example given in the class. We will refer to it as S_1.

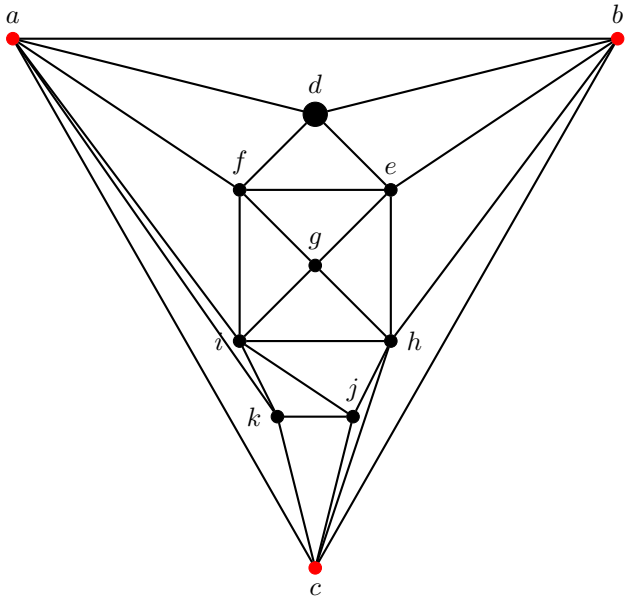

Figure 1: The triangulation S_1.

We begin by listing the vertices and their degree: $d(a) = 6$, $d(b) = 5$, $d(c) = 5$, $d(d) = 4$, $d(e) = 5$, $d(f) = 5$, $d(g) = 4$, $d(h) = 6$, $d(i) = 6$, $d(j) = 4$, and $d(k) = 4$.

For $k = 12$, the algorithm proceed to chose a set of non adjacent and non boundary vertices with degree $\leq k$. Let such set be $R_1 = \{d, g, j\}$, as Figure 2 shows.

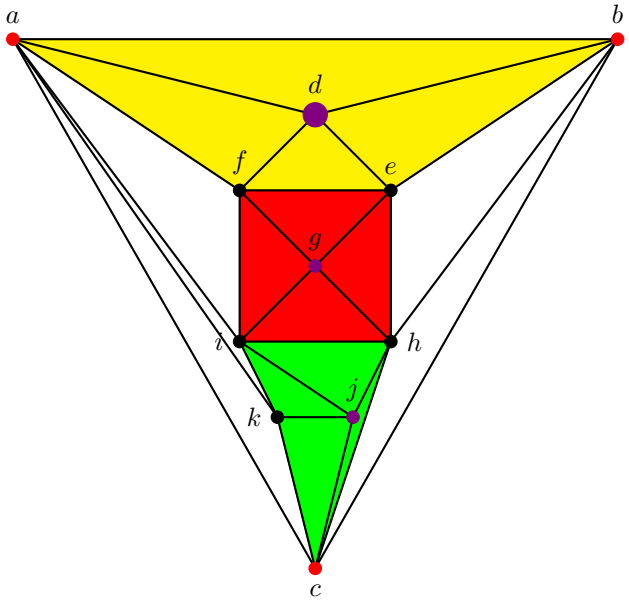

Figure 2: The set R_1 colored in purple, and the three region that will be triangulated.

The next step is to remove R_1, and triangulate the new graph.

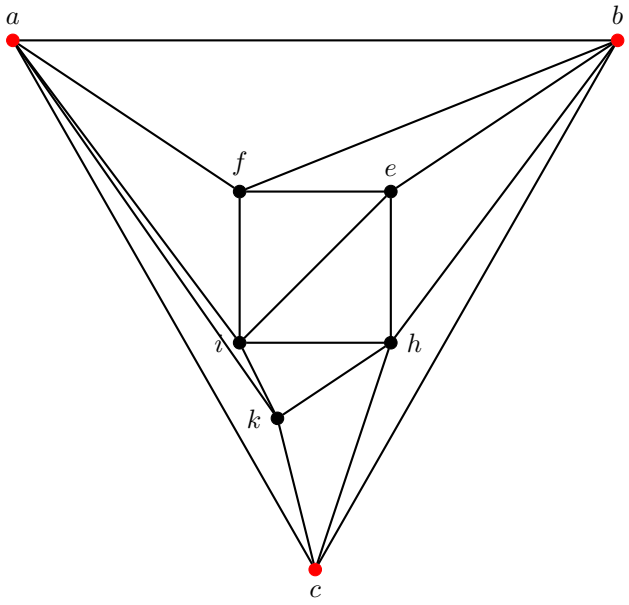

Figure 3: Removing R_1 and triangulating the new graph S_2.

After the removal of R_1, the regions that are colored in yellow ($afeb$), red ($fihe$), and ($ikch$) are deleted and subdivided to new ones (afb, feb, fie, ihe, ikh, and kch). So, we want to express how these are connected to triangles in the previous sub-regions ($adb, afd, fed, deb, fig, fge, gih, ghe, ikj, kcj, ijh$, and jch). The last operation will lead to the creation of the following hierarchy.

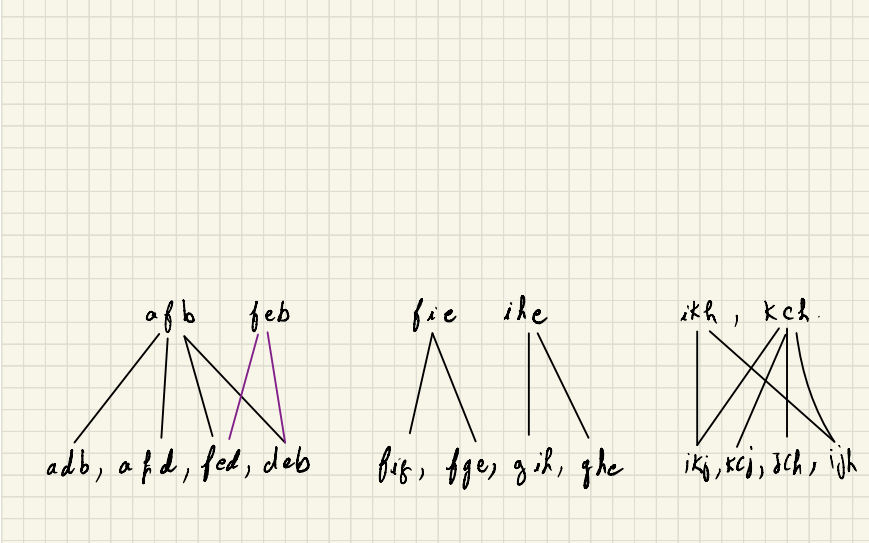

Figure 4: The first hierarchy.

In the new graph S_2 (Figure 3), we have to pick a new independent non boundary vertices and repeat the same procedure. Thus, let $R_2 = \{f, h\}$.

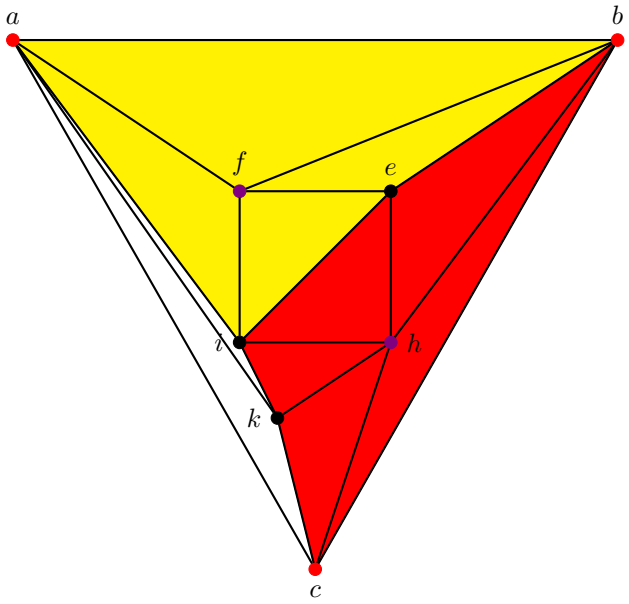

Figure 5: R_2 is in violet.

Now, we have to remove the vertices and retriangulate.

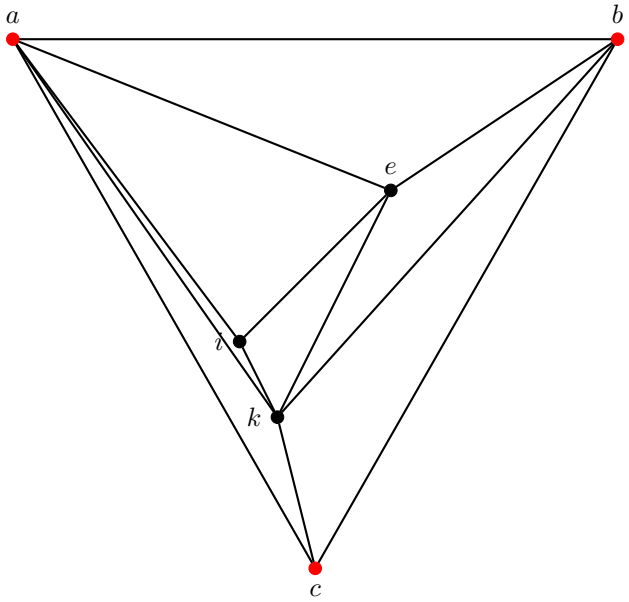

Figure 6: The new graph S_2 after removing R_2.

Now, we can continue building the hierarchy as follow.

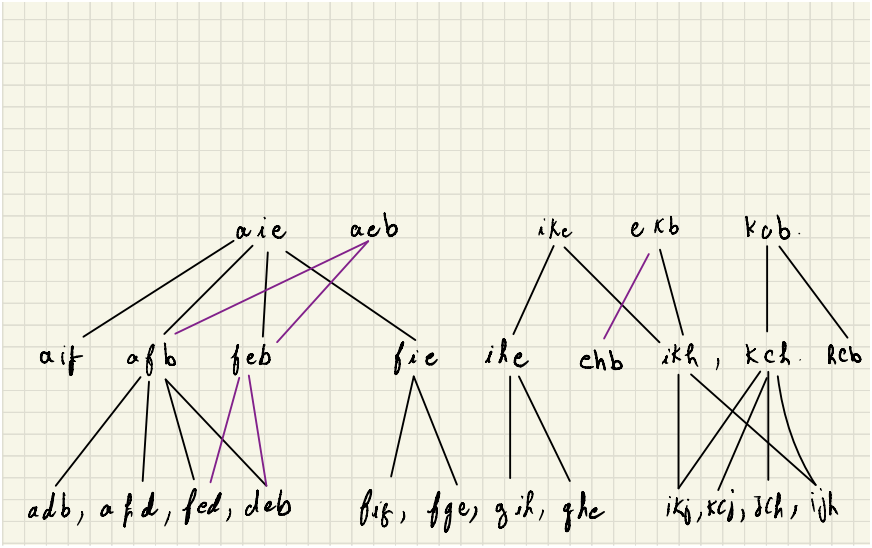

Figure 7: The second hierarchy.

Again, we pick a new independent set from the non-boundary points $R_3 = \{e\}$. Now, compute $S_3 = S_2/R_3$, and triangulate it.

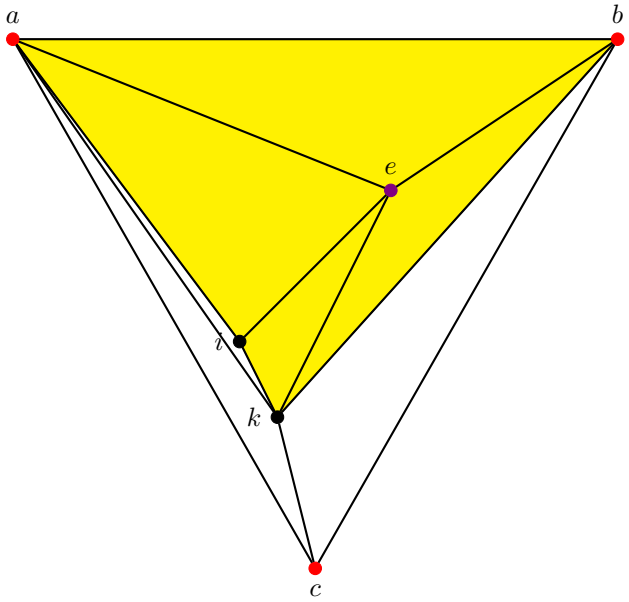

Figure 8: R_3 colored in violet.

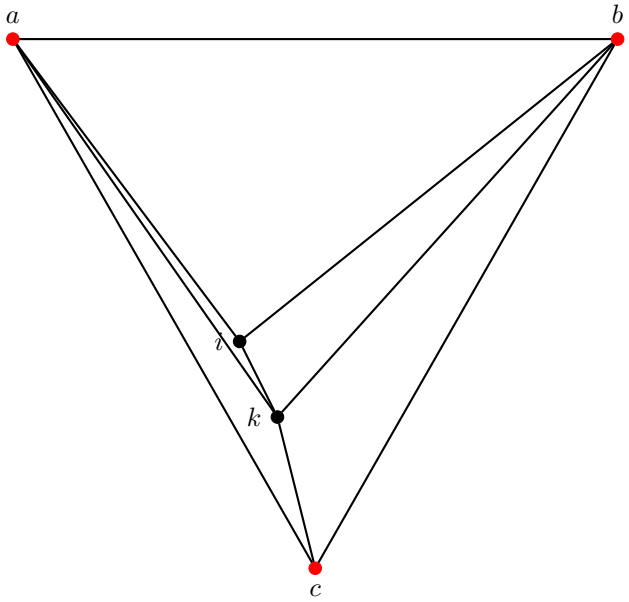

Figure 9: The new graph S_3.

we get the following hierarchy.

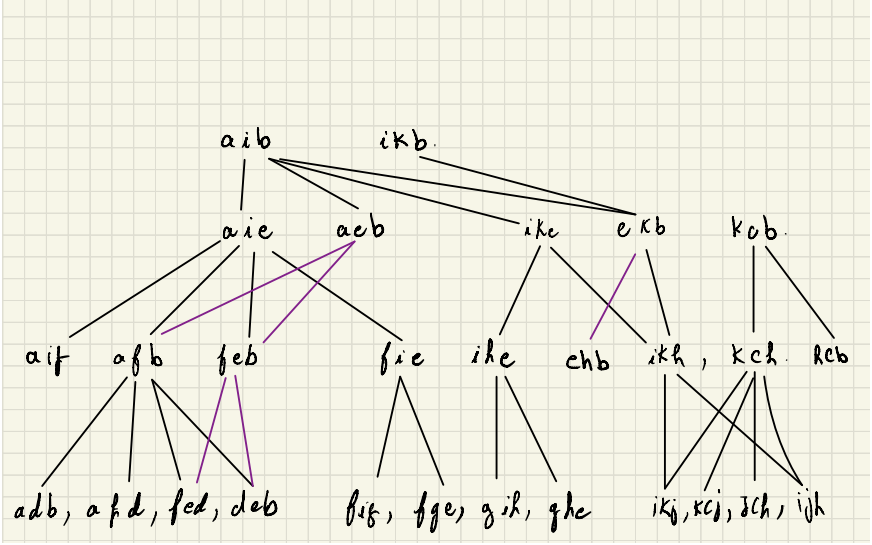

Figure 10: The third hierarchy.

Now, we have $R_4 = \{i\}$.

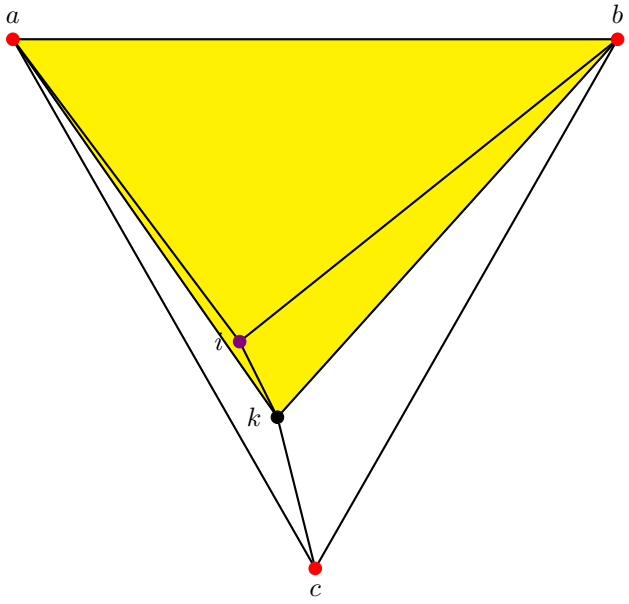

Figure 11: The set R_4 colored in violet.

Remove R_4, since we have a triangle we do not need to triangulate.

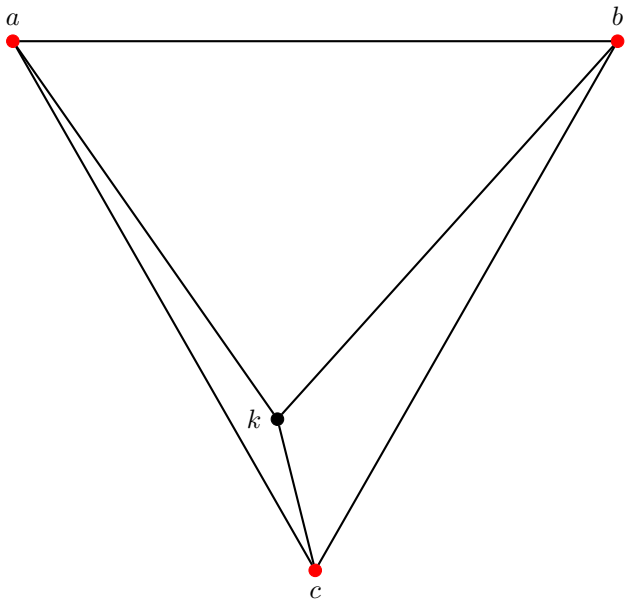

Figure 12: The new graph S_4.

We update the hierarchy to the following triangulation.

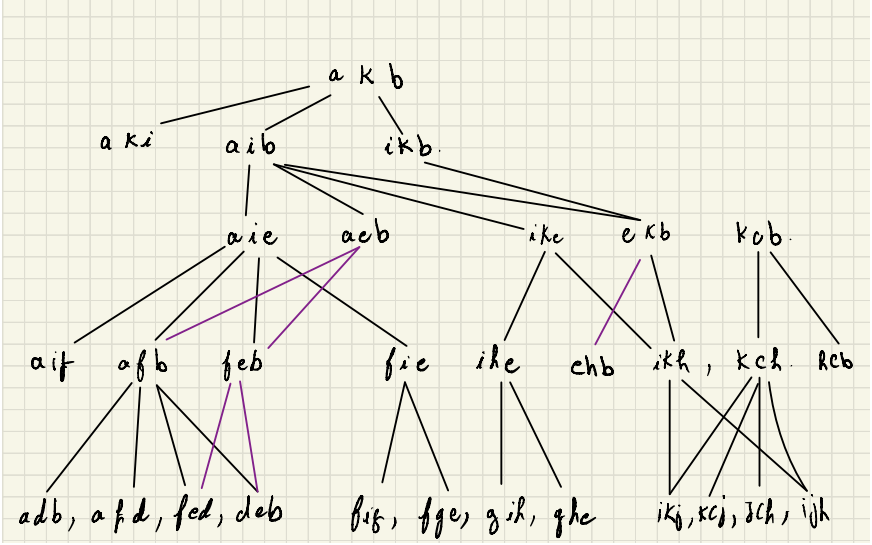

Figure 13: The hierarchy after removing R_4.

Finally, we have $R_5 = \{k\}$.

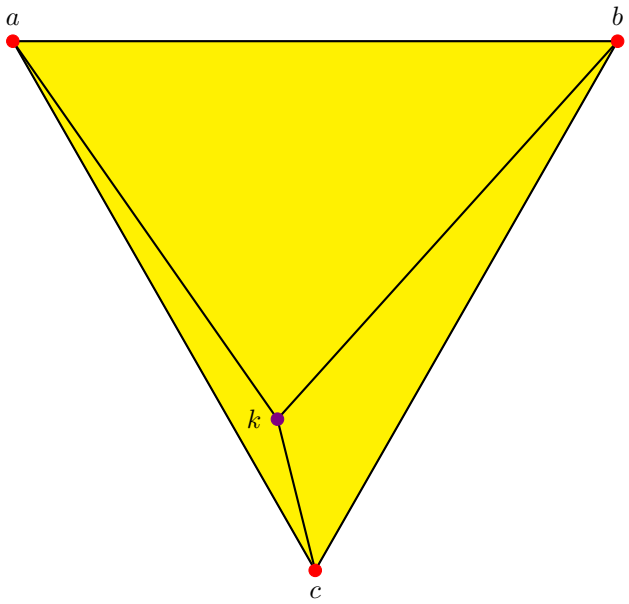

Figure 14: The set R_5 in violet.

Removing R_5 leads to the following.

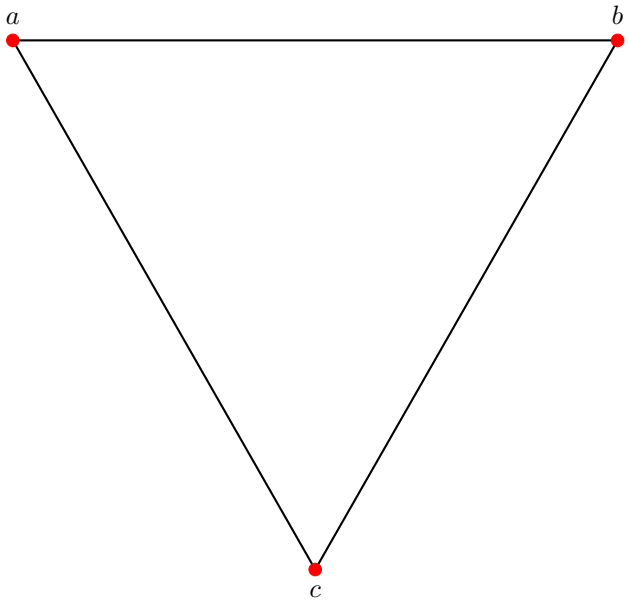

Figure 15: The triangulation S_5.

And, the final hierarchy will be the following.

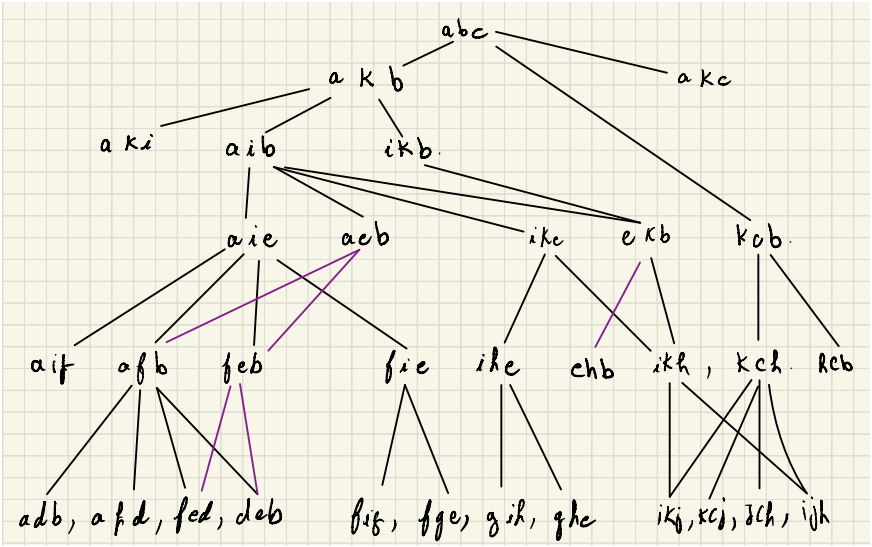

Figure 16: The Final Hierarchy

Query execution: Consider the following example where q is the query points. The query will follow

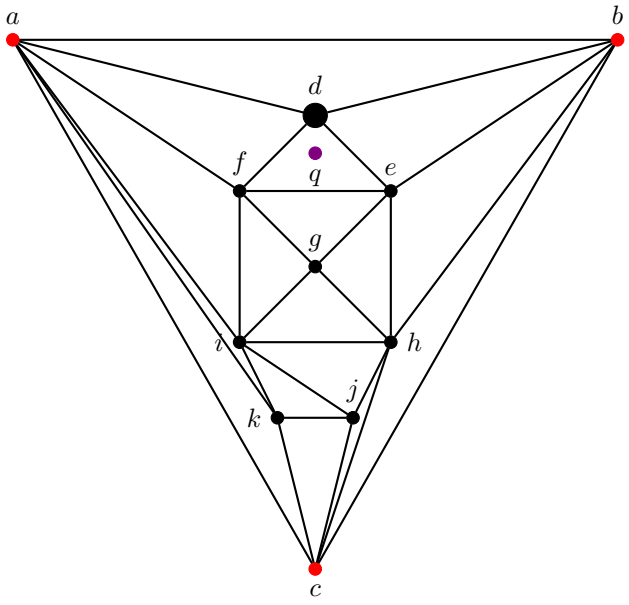

Figure 17: The query point q.

the red line in the hierarchy. It recognises which triangle it belongs to by performing comparisons with the three sides.

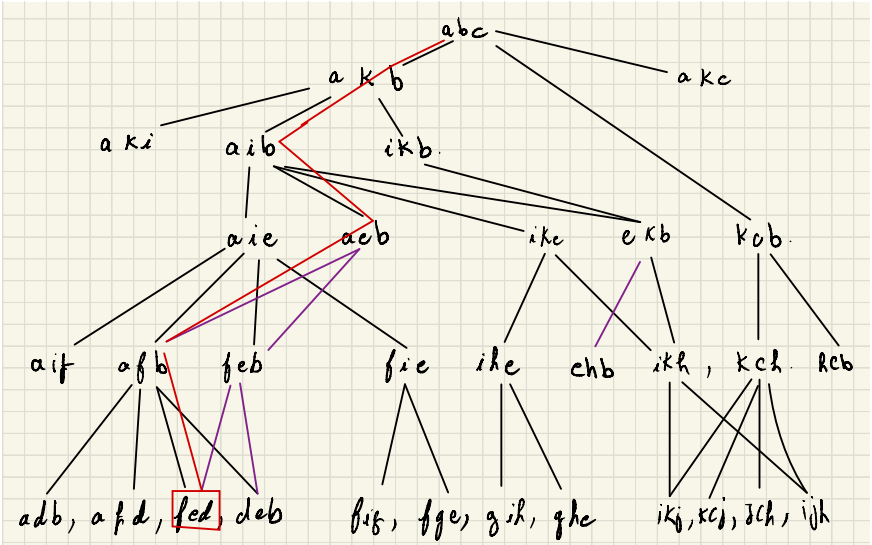

Figure 18: The path that the query algorithm take, highlighted in red.

□

Question 2: The Polygon and the Stick

Assume that you have a simple, n-vertex polyogon P. You are allowed preprocessing. Then, you are to answer efficiently the queries of the following type: for a given point q in P, find the longest horizontal stick containing q that fits inside P.

Then, use the preprocessing as a black box to answer the following two queries.

- Find the shortest horizontal stick that fits inside the polygon.
- Find the longest horizontal stick that fits inside the polygon.

Derive all time and space complexities for Preprocessing and queries.

Solution. The preprocessing will be similar as the one for the slab method or chain decomposition method. We will modify slightly the query. In the slab case, once we reached the slab where the point $q = (x_q, y_q)$ is located, now we have two segments, one to the left of q and one to its right. Then the next step is to compute the intersection of these segments with the line $y = y_q$. And the longest horizontal stick containing q will be the one that join these two intersection points. In the case of the chain decomposition, it will be basically the same, once we locate the two chains that surround the point q, we compute the intersection of $y = y_q$ with the chains and the horizontal stick will be segment that join the two segments on the chains that are immediately to the left and to the right of q. This can be done in $O(\log n)$ once we locate the two chains.

In the first case, the query complexity is $O(\log n)$, the preprocessing complexity is $O(n^2)$. In the second case, the query complexity of $O(\log^2 n)$, the preprocessing complexity if $O(n \log n)$.

Now, to find the longest and the shortest horizontal stick that fit inside the polygon, it is sufficient to query the vertex points of the polygon (here by the shortest, I assume it means the longest shortest horizontal stick, as the shortest stick can be infinitesimally small). Then, record the minimum and the maximum horizontal stick. The complexity of these queries is $O(n \log n)$ or $O(n \log^2 n)$ in the case of slab method and the chain decomposition respectively.

To see why the above is true, we will give the following brief and informal justification, Suppose that you find a horizontal segment that is the longest (similar reasoning could apply in the case of the shortest), and it is not passing by a vertex. Then we can swipe the segment vertically up or down. We have three cases: either it will grow, stay the same, or shrink. If it is shrinking that mean taking it up will make it larger and that is a contradiction. Suppose it is getting larger then it is an obvious contradiction. So the only case that is left to examine is when its length stays constant. That means the segment is moving on two parallel sides and going up or down will eventually reach a vertex as the polygon is a closed curve. Consequently, there is always a largest line that pass by a vertex.

□

Question 3: GIS and The Truckers

You are asked by the City of Ottawa to help with a GIS task. The political side is taken care off. Inside the city trucks are parked on several streets. The trucks may partially or completely block streets. All trucks are to be moved towards the Queensway, You are asked to provide GIS expertise to accomplish the task in as little time as you can (albeit not optimal). How would you aid, what GIS tasks need to be evoked, what data do you need , ...

Solution. In the first step we must assemble the following data. The name of each street, the number of trucks on a given street, the capacity of the street in a unit of time (for example in 10 min, or 1 hour) i.e. how much care we can move trucks on that street in a given time given at a fixed speed.

Furthermore, we must have a map that describes how the streets are connected. For example. Wilginton street is connected to several street such as Elgin, Bank, Metcalf, and O'Connor. We also need to know the driving direction for example vehicle can go from Wilington to O'Connor but not vice versa.

After assembling all the information, the first step is to construct a directed graph $G = (V, A)$ where V is the set of vertices which each represent a street, and A is the set of arcs, directed edges. For instance, we have the following graph.

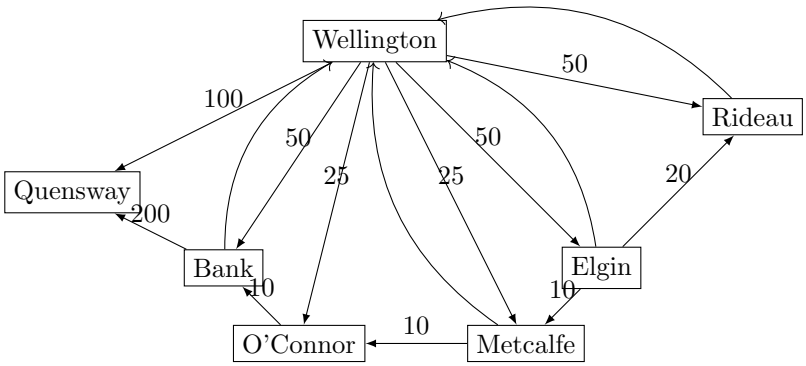

Figure 19: An Example of Graph Representation of the Street of Ottawa. The numbers on the arcs indicates approximately how many vehicle the street can handle in a given time unit.

To solve the problem, we suggest to model it as flow problem. So, we will add an artificial source node, where the outgoing edges of the source node has capacity equals to the number of trucks parked on that street. Moreover, we will let Queensway be the sink node. Now to solve the problem, we can run the most

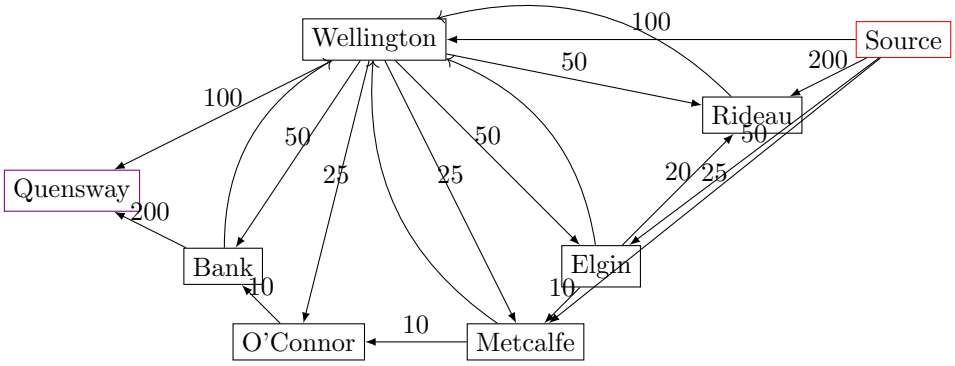

Figure 20: The flow network.

efficient algorithm for Max-Flow. So, we can determine how many car can move in which direction in a given time unit. One advantage, we could exploit, is that our graph most likely is a planar graph. Thus, we might have a faster algorithm on planar graphs. □