Link-cut trees

Eva Rotenberg

References and reading

A Data Structure for Dynamic Trees, Daniel Sleator and Robert Tarjan, 1982

Exercises

1 Space travel

The purpose of this exercise is to train modelling and problem solving for dynamic graphs. (Not as-such relating to the material.)

1.1 You are commanding the space ship Traveller stuck in a distant part of the galaxy. On board, you have n members of the crew. You now want to divide this crew into four shift teams, 1, 2, 3, 4, that work on the space ship at disjoint times of the day. Challenge: you want to avoid putting a person on the same team as one of their arch enemies. Each team should be of at least 1 person and at most $n - 3$ persons, and n is greater than 4. Luckily, due to their advanced star fleet training, each member of your crew is in a conflict with at most 3 different people of the star ship at the same time.

- Show how to model this as a graph problem, and,
- Give an efficient algorithm for solving it.
- What is the running time of your algorithm?

1.2 Now, as time goes by, relationships change, enemies become friends, and vice versa. Still, at all points of time, no person is in a conflict with more than 3 people.

- Show how to model this as a dynamic graph.
- How fast can you perform the necessary update when a conflict disappears?
- How fast can you perform the necessary update when a new conflict arises?
- Is your solution better than assigning people to new teams all over again?

Definition (colouring) A proper k-colouring (or just *colouring*) of a graph is an assignment of colours $1, 2, \ldots, k$ to the vertices of the graph, such that no nieghbours have the same colour. (In other words, no edge is monochromatic).

The following questions relate to the material.

2 Forest of paths

Assume you have a dynamic graph which at all times is a forest of paths.

2.1 show that you can maintain a 3-colouring of the vertices. How many vertices change colour when an edge is inserted? What is the update time?

2.2 Give a data structure that in addition to handeling updates, also answer questions to the *parity of distance* between vertices. That is, given u and v, if they are connected, answer whether they are connected by a path of even or odd length. (Each edge has unit length.)

3 Light height

Given a heavy path decomposition of a tree, define the *light height* of a vertex as follows: if the vertex has no light children, its light height is 0. If it has at least one light child, let max denote the maximal light height of its light children, and set its light height to be max +1.

3.1 What is the maximum value of the light height of a vertex?

3.2 Analyse the link and cut operations: how many vertices can change light height because of one link or cut operation?

3.3 Give an algorithm that updates the light height of vertices in the graph. Can you get poly-logarithmic update time? (Hint: use a binary tree over the light children of a vertex.)

4 Colouring trees

Given a dynamic forest.

4.1 Can you maintain an $O(\log n)$ colouring where only $O(\log n)$ vertices change their colour upon edge-insertion and edge-deletion?

4.2 Can you do so in poly-logarithmic update time?

Definition (bipartite). A graph is *bipartite* if its vertices can be split into two sets in such a way, that each edge has an endpoint in either set. In other words, it admits a proper two-colouring of its vertices.

5 Incremental bipartition testing.

5.1 Show that a forest is bipartite.

5.2 Assume a graph arrives edge by edge. Give an algorithm that reports the first time an edge arrives such that the graph is no longer bipartite.

(Hint: maintain a link-cut tree over a spanning tree of the graph.

Hint hint: See exercise 2.2.)

5.3 [$*$] Discuss ideas for also handling edge-deletions, while still being able to report the first time an edge arrives such that the graph is no longer bipartite. Which new challenges arise?

- Assume you have a oracle who repairs your spanning tree upon edge-deletion, that is, if an edge is deleted from the spanning tree, it gives you a new reconnecting edge from the graph if one exists, in $O(1)$ time. How efficiently can you detect the first edge that violates bipartition? [$*$]