Weekplan: Dynamic Connectivity

Philip Bille Inge Li Gørtz Christian Wulff-Nilsen

References and Reading

[1] Faster Deterministic Fully-Dynamic Graph Connectivity, C. Wulff-Nilsen, SODA 2013: 1757–1769.

We recommend reading [1] until and including Section 3.
Erratum for [1]: on Page 4, line 10, it should read $\log(n(u)/n(v)) + O(1)$.

Exercises

1 Worst-case time

1.1 What is the worst-case update time of the data structure in the slides?

1.2 What is its worst-case query time?

2 Search procedures with distinct number of edges explored In the examples in the slides showing the two parallel search procedures after an edge deletion, they both have the same number of edges to explore (plus/minus 1). This is not true in general.

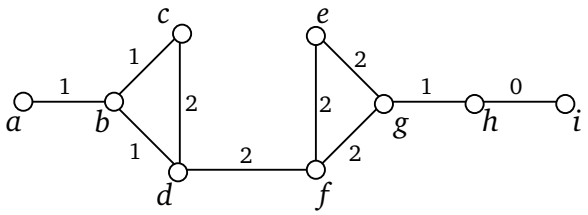
In the general case, let P_u and P_v be the search procedure from C_u and from C_v, respectively. We use the following approach to handle an edge deletion and consider two cases similar to those in the slides:

1. The two search procedures visit the same cluster C_{uv} (the parent cluster is not split). Assume C_{uv} was first visited by P_u (the other case is symmetric). Let $\mathscr{C}_u$ be the clusters visited by P_u. Let $\mathscr{C}_v$ be the clusters visited by P_v, excluding C_{uv}. If $\sum_{w \in \mathscr{C}_u} n(w) \leq \sum_{w \in \mathscr{C}_v} n(w)$, the edges visited by P_u are contracted; otherwise, the edges visited by P_v are contracted.

2. The two search procedures visit disjoint sets of clusters (the parent cluster is split). Assume P_u is the first to finish (the other case is symmetric). Then the levels of edges explored by P_u are increased, *provided* that this does not violate the invariant. If the invariant is violated, then P_v finishes its search (visiting all remaining edges reachable from C_v) and the levels of its visited edges are increased.

2.1 Argue that the invariant is maintained using the above approach and show that the edge level increases suffice to pay for the two search procedures (use the same level of detail as in the slides).

3 Updates to example graph The following example shows a graph G with edge levels indicated.



3.1 Explain why the invariant of the data structure for G is violated.

3.2 Consider a modified version of G where k isolated vertices are added. Show that $k = 11$ is the smallest k needed to satisfy the invariant. What is $\ell_{\max}$ for this value of k?

3.3 Show the cluster forest $\mathscr{C}$ for G with $k = 11$. You may ignore the k isolated vertices in your drawing.

3.4 Show G and $\mathscr{C}$ after `insert`(c, e). What changes occur in $\mathscr{C}$?

3.5 Show G and $\mathscr{C}$ after the additional update `delete`(d, f) (i.e., after `insert`(c, e)).

3.6 Show G and $\mathscr{C}$ after the additional update `delete`(c, e) (i.e., after `insert`(c, e) and `delete`(d, f)).

4 Cluster sizes and edge**-bitmaps**

4.1 Show bitmaps edge(u) and sizes $n(u)$ for all nodes u in the cluster forest $\mathscr{C}$ that you obtained in Exercise **3.3**.

4.2 Show that edge(w) is the bitwise or of edge(u) and edge(v) where $C(w)$ is the merge of $C(u)$ and $C(v)$.

5 Local trees Let node u have children a, b, and c in $\mathscr{C}$ with $n(a) = 1$, $n(b) = 4$, and $n(c) = 5$. Let another node v have children d, e, f, and g with $n(d) = n(e) = 2$ and $n(f) = n(g) = 4$.

5.1 Show local trees $L(u)$ and $L(v)$.

5.2 Suppose an update to G results in merging u and v into u. Using the description in the first paragraph of Section 3.6 of [1], show how to obtain $L(u)$ resulting from the merge.

5.3 Argue that the worst-case time to merge two local trees is $O(\log n)$ and why this is not a problem for the amortized analysis. *Hint:* The number of cluster pairs merged is at least the number of edge level increases.

5.4 An edge deletion in G may also involve splitting a cluster (or multiple clusters) as we saw in the lecture. Explain how to split the corresponding local tree in two in $O(\log n)$ time. Argue that this is not a problem for the amortized analysis. *Hint:* For the last part, show that only $O(\log n)$ clusters can be split per edge deleted in G.

6 Height of $\mathscr{C}_L$ Show that $\mathscr{C}_L$ has height $O(\log n)$ where $n = |V|$. *Hint:* Show first that for a node u and a child v of u in $\mathscr{C}$, v has depth at most $\lg(n(u)/n(v)) + O(1)$ in $L(u)$. Then use a telescoping sums argument.

7 Initial/final graph In [1], it is assumed that we start with $E = \emptyset$.

7.1 Why is this important for the analysis?

7.2 Adapt and analyze the data structure for the case where we start with E arbitrary and end the update sequence with $E = \emptyset$. What is the amortized update time for the adapted data structure?

7.3 How bad can the amortized update time be if we do not assume that E is empty at the beginning nor at the end of the update sequence?

8 Balanced binary search trees As explained in [1], each leaf of $\mathscr{C}_L$ has a balanced binary search tree (BBST).

8.1 Explain how the BBSTs are used by the data structure.

8.2 Explain why for the $O(\log^2 n)$ amortized update bound, these BBSTs are not needed. What simpler data structure can be used instead in each leaf? Try to avoid increasing the $O(m + n \log n)$ space bound.

9 Incremental connectivity In the *incremental connectivity* problem, we only allow insertions. As mentioned in [1], this can be viewed as the union-find problem. Explain why this is the case.