GraphMerge Correctness

Evan Cook

August 8, 2023

1 Preface

This document contains a proof of correctness for the graph merge algorithm related to my concavity algorithm. I felt that the algorithm's correctness was not immediately clear from my description and in-code comments, and thus provide this file.

This proof presumes an understanding of the concavity graph merge algorithm. I will reference the algorithm's functionality on a high level, and various functions/data structures used in the algorithm. Check out Concavity Measure 3D.ipynb for more details on these. It may also help to have the algorithm nearby to look at while reading the proof.

2 Proof of Correctness

Our algorithm is effectively a series of merges performed on our 'hold' list. To produce the final reassignment, we map each segment to its root as determined by the helper function find_root. Segments that are merged together should share the same reassignment ID; segments that aren't merged together should not.

Henceforth, I use the term 'segment i' to refer to the cell segment with ID i in our segmentation. For correctness, it suffices to prove that $\forall i, j$, segments i and j share a root $\iff$ they are to be merged. We use an inductive argument on the number of merges n.

- $\mathbf{n = 0}$: After 0 merges, every segment is separate. The only segment pairs that can be considered 'to be merged' are a segment with itself, i.e. segment i with segment j when $i = j$. Note also that every segment's root is itself – there are no merges. But then we have the following:

  segment i and j share a root $\iff i = j \iff$ segment i and segment j are to be merged

  Thus the result is proven. (Alternatively, if one's idea of merging is between segments i, j s.t. $i \neq j$: Note that prior to any merges, every segment's root is itself. Simply observe that there are no merges and also no shared roots between any pairs i, j s.t. $i \neq j$ since $r_i = i \neq j = r_j$.)

- $\mathbf{n \to n + 1}$: By the inductive hypothesis, suppose that our result holds after n merges. Let our (n+1)th merge be between segment i and segment j, supposing WLOG that i < j. Segments not merged to either i or j are unaffected, and their correctness is preserved. (This is clear – linking i and j does not affect segments not connected to either.) Hence we must only prove correctness of the segments merged to either i or j.

  In particular, the segments already merged in prior steps to i and j should now be merged together. By the inductive hypothesis, the segments previously merged to segment i are **precisely** those that share a root with i and the segments previously merged to segment j are **precisely** those that share a root with j.

Denote r_i to be the root of segment i and r_j to be the root of segment j. Because i < j, our algorithm defines the new root of r_i to be r_j.

All segments previously merged to i and all segments previously merged to j (including i and j themselves) should share a root. Recall the functionality of find_root. If a segment was merged to i in the first n merges, find_root will step to r_i, step once more to r_j, and terminate. If a segment was merged to j in the first n merges, find_root will step to r_j and terminate. Hence all of the above share r_j as a root.

To clarify, I explicitly state both directions of the iff result:

- segment x and y share a root $\rightarrow$ they are to be merged:
  Take **arbitrary** segments x, y that share a root after merge (n+1). Either x, y share a root as a result of the (n+1) merge of i, j, or they already shared a root after merge (n). If they already shared a root after merge (n), they were to be merged after merge (n) [inductive hypothesis] and therefore are still to be merged.

  If segment x and segment j only share a root after merge (n+1), it must be that x shared a root with i and y shared a root with j after merge (n). (Or vice versa – assume the above WLOG). By the inductive hypothesis, x is to be merged with i and y is to be merged with j. But we are merging i with j, and our merge passes through $x \iff i \iff j \iff y$. Thus x and y are to be merged.

- segment x and y are to be merged $\rightarrow$ they share a root:
  Take **arbitrary** segments x, y that are to be merged after merge (n+1). Either x, y were merged via the (n+1) merge of i, j or they already were merged after merge (n). If they were already merged after merge (n), they shared a root after merge (n) [inductive hypothesis] and therefore continue to share a root now.

  If segment x and segment y only are to be merged after merge (n+1), it must be that x was to be merged with i, y was to be merged with j after merge (n) or vice versa. By the inductive hypothesis, x shared a root with i and y shared a root with j. But with merge (n+1), we set the new root of i to the root of j. Hence after merge (n+1), both x and y share a root – namely the root of j.

Hence both directions of our desired iff statement are proved, concluding our inductive step.

Our inductive result proves that after n merges, segments only share a root iff they are to be merged. Set n to be the number of merges carried out via our concavity algorithm. It follows that at its termination, segments to be merged are exactly those that share a root.

At the end of our algorithm, we reassign all segments to their root. Segments that we want merged are precisely those that share a root, which are precisely those that receive the same ID (their shared root) and appear merged in the final algorithm. Thus the final result of the algorithm as defined is proven correct.