15210: Parallel and Sequential Data Structures and Algorithms
thesauruslab
Zikang Wang (zikangw)

6.1

1. Run the given algorithm to find all the shortest paths to the source on $G = (V, E)$.
2. Construct/reorganize a new graph/tree $G' = (V, E')$ from G using the shortest paths, and do the following:
   1) Since path has an order, G' is a directed graph.
   2) Eliminate all the edges that are from a lower-level vertex to a higher-level vertex.
   3) Label each vertex with the length of its shortest paths and denote it as $level(v)$.
   4) Count the number of parents and siblings for each vertex and denote it as $in(v)$ (That is, those from a lower level to higher level).
3. Iterate all edges. For each edge e that is from vertex u to v:
   1) If $e \notin E'$, then it won't affect the shortest paths.
   2) If $e \in E'$:
      a) If $in(v) \geq 2$ (v has in-edge(s) other than e), removing this edge won't affect current shortest paths
      b) If v only has u as parent/sibling, then removing this edge will change the shortest paths. That is, we have to go to a lower level and go back up to visit this vertex, which would increase distance by at least 2 (down + up).
4. From 3, if the removal of any e will affect shortest paths, this graph is not robust (This can be done by simply setting a flag variable).

Step 1 has W_{SP} work. Step 2 can be integrated into the ASP algorithm therefore introduces no additional work or span (we can add operations of constant work/span in each step of makeASP). Step 3 requires a scan on each edge. Necessary information is stored so the time for each edge can be constant (e.g. in a sequence of tuple (vertex, in)), and the iteration of edges can be parallel. Step 4 is constant time. Therefore, the total work is

$$W = \Theta(W_{SP}(G) + m)$$

The span is

$$S = \Theta(S_{SP}(G) + 1 \cdot logm) = \Theta(S_{SP}(G) + logm)$$

Naive Method (PLEASE COUNT WORDS SEPARATELY)

Iterate all edges. Delete one edge and run makeASP to find all shortest paths. Then iterate all paths (that is, all vertices) to see if the length of any of them has increased more than 1.

For each edge, we need to run makeASP once, and then check each vertex in instant time. Therefore, for m edges,

$$W = \Theta(m \times (W_{SP}(G) + n)) = \Theta(m(W_{SP}(G) + n))$$

And the span would be the length of one iteration of edges. There are n vertices and we can check them in parallel, so

$$S = \Theta(S_{SP}(G) + 1 \cdot logn) = \Theta(S_{SP}(G) + logn)$$