Comp 322

HW2 Written

Michelle Pang

1).

```
1   acc ← accumulator (OR, boolean);
2   finish (acc) {
3       for each Assignment f of a value in [k] to each node in V do:
4           colorable ← accumulator ( AND, boolean);
5           async {
6               finish (colorable) {
7                   for each {u,v} ∈ E do:
8                       async {
9                           if f(u) = f(v) then:
10                              colorable.put (false);
11                      }
12              }
13              acc.put (colorable.get());
14          }
15  }
16  return acc.get();
```

2). Let $n = |V|$, $m = |E|$, then there are k^n total mappings for the for-each loop on line 3, and m total mappings for the for-each loop on line 7. So total WORK $= O(k^n \cdot m)$

3). Yes. Since the sequential algorithm can finish without checking all possibilities while the parallel algorithm only finishes when all possibilities are checked, the sequential algorithm may finish execution earlier than my parallel algorithm. Thus it is possible that my parallel algorithm has a larger value of WORK than sequential algorithm.

4). No. Because parallel algorithm has to check all possibilities. So it will always have WORK that is greater than or equal to the sequential algorithm's work.

5). It is possible for my algorithm to exhibit a data race, and that would be benign, because although multiple tasks write to the same location, they all store "false" in the object colorable.