Recitation 13

Hashing

13.1 Announcements

- *DPLab* has been released, and is due **Monday, April 18**. It’s worth 140 points.
- *PASLLab* will be released on Monday, April 18.
13.2 Removing Duplicates

Removing duplicates is a crucial substep of many interesting algorithms. For example, in BFS, consider the step where we construct a new frontier. One viable method would be to generate the sequence of all out-neighbors, and then remove duplicates:

\[ F' = \text{removeDuplicates}\langle v : u \in F, v \in N_G^+(u)\rangle \]

So, how fast is it to remove duplicates? Can we do it in parallel?

13.2.1 Sequential

Before we think about parallelism, we should acquaint ourselves with a good sequential algorithm solving the same problem. This way, we know what to shoot for in terms of work bounds, since we want our parallel algorithm to be asymptotically work-efficient.

**Task 13.1.** Describe a sequential algorithm which performs expected \(O(n)\) work to remove duplicates from a sequence of length \(n\). Also argue that \(\Omega(n)\) work is necessary in order to solve this problem, and conclude that your algorithm is asymptotically optimal.

*Hint: try hashing elements one at a time.*

13.2.2 Parallel

**Task 13.2.** Implement a function

\[
\text{val removeDuplicates : } (\alpha \times \text{int} \rightarrow \text{int}) \rightarrow \alpha \text{ Seq.t} \rightarrow \alpha \text{ Seq.t}
\]

where \((\text{removeDuplicates } h \ S)\) returns a sequence of all unique elements of \(S\), given that \(h(e, m)\) hashes the element \(e\) to a uniform random integer in the range \([0, m)\) (thus the probability of collision for any two distinct elements is \(1/m\)).

*Hint: as a first attempt, try simultaneously hashing as many elements as possible all at the same time. What do you do when elements collide?*