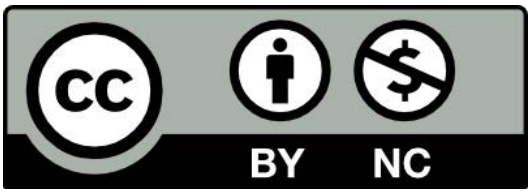

Reuniting Families - A Day One Assignment

Abstract

The *Reuniting Families* "unplugged" assignment is designed for the first day of CS 1 / CS 0. From a broadening participation perspective, what one does on the first day to introduce the discipline can be vital. The goal of this assignment is to illustrate that when doing computer science:

- Algorithms are central to the discipline.
- Algorithm efficiency really does matter, and harnessing parallelism is important when dealing with large data sets.
- CS is typically done in a group setting.
- Computing is socially relevant; its not primarily about controlling animations, game programming, or developing the next killer app. (i.e. The all too frequent avenues for introducing the discipline.)
- Communication skills matter.
- Simplification in developing algorithmic solutions to hard problems is an important problem solving technique.
- Iterative improvement is an important process in evolving solutions.

For the *Reuniting Families* activity, the class is asked to consider a (natural) disaster, e.g. hurricane, tsunami, earthquake, terrorist attack.[1] Furthermore, the location is a smallish city. Regardless of the specifics of the

[1]One might elect to connect to a recent event familiar to the student body. However, sensitivity must be exercised to avoid potentially painful personal connections.

disaster, the local *uncovered* soccer/football/rugby stadium is left intact and the aid workers have directed all survivors to congregate at this single undamaged stadium. It is assumed that the stadium is sufficient to hold all the survivors. After the survivors of a nuclear family[2] have been identified, that family can leave the stadium to their waiting Red Cross tent. The students, working in groups (e.g. 3-4) must devise a protocol, i.e. algorithm, for the aid workers to use to reunite the survivors of each nuclear family unit.

Student teams are given sufficient class time to develop and write up their solutions. Teams are then asked to present their solution to the class where they receive feedback on understandability, feasibility, efficiency, and scalability. (A code/algorithm review.) There are many possible implementation scenarios depending on class size and class schedule; presentations occur on the second day, only a (small or representative) subset of teams present, etc.

To focus on algorithm efficiency, students are told that the survivors are sitting in the stadium under a very hot sun. Hence, not only are survivors anxious to be reunited with their surviving family members, but also need to get out of the sun as soon as possible.

Typically, given how little information is provided at the onset, many clarifying questions get posed. The recommended answers the instructor provides are designed to simplify the problem as much as possible. Questions that often get posed include:

- How many aid workers are there?
  Answer: How ever many you want.
- How many entrances are there to the stadium?
  Answer: How ever many you want.
- How many sections are the in the stadium?
  Answer: How ever many you want.
- What can be assumed about people's names?
  Answer: Using the Latin alphabet, each nuclear family is identified by a distinct (sur)name which all surviving members know and can spell.[3] If

[2] A "nuclear family" in this context are those individuals, regardless of surname, blood and/or legal connections who live together "under one roof."

[3] While nuclear family members may not all share the same surname, for the sake of algorithmic simplicity, there is universal consensus around each nuclear family's identifying name. For example, it may be the surname or Google username of the oldest nuclear family member or its primary breadwinner.

there are n nuclear family units, there are n unique identifying nuclear family (sur)names. Hence, two brothers, each with their own families, would have two different identifying nuclear family (sur)names.

- Is there a working public address system?
  Answer: There is no power, but aid workers can have bullhorns.

- Do survivors have cell phones?
  Answer: Yes, but the towers have been knocked out and/or all the batteries are depleted - hence the phones are of no use.

- If aid workers are told to record the names of each survivor, say upon entry to the stadium, and possibly their seat location, perhaps assigned when entering the stadium, is there a way to automate the sorting and/or searching of this list?
  Answer: No automated processes are allowed. (e.g. Scan in survivor registration sheets, use OCR and then sort the list.) If sorting and/or searching is to be done by aid workers, it must be done by humans using processes described by the group.

- Can signs be made?
  Answer: Yes.

- Can survivors, upon entry to the stadium, be directed to distinct sections of the stadium?
  Answer: Yes, the survivors are capable of following any set of reasonable instructions.

Not only does the above set of simplifications reduce the problem to one the students can solve, but hopefully also illustrate important problem solving techniques: problem simplification/reduction and abstraction.

Student groups are asked to be cognizant of how long it will take to run through their proposed solution with survivor counts of 100, 1000, and 80,000 – how well does the solution scale?

The *Reuniting Families* problem presents a unique first day group assignment, which while not sacrificing coverage of core computing concepts has the potential for making a strong and lasting first impression regarding students' understanding of the importance and relevance of computing.

Moving forward with this exercise one might also require each student, working individually, to write up for the next session, to the best of their

ability, either their favorite algorithm that they heard (which might still be their own) or a new one of their own creation. A followup activity might be to assign a short paper on Google's Person Finder (`https://google.org/personfinder/global/home.html`) which was first used experimentally in Haiti after a 2010 earthquake.

Orthogonally, one might hold a discussion regarding the limitations of the simplifying assumptions made. Which assumption was the most unreasonable? How might one address it? This exercise is useful in illustrating the concept of *iterative improvement*. By what process does one migrate from a version one solution to a version two solution: identifying version one's greatest weakness and attempt to address it.