CS180: Algorithms

Professor Sarrafzadeh

Fall 2019

Contents

CS180: Algorithms

Greedy

- the **greedy** paradigm is a problem-solving approach where the solution set is *greedily* minimized by repeatedly eliminating a possibility from consideration *without global analysis*
  - *pros*:
    - very fast, since there is no need for global analysis
  - *cons*:
    - more difficult to prove correctness for, since there is no global analysis

Famous Problem

- *problem*:
  - a **famous** person is defined as someone who *everyone* else knows, but knows *noone* else
    - the **model of computation (MOC)** or basic set of permitted operations for this problem is asking a *pair* of people at a time if they know they other
    - note that there cannot be two famous people in a room, since they would have to know each other, and that there may not be a famous person in the room
  - find if there a famous person in a room of n people
- *solution #1*:
  1. *repeatedly*, pick an arbitrary person p
  - for every other person p', ask if p knows p'
    - if p knows p', p can't be famous
  - then, for every other person p', ask if p' knows p
    - if p' doesn't know p, p can't be famous
- *analysis #1*:
  - for each candidate, $2 \times (n-1)$ questions are asked
  - in the worst case, $2n \times (n-1)$ questions are asked
  - thus, solution #1 has a complexity of $O(n^2)$
- *optimization*:

- is it possible to improve on solution #1?
    - when considering every pair, there are $\binom{n}{2} \approx n^2$
    - but not *every* pair is needed for an algorithm to be succesful
- use the greedy paradigm, and try to reduce the problem size by one repeatedly
    - ie. eliminate one person from being famous with every question

- *solution #2*:
    1. *repeatedly*, pick two arbitrary people a and b, ask if a knows b
    - if a knows b, then a *cannot* be famous
    - otherwise, if a does not know b, then b *cannot* be famous
    - whatever person is *eliminated* will not be asked picked again in this step
    2. after $n-1$ questions, only one candidate c remains
    - check if c knows noone else ($n-1$ questions)
    - check if everyone else knows c ($n-1$ more questions)
- *analysis #2*:
    - $3 \times (n-1)$ questions are asked
    - thus, solution #2 has a complexity of $O(n)$
    - the **lower bound** for complexity is n since everybody must be asked a question
        - thus, the algorithm is **optimal**