TCSS 343 - Week 6

Jake McKenzie

August 29, 2018

Greedy Algorithms

"We don't much care if you don't approve of the software we write."

...

Eric Hughes

"Greedy algorithms: do the thing that looks best right now, and repeat until nothing looks good anymore or you're forced to stop."

...

Jeremy Kun

"The best programs are the ones written when the programmer is supposed to be working on something else."

...

Melinda Varian

0. Construct a greedy algorith that finds postive integer solution (x, y, z, a, b) for which

$$\frac{1}{x} + \frac{1}{y} + \frac{1}{z} + \frac{1}{a} + \frac{1}{b} = 1$$

Avoid the trivial solution where $x = y = z = a = b = 5$ such that $x \neq y \neq z \neq a \neq b$.
(**Remark:** There are 72 tuples that satisfy this equation.)
This was a question asked to a high school student that a puzzled tutor was desperately asking for help for on Math Stack Exchange. It sort of exploded in popularity and many mathematicians found many greedy approaches in solving it, some centuries old others new! I know of two greedy strategies that I've found but apparently there are many. Time for you to find one!

You want to invite as many people to your party as possible. But, theres a catch: you dont quite trust the other diplomats, all of whom speak multiple languages. So, youd like to make sure that you can understand what everyone is saying at your party. As the Canadian ambassador to Svenborgia, you speak English, French, and Svenborgian. You want to make sure that no two of your party guests speak the same language, other than the three you speak. For each diplomat, you have a list of of every foreign language (i.e., other than English, French, or Svenborgian) that they speak. We refer to this as the International Party Guest, or IPG, problem.

For example: suppose there are three diplomats. If diplomat 1 speaks language 1, diplomat 2 speaks languages 2 and 3, and diplomat 3 speaks languages 1, 3, and 4, we would represent this instance as {{1}, {2, 3}, {1, 3, 4}}. The optimal solution to this instance is to invite diplomat 1 and diplomat 2.

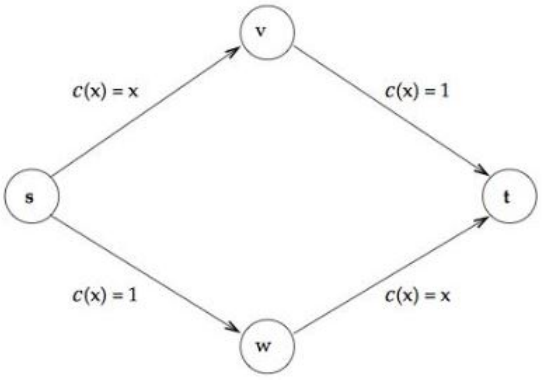
Consider the four following greedy algorithms designed to maximize the number of guests you can invite. For each, determine whether the algorithm is optimal. If it is, briefly sketch a proof of optimality. If its not, give a counterexample(next page).

1. **Greedy Strategy 0:** if no two diplomats speak the same foreign language, invite them all. Otherwise, invite the diplomat who speaks the fewest languages only if they don't share a language with the next diplomat in the invite pool, and recurse.

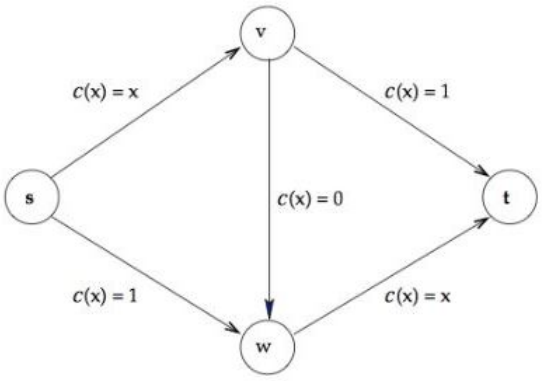
2. **Greedy Strategy 1:** if no two diplomats speak the same foreign language, invite them all. Otherwise, remove the diplomat who speaks more than half the possible languages, and recurse.

3. **Greedy Strategy 2:** if no two diplomats speak the same foreign language, invite them all. Otherwise, remove the diplomat who speaks the most languages, and recurse.

4. **Greedy Strategy 3:** if no two diplomats speak the same foreign language, invite them all. Otherwise, invite the diplomat who speaks the fewest languages, remove all other diplomats who share a language with the one you just invited, and recurse.

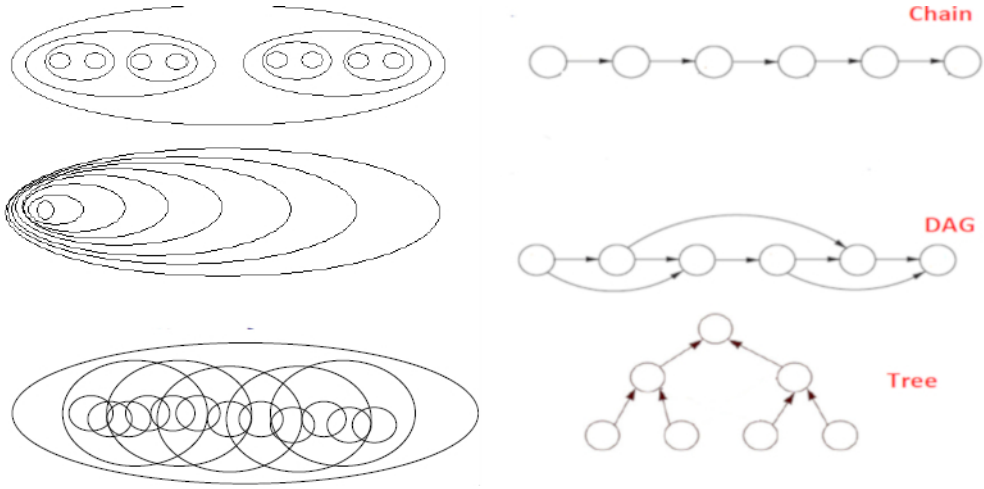


5. We have a group of people commuting home from a vertex s to t. Let the edges with cost $c(x) = x$ be the proportion of drivers traveling along those edges at any given time. Let's say drivers employ a greedy strategy. Find the total cost (units are hours) to get home for the drivers along this graph.



6. Let's say the same group of people were traveling along the same path, but all of a sudden Google invents a teleportation device between vertex v to w which adds no additional time to their commute, what now is the total time it takes to get home for the drivers along this graph?

7. Why must we be careful when implementing greedy algorithms? When do greedy algorithms fail?



8. Above are venn diagrams and graphs that match to the three algorithmic design techniques covered in class so far. Please match each design technique to their venn diagram and graph.

a) **Dynamic Programming:** Overlapping sub-problems, smart brute force, many choices for subproblems

b) **Greedy:** A sub-problem defiines the next one with a single (greedy) binary choice.

c) **Divide and Conquer:** Non-overlapping subproblems, recursion can and is often used.

9. Recall the Huffman greedy encoding algorithm covered in your data structures course. Now consider the fibonacci string
$\mathscr{F}(9)$ ="abaababaabaababaababaabaababaabaab" which was computed by $\mathscr{F}(1)$ = "b" and $\mathscr{F}(2)$ = "a" where $\mathscr{F}(n) = \mathscr{F}(n-1) + \mathscr{F}(n-2)\ \forall\ n > 2$. Find the binary Huffman code for $\mathscr{F}(9)$. What does this tell us about the properties fibonacci string? (**Remark:** I found this example in Maxime Crochemore wonderful book Algorithms on Strings)

A. You are given a sequence of C cookies rated by their tastiness factor. Partition the sequence into segments such that the sum of the tastiness in each gobble full is smaller than a given cap H (handfull factor). Give an algorithm that minimizes the number of segments. (This was a question asked by Facebook on Glassdoor. They wanted it done in better than $n \log n$ but let's stick to a greedy approach)

B. Assume the tastiness factor for the cookies C could be negative, meaning they're fake cookies that are actually frutis and vegetables disguised as cookies. Come up with a sequence of cookies C with a handfull factor H where your greedy algorithm does not come up with the optimal solution.

C. Consider the cashier problem you saw in class, our coin values are: $\{1, 2, 4, 8, 16, \ldots, 2k\}$, for some positive integer k. Give a modified cashier's algorithm that runs in $O(\log n)$ to solve the problem if the value to be paid is $y < 2^{k+1}$.

D. Find which coin you could remove from the U.S. coinage system as to keep the greedy cashier algorithm from returning the optimal result. To show this you need to find the coin and counter example for each.

E. Which coins could we add which would make the coin system in the U.S. MORE canonical, which means simply, it returns even fewer coins as change? (HINT because this is harder: The numbers are prime and less than 11.)

Greedy vertex-colouring algorithm

```
c(v_1) ← 1
for i = 2 to n do
    for j = 1 to i − 1 do
        S ← ∅
        if v_j is adjacent to v_i
            then S ← S ∪ {c(v_j)}
    k ← 1
    while k ∈ S do
        k ← k + 1
    c(v_i) ← k
```

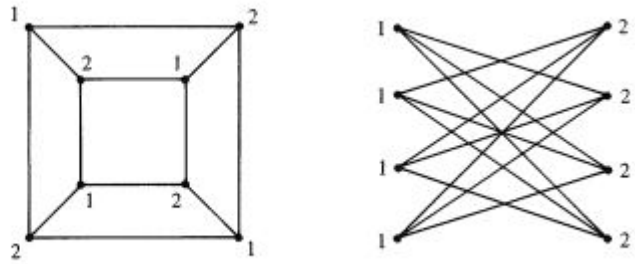


Fig. 8.12
The cube as a bipartite graph.

F. Find the orderings of the vetex of this cube graph (Fig 8.12) for which the greedy algorithm requires 2,3,4 colours respectively.

10. Is this greedy strategy optimal, that is to say does it use the minimum number of colours to colour all vertices?

11. Let $G = (V, E)$ be an undirected graph. A vertex cover for G is a subset $S \subseteq V$ such that every edge in E is incident to at least one vertex in S. Consider the following algorithm for finding a vertex cover for G. First, order the vertices in V by decreasing order of degree. Next execute the following step until all edges are covered. Pick a vertex of highest degree that is incident to at least one edge in the remaining graph, add it to the cover, and delete all edges incident to that vertex. Show that this greedy approach does not always result in a vertex cover of minimum size.

12. Consider the Gerrymandering problem: Suppose we have a set of n precincts $P_1, P_2, \ldots, P_n$, each containing m registered voters. Were supposed to divide these precincts into two districts, each consisting of $\frac{n}{2}$ of the precincts. Now, for each precinct, we have information on how many voters are registered to each of two political parties. Well say that the set of precincts is susceptible to gerrymandering if it is possible to perform the division into two districts in such a way that the same party holds a majority in both districts

Give an algorithm to determine whether a given set of precincts is susceptible to gerrymandering; the running time of your algorithm should be polynomial in n and m

Example: Suppose we have $n = 4$ precincts, and the following information on registered voters: Precinct 1 has 55 voters for party A, 45 voters for party B; Precinct 2 has 43 for A, 57 for B; Precinct 3 has 60 for A, 40 for B; Precinct 4 has 47 for A, 53 for B.

This set of precincts is susceptible since, if we grouped precincts 1 and 4 into one district, and precincts 2 and 3 into the other, then party A would have a majority in both districts.