CS 170
Fall 2016

Efficient Algorithms and Intractable Problems
Christos Papadimitriou and Luca Trevisan

Final Project

Horsing Around

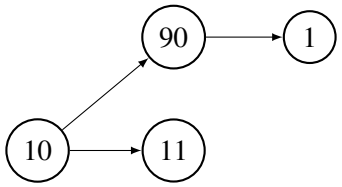
Lucos Papasan is entering a horse race! He has m horses, and he must divide them up into relay teams (of not necessarily equal size) to enter in the competition. In the race, each team runs a total 10000 m and the fastest team wins. Larger teams will perform better, because each horse will have to run less distance. The goal is to find teams to maximize Lucos's chance of winning with any team.

Lucos, being a specialist in horses, has created a model that will determine the likelihood of a given team winning. His overall chance of winning is the sum of each individual team's likelihood of winning. Each horse has a *performance rating* p_i that models how fast the horse is. Lucos's model has determined that the likelihood of winning for a team of n horses is given by

$$\left(\sum_{i=1}^{n} p_i\right) n$$

The factor of n is due to the fact that horses in larger teams have to run less distance, and so they can run faster.

There is one more constraint. Each horse will only race if it runs before one of its friends in the relay. If a horse refuses to race, then all of your teams will be disqualified, making your likelihood of winning 0. Luckily, Lucos has a list of ordered pairs of horses (h_1, h_2), where h_1 considers h_2 to be its friend (note that friendships are not necessarily reciprocal). We can formulate this problem as a graph $G = (V, E)$ where the vertices are the horses, and an edge (h_1, h_2) exists if h_1 can come before h_2 in a relay team. A relay team is represented by a path in the graph. For example, given this graph,



the best teams we can make are $(10, 90, 1)$ and (11) which gives a total score of $(10+90+1)\cdot 3+11 = 314$. Each node in the graph can be only used once; each horse must be assigned to exactly one team.

Your goal is to find a set of relay teams that maximizes the likelihood of winning the race.

Constraints

Suppose that the graph (directed) is $G = (V, E)$. We have the following constraints:

- $0 < |V| \leq 500$
- $(v_1, v_2) \in E$ iff v_1 can come before v_2 in a relay team.
- Each v_i has performance rating p_i.

Goal

Partition V into lists of vertices $V_1, \ldots, V_k$ such that

- The V_i's are disjoint as sets, i.e. $V_i \cap V_j = \emptyset \; \forall i \neq j$
- The union of the V_i's covers V, i.e. $V = \cup_{i=1}^{k} V_i$.
- Each V_i is a path in G, i.e. if $V_i = (v_1, \ldots, v_{n_i})$, then $\{(v_1, v_2), \ldots, (v_{n_i - 1}, v_{n_i})\} \subset E$.
- $\sum_{i=1}^{k} s(V_i)$ is maximized, where $s(\cdot)$ is the likelihood score of the relay team V_i.

Deliverables

Phase 1: Input Files: Due November 21, 10:00 PM

You will create three hard instances of **HORSE**. Make sure to follow the input format exactly, or your file will be discarded and you will receive 0 points. Each input file should be formatted as follows:

- The first line contains an integer which is the number of vertices $|V|$
- The remaining $|V|$ lines are the adjacency matrix, where $\text{adj}[i][j] = 1 \iff (i, j)$ is an edge. The entries of the matrix are separated by spaces.
- The diagonal entries of $\text{adj}[i][j]$ contain the performance rating of the ith horse, which is an integer in $[0, 99]$

You will submit your instance files through glookup. In your submission include your three files, labeled 1.in, 2.in, 3.in. Also, submit a README file containing the names, Berkeley emails, and student IDs of your group members. Put one student per line and separate each entry by a comma and a single space. Make sure to run the input validator and README validator. If you are confused about the input/README formats, see the attached examples.

Phase 2: Output Files, Code, Write-Up: Due December 5, 10:00 PM

After Phase 1, we will release the input files generated by each group on Piazza. The input files will be named "i.in", where i is the number of the instance. You should expect around 200-400 input files. You are responsible for generating a solution for each instance. You will submit your solutions as a single output file, where the ith line is your solution to the ith instance. Each solution should be represented a list of paths, each separated by a semicolon. The path is represented as a sequence of vertices, separated by spaces. For example, if the paths are $(1,4,5)$ and $(2,3,6)$, your output would be `1 4 5; 2 3 6`. We will release an output format validator at the start of phase 2.

Submissions will be done through glookup. In your submission, include your output file and your code, as well the same README file from phase 1. You will also submit a final report over Gradescope as a team. In at most one page, explain at a high level what strategies and optimizations you used. Cite any outside resources you used.

Grading

You will be graded on the following items. The project is worth 2 homeworks and cannot be dropped.

- Valid input files (20 pts)
- Quality of algorithm and results compared to other teams (70 pts)
- Final Report (10 pts)

We will also give bonus points to teams who submit especially hard instances.

Advice

- If you're stuck, try coming up with a greedy strategy that produces *some* answer, and then work on improving that answer.
- Have your algorithm make *randomized* choices: you can then run your algorithm many times and take the best solution.
- You **DO NOT** need to prove that your algorithm achieves a given approximation ratio. Any provable approximation ratio will be very bad.

Miscellaneous Rules and Reminders

- You may work in teams of at most 4.
- You may use any language, publicly available libraries, and research papers, but make sure to cite your sources if you decide to copy something or use a published method.
- If your input file or output file does not conform to the specifications, you will not get any credit for them. Make sure you carefully read the specifications and run the verification scripts before you submit your files to make sure they are correct. There is always one group who doesn't run the script and submits incorrect files, don't be that team.
- The staff is unable to give very involved technical support with specifics on implementation. Choose a language that you are comfortable debugging in. We will help, however, with running through general approaches and strategies if you'd like.