CSC 4120 Project - Pandemic Control

Tian Lei	Bao Jingzhi	Liu Diante
121090515	121090003	119010187

The Chinese University of Hong Kong, Shenzhen

Date: December 17, 2022

1 Part 1

1.1 Theory Questions

1. **Proof.** First, we consider a no-vaccinated scenario. Before the move i ($i > 0$), the virus spread through levels $0, 1, \ldots, i-1$. The vertices in the i-th level have been infected during i-th round. This can be easily proved rigorously by induction. Hence if one vertex v at level $< i$ can be vaccinated, it means at least one of its ancestors has been vaccinated. This kind of vaccination is meaningless because v and its children will never be infected. Now we can conclude that **the government should always vaccinate a vertex at level $\geq i$ in the move i.**

   Then we prove in round i, vaccinating a vertex at level i is the best strategy. Without of loss of generality, suppose in the move i, we vaccinate a vertex u at level k ($k > i$). Instead of vaccinating vertex u, consider u's ancestor (e.g. v) at level i.

   Case 1. If v will not get infected after move i. It means an ancestor of v blocks the virus above. So vaccinating u is also meaningless.

   Case 2. If v will get infected after move i. Absolutely, vaccinating v is better than u because u is a subtree of v.

   Hence we can say that **vaccinating a vertex at level i is better than a vertex $> i$.**

   Overall, the government should always vaccinate a vertex at level i in the move i. □

2. By counterexample. We follow the schema below to construct a tree that the guaranteed fraction c can be arbitrarily close to 0:

   1) The root vertex has 3 children.
   2) For level $i \geq 1$, every vertex has 2 children except the leftmost vertex has 3.
   3) Repeat step 2) until the tree reaches any given level n.

   Greedy Algorithm

   In level $n \geq 1$, the leftmost vertex is designed to trap the `Greedy Algorithm`. The total number of

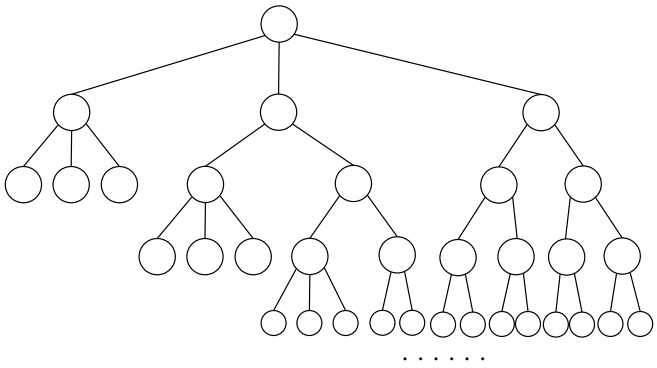


Figure 1: Tree construction

uninfected vertices with level $\leq n$ after round n is $1 + 4(n-1)$.

In particular, the level n ($n \geq 2$) has $5 + 2^{n-1}$ vertices in total, and only 4 of them will not be infected.

Hence, the proportion of saved vertices after round n ($n \geq 2$) is

$$\frac{1 + 4(n-1)}{4 + \sum_{i=2}^{n}(5 + 2^{i-1})} = \frac{4n - 3}{5n - 3 + 2^n},$$

which goes to 0 as $n \to \infty$.

Optimal Algorithm

In each move, we vaccinate the rightmost vertex in the corresponding level. The spread of the virus ends by move 3, and only 9 vertices get infected.

Therefore, we can conclude that

$$c = \frac{S_{greedy}}{S_{optimal}} = \frac{1 + 4(n-1)}{\sum_{i=2}^{n}(5 + 2^{i-1}) - 5} = \frac{4n - 3}{5n - 12 + 2^n}$$

after move n ($n \geq 2$). $\lim_{n \to \infty} c = 0$.

Visualization

Note: In the figures below, we use green nodes to denote the vaccinated people and red notes to indicate the infected people.

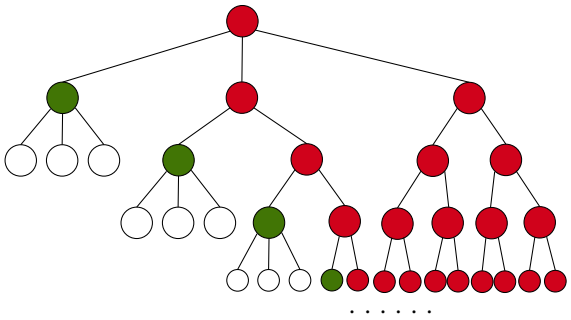


Figure 2: Greedy Algorithm

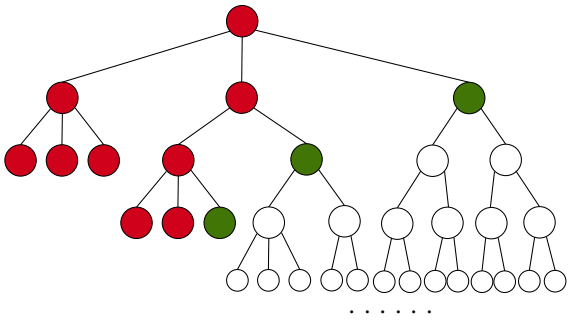
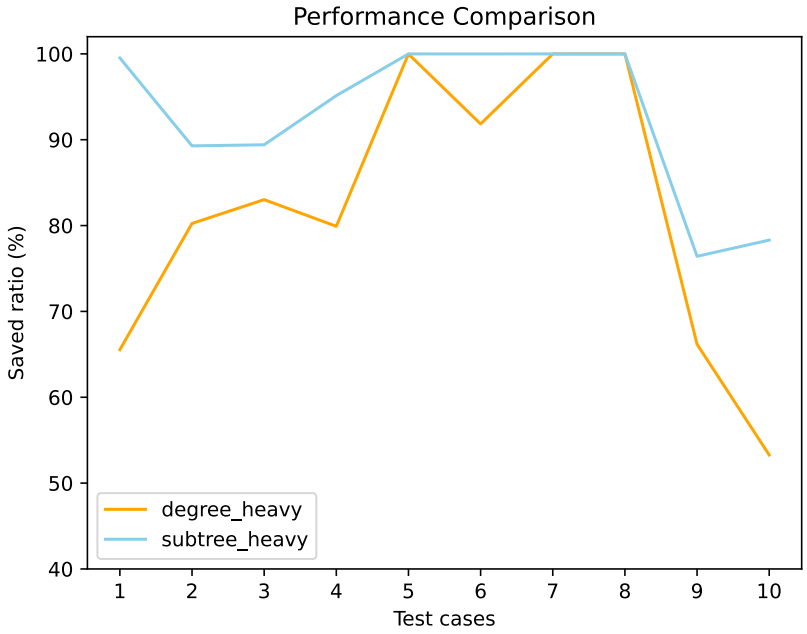


Figure 3: Optimal Algorithm

3. We test our algorithms on the given data. The performance of the two algorithms is shown in the figure below. The raw data of this figure can be found in the attachments.



4. **Proof.** For convenience, we use G_i to denote the node vaccinated by the **greedy algorithm** at level i, O_i to denote the node vaccinated by the **optimal algorithm** at level i, and $w(u)$ to denote the weight(size) of the subtree rooted at u.

   We divide the optimal strategy (that saves the most number of persons) into two parts according to the position of O, denote as $S^A_{optimal}$, $S^B_{optimal}$.

   More specifically,

   $$S^A_{optimal} = \sum w(O_A), \text{ where } O_A \text{ is the descendent of some } G_k \text{ (including } G_k = O_A),$$
   $$S^B_{optimal} = \sum w(O_B), \text{ where } O_B \text{ is not a descendent of any } G_k \ (= S_{optimal} \backslash S^A_{optimal}).$$

   Then we analyze $S_{optimal}$ and S_{greedy} for any given results generated by the optimal and greedy algorithms.

   Obviously, $S^A_{optimal} \leq S_{greedy}$, because for any vaccinated node O_A, its ancestor has been vaccinated by the greedy algorithm.

   On the other hand, for any vaccinated node in Part B at level i (namely O_i), by definition, the node O_i will be infected after turn i by both the greedy algorithm and optimal algorithm. In other words, the node O_i is a candidate for the greedy algorithm. While $G_i \neq O_i$, by the schema of greedy algorithm we can conclude that the weight(size) of G_i is greater than or equal to O_i. Hence, the greedy algorithm always does better for any O_i (in part B) at any level. So $S^B_{optimal} \leq S_{greedy}$.

   From what we discussed above, $S^A_{optimal} + S^B_{optimal} \leq 2S_{greedy}$. Equivalently, $c = \frac{S_{greedy}}{S_{optimal}} \geq \frac{1}{2}$. $\square$

1.2 Demo Output

The output is on *testCase.txt*.

```
------------------------degree-heavy------------------------
Inatially(round 0), 1 is infected.
round 1: vaccinate node 2, saved 1 in total until this round.
round 2: vaccinate node 8, saved 6 in total until this round.
round 3: vaccinate node 12, saved 9 in total until this round.
Finally, saved 9 in total.
------------------------------------------------------------

------------------------subtree-heavy-----------------------
Inatially(round 0), 1 is infected.
round 1: vaccinate node 3, saved 1 in total until this round.
round 2: vaccinate node 4, saved 4 in total until this round.
All people in the levels below are safe. No need to vaccinate.
Finally, saved 7 in total.
------------------------------------------------------------
```

Visualization: (the detection number increases from top to bottom from left to right)

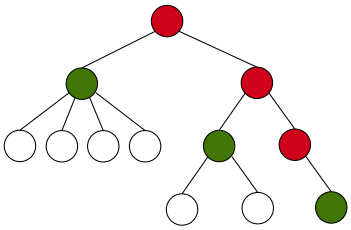


Figure 4: Greedy heavy algorithm demo

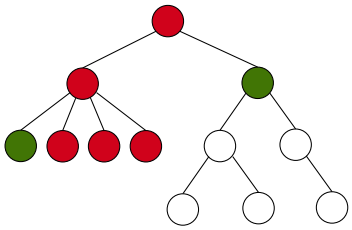


Figure 5: Subtree heavy algorithm demo

Source code: `/phase1/degree-heavy.py`, `/phase1/subtree-heavy.py`, `/phase1/test.ipynb`.

2 Part 2

1. See the source code in the attachments.

   For convenience, we provide a jupyter notebook file `run.ipynb` with annotations.

2. By definition in description, the 2×2 (average) payoff matrix is

$$\begin{bmatrix} (\pi^R_{(SH,SH)}, \pi^B_{(SH,SH)}) & (\pi^R_{(SH,SS)}, \pi^B_{(SH,SS)}) \\ (\pi^R_{(SS,SH)}, \pi^B_{(SS,SH)}) & (\pi^R_{(SS,SS)}, \pi^B_{(SS,SS)}) \end{bmatrix}.$$

   The average result of 100 experiments (based on 10 test data with random coloring) shows

$$\begin{bmatrix} (3.76, -3.76) & (-293.005, 293.005) \\ (285.845, -285.845) & (4.05, -4.05) \end{bmatrix}.$$

3. According to the definition of Nash equilibrium, $\{SH, SH\}$ is not a Nash equilibrium because each one can change its strategy to SS to gain a better payoff. With the average payoff matrix above, only $\{SS, SS\}$ is a Nash equilibrium because the payoffs account for only about 1% of $\{SH, SS\}$ and $\{SS, SH\}$. Within the reasonable sample error deviation, $\{SS, SS\}$ reaches an equilibrium.
4. The total # of saved individuals when i) players choose the Nash equilibrium strategy ($\{SS, SS\}$) and when ii) both players cooperate and play the SH strategy ($\{SH, SH\}$) is displayed in the table below.

Strategy pair	**total # of saved person**
(SH, SH)	20761850
(SS, SS, R)	16357052
(SS, SS, B)	16589150

What we observed above is that the $\{SH, SH\}$ strategy saves more total # of individuals. Based on the data in experiments, the result of $\{SH, SH\}$ strategy is 1.25x of the Nash equilibrium strategy ($\{SS, SS\}$), which validates the 'sad' fact: although cooperation (i.e., altruism) by both players benefits society, and each saves a more significant number of same-color individuals, the fact that they are selfish leads to an outcome where society suffers the most.

3 Attachments

3.1 Data and Code

Project report materials

3.2 Algorithms performance comparison results

Test Case	**Saved ratio of degree heavy algorithm** (%)	**Saved ratio of subtree heavy algorithm** (%)
Case 1	65.539503208283	99.523678305712
Case 2	80.246398832552	89.284096330478
Case 3	83.020400825575	89.410308622555
Case 4	79.916085048664	95.134653635724
Case 5	99.990168579938	99.993638492901
Case 6	91.837727086464	99.991258024860
Case 7	99.995559936418	99.996447949134
Case 8	99.996179323454	99.996179323454
Case 9	66.177166413066	76.418546545075
Case 10	53.279237761348	78.304754096043

3.3 SH and SS strategy implement results

The final test data is the given 10 test data with random coloring.

Format: `(Strategy of R, Strategy of B, The first)`.

File name: `phase2/generated/testCase[case_num].[sub_case].txt`.

Startegy pair	**Average payoff of A**	**total # of saved person**
(SH, SH)	3.76	20761850
(SS, SS, R)	50.78	16357052
(SS, SS, B)	-42.68	16589150
(SS, SH, R)	334.44	18032200
(SS, SH, B)	237.25	19440941
(SH, SS, R)	-265.27	19073395
(SH, SS, B)	-320.74	18168894