Homework #4 (Hand-Written)

Purple Correction Date: 12/14/2022 14:00
Blue Correction Date: 12/6/2022 00:00
Red Correction Date: 12/4/2022 00:00
Due Time: 2022/12/20 14:20
Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **two hand-written problems**.
- You should upload your answer to **Gradescope** as demonstrated in class. For each sub-problem, please label (on Gradescope) the corresponding pages where your work shows up. NO LATE SUBMISSION IS ALLOWED.
- **Collaboration policy.** Discussions with others are strongly encouraged. However, you should write down your solutions **in your own words**. In addition, for **each and every** problem you have to specify the references (e.g., the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem. You may get zero point due to the lack of references.

Problem 1 - Laser Tank (Hand-Written) (30 points)

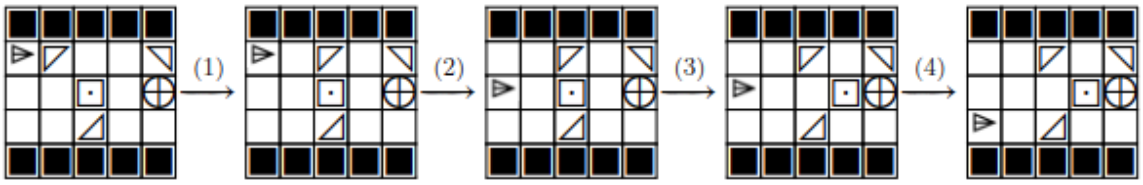
Laser tank is a turn-based puzzle game. You can get a playable version here. Just like many other puzzle games (e.g., tetris and minesweeper), deciding if a laser tank puzzle is solvable is NP-hard. In the following section, you'll reduce a 3-CNF satisfiability problem instance to a puzzle of this game and prove it's NP-hard.

Introducing the game

Laser tank is a turn-based puzzle game played on a 2D grid (the *board*). Each turn, the player can either move the tank or fires a laser. The laser interacts with different *pieces* on the board, and the goal is to hit a certain *piece* with the laser. The *pieces* we use are *mirrors* (the right-angled triangles), *solid blocks* ■, *movable blocks* ⊡, the *tank* ⊳, and the *goal* ⨁.

Tank (⊳) is the only *piece* controlled by the player with two actions: move or fire. In our version, the movement of tank is restricted to vertical axis (i.e., the tank can only move up or move down instead of all four directions). The tank can fire a laser from the front (i.e. to the right). If the laser hits a mirror on a slanted edge, it is reflected. When a mirror is hit on one of the two (non-reflective) short edges by the laser, the mirror is pushed along the direction of the laser. A movable block is pushed one step if it is hit by the laser. A movable block or a mirror is only pushed if the tile directly behind it is empty. The aim of the puzzle is to hit the goal piece with the laser. The solid blocks do not allow lasers or the tank to pass through and they do not move when hit by the laser.

Here is a small instance of the problem, with a step-by-step solution. The tank fires a laser which moves a mirror (1), then takes moves one step down, (2). It then shoots a laser at the movable block (3), and finally moves in position to have a shot of the goal (4). At this moment, emitting a laser solves the problem.



Gadget

A gadget is a small part of the board with some tiles specified to be input tiles and output tiles. For convenient, please follow the below icon convention. A tile is said to be shootable if the tank can shoot the position with movements but without pushing any piece in advance (e.g. in the illustration above, the goal is not shootable before step 1, but is shootable after step 4).

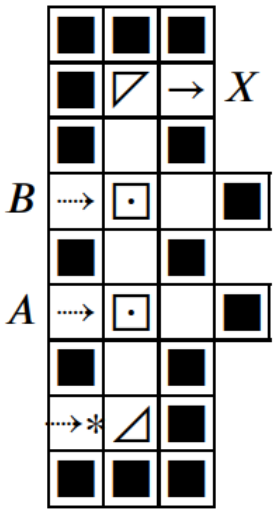
1. Use dotarrows to indicate input.
2. Use dotarrows with star to indicate tiles always shootable.
3. Use arrows to indicate output.

The subproblems

A blank page filled up with grid is provided, you can print it out and draw your answers on it to save time. In the "gadget construction" subproblems, please draw your construction clearly and briefly explain you answer **within 10 lines**.

(0) Example subproblem: AND Gadget (0 point)

Please design a gadget with two input tiles and one output tile such that the output tile can become shootable if and only if both input tiles are shootable. This gadget would serve as the "AND" gate for 3-CNF problem.
Example answer:



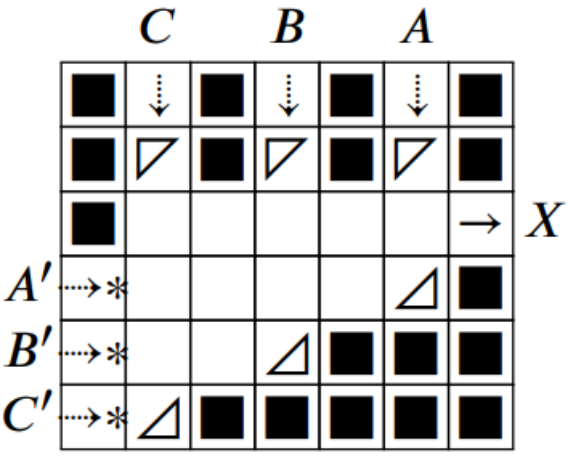
(1) OR Gadget (4 points)

Please design a gadget with three input tiles and one output tile such that the output tile can become shootable if and only if at least one among the three input tiles is shootable. This gadget would serve as the "3-way OR" gate.

Solution

The original idea of this problem came from the paper LaserTank is NP-complete.

(1)



"if": Suppose the shootable input tile is A, Shoot at A one time, the mirror would be pushed down and forming path from A' to X. Then X can be shooted from A'. If the shootable input tile is B or C, apply the same arguments with A' replaced with B' or C' correspondingly.

"only if": Lasers from always shootable tiles can only hit solid blocks after reflections if none of A, B, and C is shootable.

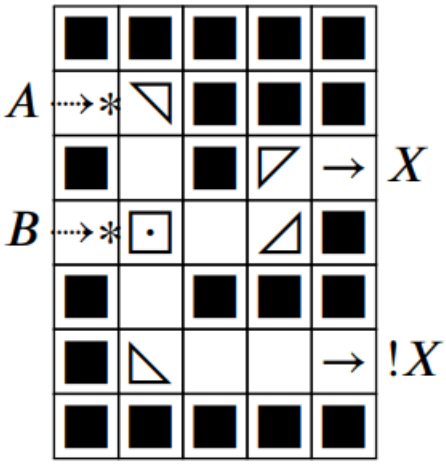
(2) Literal Gadget (4 points)

Please design a gadget with two output tiles such that

- Both tiles can become shootable from the beginning state.
- Once one output tile becomes shootable, the other tile can no longer be shootable in the future.

This gadget guarantees `(X || !X) == true` and `(X && !X) == false`.

Solution



To make X being shootable, first shoot at A, then X can be shooted from B and the path from A or B to !X is blocked by the movable block forever since it can't be pushed away.

To make !X being shootable, first shoot at B, then !X can be shooted from A and the path from A or B to X is blocked by the movable block forever since it can't be pushed away.

(3) Switch Gadget (2 points)

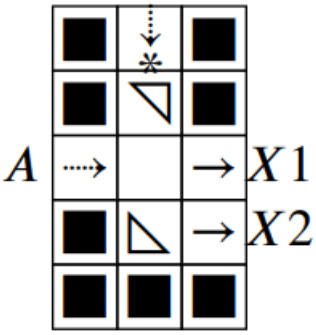
Please design a gadget with one input tile , one always shootable tile and two output tiles such that

- In the begining: The first output tile is shootable if and only if the input tile is shootable, while the second output tile is not shootable.
- Once after the always shootable tile is shooted: The first tile is not shootable, while the second tile is shootable if and only if input tile is shootable.

This gadget enables you to reuse the "shootability" of an output tile from other gadget.

From here on, please do **NOT** draw the details inside the gadgets. To utilize a gadget, just draw a box and label the name of the gadget and specify I/O tiles. You can cite the above-mentioned gadgets even if you didn't answer those subproblems.

Solution



Originally, shoot from A leads to X1. Once the always shootable tile is shooted, the path to X1 is blocked and shoot from A leads to X2.

(4) Practicing Laser (4 points)

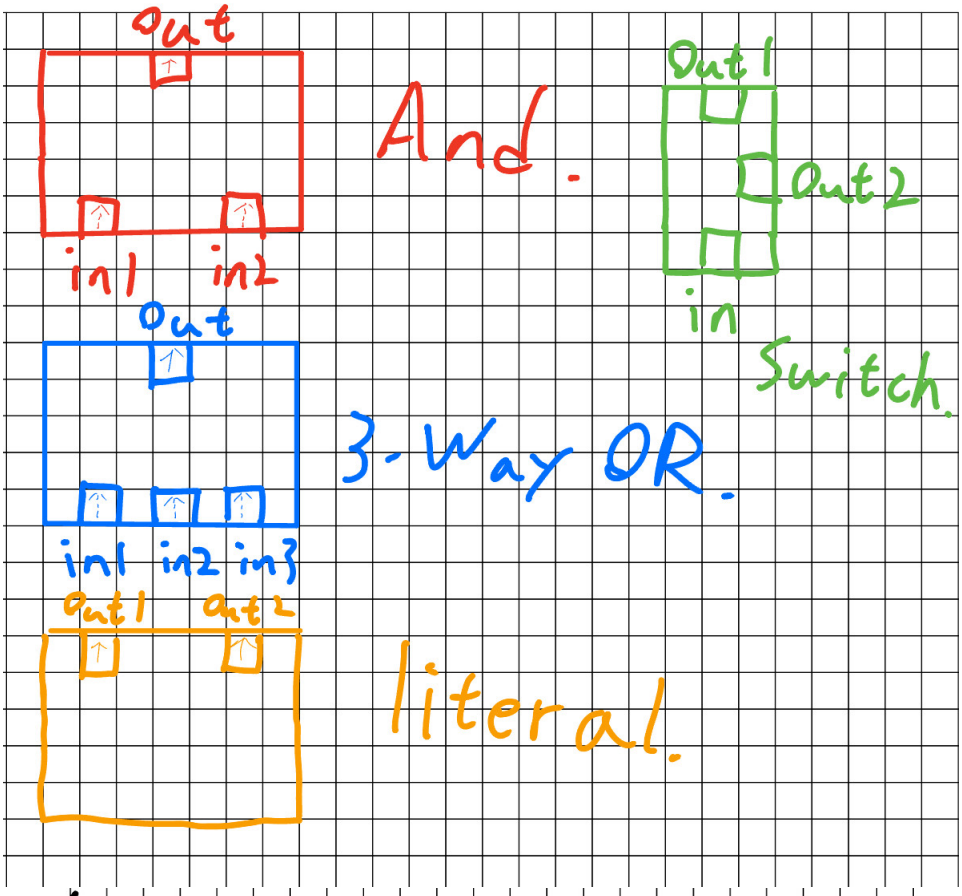
Construct a game corresponding to the formula `(A || B) && !A && !B`, the shootability of goal should map to the satisfiability of the formula.

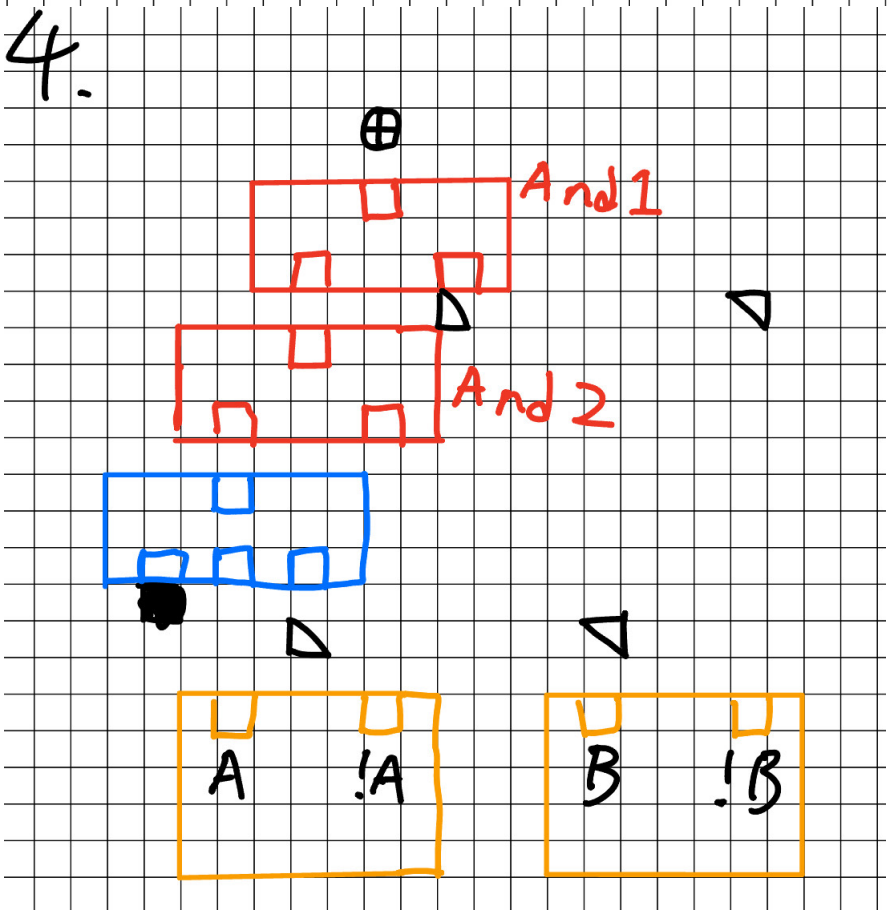
Do not evaluate truth value of the formula. Answers just draw a random laser tank puzzle, claiming it has same solvability to the truth value of formula would not be accepted. The answer puzzle should contain structure corresponding to logic literals and connectives of the formula.

P.s. N-way OR (N: positive integer) has basically the same structure, so use 2-way OR without its construction is acceptable in this subproblem.

Solution

For convenient, all the below pictures are rotated $\pi/2$ counterclockwise.


out
in1
in2
And.
out1
out2
in
Switch.
out
in1
in2
in3
3-Way OR.
out1
out2
literal.


4.
And1
And2
A
!A
B
!B

Answer explanation

By construction,

The target can become shootable iff

Both input tiles of And1 can become shootable iff

Output tile of And2 and !B can become shootable iff

Bothe input tiles of And2 and !B is shootable iff

Output tile of OR gadget can become shootable and !A is shootable and !B is shootable iff

(A or B is shootable) and !A is shootable and !B is shootable.

If the formula is satisfable(say the assignment is A=a, B=b). We can operate on the literal gadgets such that A is shootable iff a is True, !A is shootable iff a is False and B,!B similarily. Then the formula is true under the assigment which means the target is shootable after the operations by arguments above.

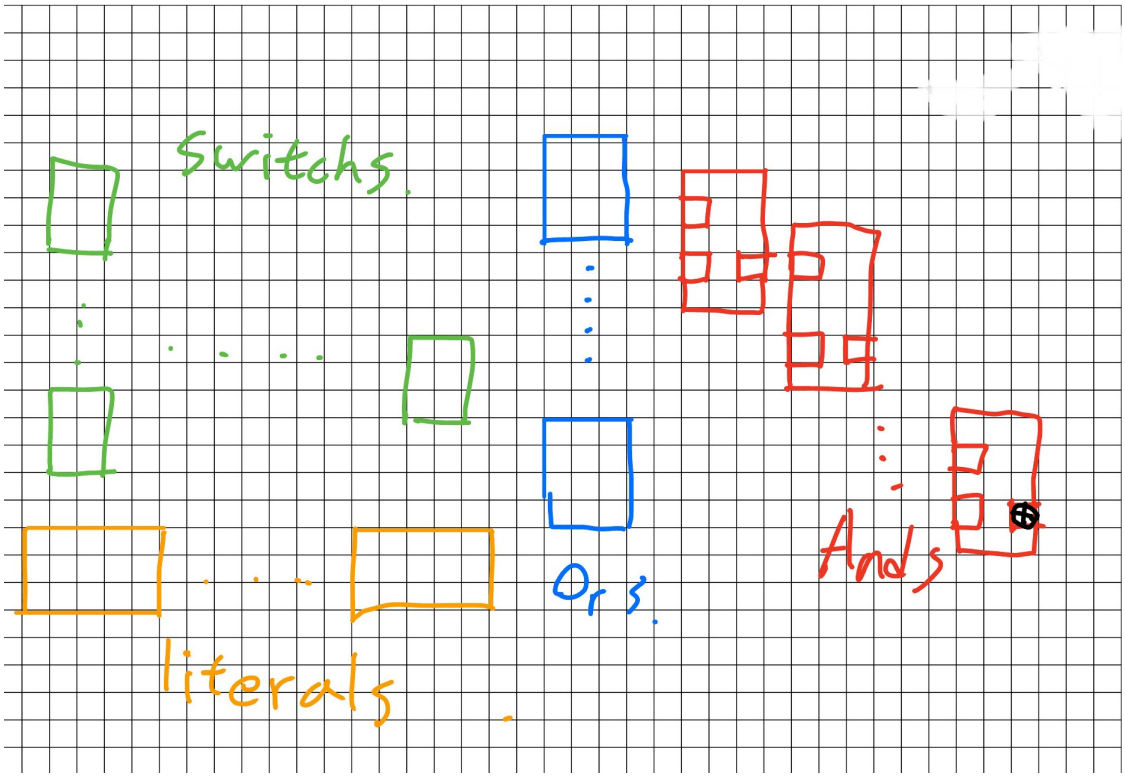
Since the literal gadget is constructed such that (A and !A), (B and !B) can't be shootable at the same time, suppose there's solution to the puzzle. Assign A to shootability of A and B to shootability of B is an satifying assignments to the formula with argument above.

Thus the puzzle is solvable iff the formula is satisfiable.

(5) 3-SAT can be Laser tank puzzle (8 points)

Design an algorithm that runs in polynomial time and reduces an instance of 3-CNF formula's satifiability problem to a laser tank puzzle.

Solution



We assume the input and output tile 1 of the switch gadget are on the same vertical line with input

tile on the bottom, and output tile 2 is on the right hand side, which can be easily ensured by adding some mirrors.

For each variable appears in the expression, we place an literal gadget for it on the buttom section.

For each 3-way OR clause, we place an 3-way OR gadget on right section. And for each literal appear in the clause, we place a switch gadget with input on the vertical line above the corresponding literal output and output to the OR gadget as input.

After above process, use AND gadgets to concatenate all outputs of 3-way OR gadgets. Place target at the output of the final AND gadget.

Size of all gadgets are constant, and the number of gadgets we places is linear in size of the 3-CNF expression. Thus the reduction process runs in polynomial time.

(6) Constructive reduction proof (8 points)

Design two algorithms. The first one converts "a boolean assignment satisfying the 3-CNF formula" to "a solution to the puzzle you constructed in (5)". The second one restores "a satisfying boolean assignment to the original 3-CNF-SAT problem" from "a solution to the laser tank puzzle". With these algorithms, you can conclude that the 3-CNF-SAT problem is reduced to laser tank puzzle, which implies the laser tank puzzle is NP-hard.

Solution

The first algorithm: Operates on the literal gadgets such that for each literal, it's shootable iff it's assigned True in the satisfying boolean assignment. Then from top to down, for each OR gadgets, we trigger the input switch gadgets. By assumption that the boolean assignment satisfies the expression, at least one input gadget is shootable. So we can make the outputs of all OR gadgets shootable. By the definition of AND gadget, we can process in order and make all outputs of AND gadgets shootable. Finally, the target is shootable and we solve the puzzle.

The second algorithm: In the final state, the target is shootable. By construction this implies all of the output of OR gadgets are shootable. For those literal assigned True, we assign the corresponding truth value to the variable. This guarantees the expression can be satisfied. For those didn't be assigned truth, we can assign arbitrary value. This give us an satisfying assignment.

With above two algorithm, we proved that the reduced laser tank puzzle has solution iff the original 3-SAT instance is satisfiable. Thus the 3-SAT problem can be reduced to laser tank puzzle in polynomial time and laser tank puzzle is NP-hard.

reference

arXiv:1908.05966v1

Problem 2 - Pledge of Z (Hand-Written) (20 points)

Majin Buu is going crazy and is ready to bombard our Earth. As one of the Z-Fighters, you, Goku, will not surrender to him. Nevertheless, Buu is so strong that you can't beat him even in Super Saiyan 3 mode. The only way to save the world is to collect as much energy as possible. You soon begin charging up your special move "Spirit Bomb", which requires the inhabitants of Earth to raise their hands to share their energy with you.

Piccolo, your smartest comrade, tells you the relations between the N people on the Earth. That is, Piccolo tells you a set of directed edges $U = \{(s_1, d_1), (s_2, d_2), \ldots, (s_M, d_M)\}$, each element (s_i, d_i) denotes that person indexed s_i can reach d_i. Note that $(s, d) \in U$ does not necessarily imply $(d, s) \in U$.

Since Buu is about to destroy the world, you can only use "telepathy" to persuade K people so that the number of people reachable from these K people is maximized. If you chose the K people optimally, let the number of people reachable from these K people be $\aleph_{\text{opt}}$. Your goal is to select K individuals using a greedy algorithm such that the number of people reachable from these K people is no less than $\left(1 - \left(\frac{K-1}{K}\right)^K\right) \cdot \aleph_{\text{opt}}$.

You find that it's harder to solve this problem than to beat Buu. You don't have time to argue with your comrades or complain anonymously on the Internet due to the emergency. So, you decide to focus on some simple subproblems first.

When solving (a), (b), (c) and (d), you should not consider the original problem.

(a) (3 points)

Given finite sets $A_1, \ldots, A_M$ such that $\left|\bigcup_{i=1}^{M} A_i\right| = N$ and a positive integer $K \leq M$. Design a greedy algorithm that runs in $O(NMK^2)$ time and outputs K indices $a_1, a_2, \ldots, a_K$ such that

$$\left|\bigcup_{i=1}^{K} A_{a_i}\right| \geq \max_{1 \leq b_1 < b_2 < \cdots < b_K \leq M} \left|\bigcup_{i=1}^{K} A_{b_i}\right| \cdot \left(1 - \left(\frac{K-1}{K}\right)^K\right).$$

For simplicity, you may use a function $f : (\text{a set of } K \text{ indices}) \to \mathbf{N} \cup \{0\}$ to denote the size of the union of K sets. More formally, $f(\{t_1, t_2, \ldots, t_K\}) = \left|\bigcup_{i=1}^{K} A_{t_i}\right|$. This function runs in $O(NK)$ time. You don't need to give a proof for correctness in this subproblem – give it in subproblem (d).

Solution

Let output={} initially. For i^{th} round, put a_i in output, where

$$a_i = \arg \max_{a \in \{1 \sim M\}} f(output \cup \{a\})$$

K rounds in total.

(b) (3 points)

Given an instance with $M = 5, K = 3, N = \left|\bigcup_{i=1}^{M} A_i\right|$, where

$$A_i = \begin{cases} X_{i-1} & ,\forall i = 1,2,3 \\ Y_{i-4} & ,\forall i = 4,5 \end{cases}, \text{ where } \begin{cases} X_i = \bigcup_{j=0}^{8} \{(i,j)\} \\ Y_i = \bigcup_{l=0}^{2} \bigcup_{r=0}^{\rho(i)} \{(l, \sigma(i)+r)\}, \text{ where } \begin{cases} \sigma(i) = 3^2\left(1-\left(\frac{2}{3}\right)^i\right) \\ \rho(i) = \left(\frac{2}{3}\right)^i \cdot 3^1 - 1 \end{cases} \end{cases}$$

Find a possible greedy selection a_1, a_2, a_3 by applying algorithm constructed from (a) such that $f(\{a_1, a_2, a_3\}) = \max_{1 \le b_1 < b_2 < b_3 \le 5} f(\{b_1, b_2, b_3\}) \cdot \left(1 - \left(\frac{2}{3}\right)^3\right)$.

Solution

$a_1 = 4 \to a_2 = 5 \to a_3 = 1$

(c) (3 points)

Given a positive integer K, please construct sets $A_1, A_2, \ldots, A_M$ such that

$$\max_{1 \le b_1 < b_2 < \cdots < b_K \le M} \left|\bigcup_{i=1}^{K} A_{b_i}\right| = \left|\bigcup_{i=1}^{M} A_i\right| = N = K^K.$$

And, it's possible for your algorithm to output $a_1, a_2, \ldots, a_K$ where

$$\left|\bigcup_{i=1}^{K} A_{a_i}\right| = N \cdot \left(1 - \left(\frac{K-1}{K}\right)^K\right).$$

Also, prove your correctness of your construction.

Solution

$$A_i = \begin{cases} X_{i-1} & ,\forall i = 1,2,.....,K \\ Y_{i-K-1} & ,\forall i = K+1,....,2\cdot K-1 \end{cases},$$

$$\text{where } \begin{cases} X_i = \bigcup_{j=0}^{K^{K-1}-1} \{(i,j)\} \\ Y_i = \bigcup_{l=0}^{K-1} \bigcup_{r=0}^{\rho(i)} \{(l, \sigma(i)+r)\}, \text{ where } \begin{cases} \sigma(i) = K^{K-1}\left(1-\left(\frac{K-1}{K}\right)^i\right) \\ \rho(i) = \left(\frac{K-1}{K}\right)^i \cdot K^{K-2} - 1 \end{cases} \end{cases}$$

And the greedy algorithm may return K+1 ~2K-1 followed by 1 that satisfy the condition.

Proof : For the first step, the greedy algorithm will choose from 1 to K+1 since A_1~A_{K+1} are all biggest sets. So, it's possible that we select K+1 as first element. For k^{th} step, s.t. k≤K-1, if we have already selected K+1 ~K+k-1, then

$$f(\{(K+1)\sim(K+k-1)\}\cup\{i\}) - f(\{(K+1)\sim(K+k-1)\})$$
$$= K^{K-1} - \sum_{j=0}^{k-2}(\frac{K-1}{K})^j K^{K-2}$$
$$= f(\{(K+1)\sim(K+k-1)\}\cup\{(K+k)\}) - f(\{(K+1)\sim(K+k-1)\}) \text{ if i} \in 1 \sim \text{K},$$

It's possible that k^{th} step select A_{K+k} since A_1 ~A_K and A_{K+k} all contain the most new elements that are not included in the past k-1 step.
For K'th step, if we already have selected K+1 ~K+K-1, then we can only choose from 1 ~K where

$$f(\{K+1\sim K+k-1\}\cup\{1\}) = K^K(1-(\frac{K-1}{K})^K)$$

while OPT selection is A_1~A_K s.t.

$$f(\{1\sim K\}) = K^K$$

Hence, the condition in the problem statement is satisfied.

(d) (3 points)

Prove that

$$\left|\bigcup_{i=1}^{K} A_{a_i}\right| \geq \max_{1\leq b_1<b_2<\cdots<b_K\leq M}\left|\bigcup_{i=1}^{K} A_{b_i}\right| \cdot \left(1-\left(\frac{K-1}{K}\right)^K\right)$$

holds for the K indices $a_1, a_2, \ldots, a_K$ returned from your algorithm in (a).

Solution

Let the first k indexes a_1~a_k selected by greedy algorithm be denoted as S_k and the OPT selection be $S^\star$. For the first step, there must exist $x \in S^\star$ s.t. $|A_x| \geq \frac{f(S^\star)}{K}$ → will select a_1 as first element s.t. $|A_{a_1}| \geq |A_x|$

For k^{th} step, $k \geq 2$, if $f(S^\star) > f(S_{k-1}) \geq (1-(\frac{K-1}{K})^{k-1}) \cdot f(S^\star)$

→ there must exist $x \in S^\star$ s.t.

$f(S_{k-1}\cup\{x\}) - f(S_{k-1}) \geq \frac{f(S^\star)-f(S_{k-1})}{K}$

→ $f(S_k) \geq (1-(\frac{K-1}{K})^{k-1}) \cdot f(S^\star) + \frac{f(S^\star)-(1-(\frac{K-1}{K})^{k-1})f(S^\star)}{K} = (1-(\frac{K-1}{K})^k) \cdot f(S^\star)$

. By induction, $f(S_K) \geq (1-(\frac{K-1}{K})^K) \cdot f(S^\star)\#$

(e) (3 points)

Solve the original problem in $O(NMK)$ time and prove your correctness. You may cite any piece of your proof in (d).

Solution

Construct a direct graph G=$< V, E >$ s.t. V = all people, E=relationship between all vertices, $|V| = N$, $|E| = M$. For k^{th} round, enumerate all N vertices and select one such that it can reach the most unreached people by doing dfs.
Correctness:
The outcome of the greedy algorithm is exactly the same as we let A_i be the set of people reachable from person i, K=K, and leverage the algorithm (a) which is proved correct in (d).
Time:
Dfs take O(M) while we do this N times during each of K rounds. Therefore, O(NMK) in total.

(f) (5 points)

Let's consider a variation of the original problem.

For each element e in U, e appears in U' with constant probability p. Let A_i be the set of people reachable from person i only with the relations in U'. Design an algorithm that runs in $O\left(NMK2^M\right)$ time and output K indices such that

$$E\left[\left|\bigcup_{i=1}^{K} A_{a_i}\right|\right] \geq E\left[\aleph_{\text{opt}}\right] \cdot \left(1 - \left(\frac{K-1}{K}\right)^K\right)$$

Also, prove the correctness. Note that you will not know what U' really is. In other words, you can only make your final solution through p and N, U given in the original problem. Also,

$$E\left[\aleph_{\text{opt}}\right] \tag{1}$$

is the optimal expected reachable people if you chose the K people based on N,U,p optimally. To see an example, please refer to NTU Cool Announcement

Solution

There are M edges in G(V,E) $\rightarrow 2^M$ possibilities in total. H=$\{h_1 \sim h_{2^M}\}$ denotes all the possible graphs, $h_i = (V, E_i)$, $P_i = (p^{|E_i|})((1-p)^{|E|-|E_i|})$=the possibility of h_i, $f_{inf}(\{t_1, t2, ..., t_k\}) = E[\left|\bigcup_{i=1}^{k} A_{t_i}\right|]$.
Let output=$\{\}$ initially.
For k^{th} round, put a_k in output, where

$$a_k = \arg\max_{a \in \{1 \sim N\}} f_{inf}(output \cup \{a\})$$

To implement this, we may first do dfs from a on each of $h_i \in H$, multiple them by P_i, and sum them up. Repeat the procedure to find a_1. After that, mark all reachable vertices from a_1 on each of $h_i \in H$.

For k^{th} round, suppose we already have all vertices on each of $h_i \in H$ reachable from some of previously selected vertices marked. Then we can calculate $f_{inf}(output \cup \{a\})$ by doing dfs from a on each of $h_i \in H$, multiple the number of encountered but unmarked vertices on h_i by P_i and sum them up.
At the end of k^{th} round, mark all encountered but unmarked vertices by doing dfs from a_k on each of $h_i \in H$

Time:

Dfs take O(M) while we do this N times on each of 2^M graphs during each of K rounds. Therefore, O(NMK2^M) in total.

Correctness:

Let the first k indexes a_1~a_k selected by greedy algorithm be denoted as S_k and the OPT selection be $S^\star = s^\star{}_1 \sim s^\star{}_K$. We know that

$\sum_{i=1}^{K} f_{inf}(s^\star{}_i) \geq f_{inf}(S^\star) \rightarrow f_{inf}(S_1) \geq (1 - (\frac{K-1}{K})^1) \cdot f_{inf}(S^\star)$.

Assume $f_{inf}(S_k) \geq (1 - (\frac{K-1}{K})^k) \cdot f_{inf}(S^\star)$. We know that $L_i = \sum_{j=1}^{K} (f_i(s^\star{}_j \cup S_k) - f_i(S_k)) \geq R_i = f_i(S^\star) - f_i(S_k)$, for i $\in$ 1 ~2^M, where f_i denote to use f() mentioned in (a) on $h_i \in H$

$\rightarrow P_i L_i \geq P_i R_i \rightarrow \sum_{i=1}^{K} (f_{inf}(s^\star{}_i \cup S_k) - f_{inf}(S_k)) = \sum_{i=1}^{2^M} (P_i L_i) \geq \sum_{i=1}^{2^M} (P_i R_i) = f_{inf}(S^\star) - f_{inf}(S_k)$

$\rightarrow$ we may select a_{k+1} s.t. $f_{inf}(\{a_{k+1}\} \cup S_k) - f_{inf}(S_k) \geq \frac{f_{inf}(S^\star) - f_{inf}(S_k)}{K}$

$\rightarrow f_{inf}(S_{k+1}) \geq f_{inf}(S_K) + \frac{f_{inf}(S^\star) - f_{inf}(S_k)}{K} \geq \frac{K-1}{K}(1 - (\frac{K-1}{K})^k) \cdot f_{inf}(S^\star) + \frac{1}{K} f_{inf}(S^\star)$

$= (1 - (\frac{K-1}{K})^{k+1}) \cdot f_{inf}(S^\star)$

$\rightarrow$ By induction, $f_{inf}(S_K) \geq f_{inf}(S^\star) \cdot (1 - (\frac{K-1}{K})^K) \#$

Reference: ALGORITHMS ILLUMINATED PART4, TIM ROUGHGARDEN