Sports Betting

Calvin Fung

June 22, 2025

1 Problem

The boarding process for various flights can occur in different ways: either by bus or through a telescopic jet bridge. Every day, exactly one flight is made from St. Petersburg to Minsk, and Vadim decided to demonstrate to the students that he always knows in advance how the boarding will take place.

Vadim made a bet with n students, and with the i -th student, he made a bet on day a_i . Vadim wins the bet if he correctly predicts the boarding process on both day $a_i + 1$ and day $a_i + 2$.

Although Vadim does not know in advance how the boarding will occur, he really wants to win the bet at least with one student and convince him of his predictive abilities. Check if there exists a strategy for Vadim that allows him to guarantee success.

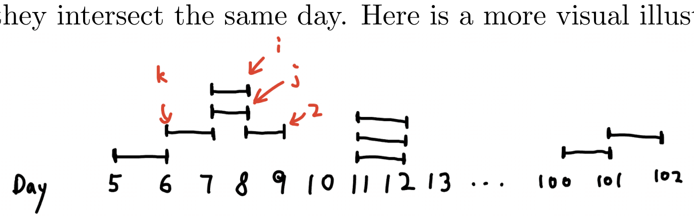
2 Solution

First we'll analyse the problem, then we'll present an algorithm that runs in randomized $O(n)$ time.

2.1 Problem analysis

Given input $a_1, ..., a_n$, WLOG assume they're sorted, we can create an interval $I_j = [a_j+1, a_j+2]$ for each student j, let $\mathcal{I} = \{I_j : j = 1, ..., n\}$ be the collection of all such intervals, notice there's one-to-one correspondence between the sorted input $a_1, ..., a_n$ and $\mathcal{I}$. Therefore, we say that : for a given input $a_1, ..., a_n$ there exists a *winning strategy* (ie. a strategy for Vadim that guarantee him to win bet with at least one student) iff we call the corresponding $\mathcal{I}$ *good*.

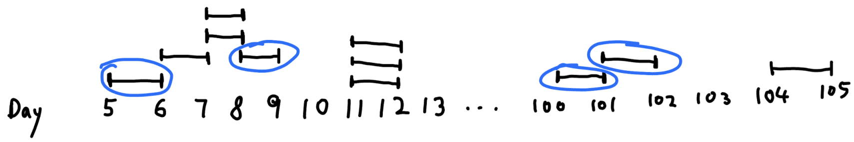
We say an interval I and a day d *intersect* iff $d \in I$. We also say an interval I and an interval J *intersect* iff they intersect the same day. Here is a more visual illustration :



- $a_k = 5$, $a_j = a_i = 6$, $a_2 = 7$
- interval i,j,k intersect day 7
- interval i,k intersect
- interval i,2 intersect
- interval i,j intersect

e.g.

There're several important structures in $\mathcal{I}$, the first one are called *wing* intervals :



denotes wing interval

- Notice removal of interval [5,6] creates a new wing interval [6,7]
- Notice [104, 105] is NOT a wing interval (see def below)

More formally, a *wing* interval is an interval $I = [d, d+1]$ such that exactly one of day d or $d+1$ intersect only 1 interval.

Lemma 1. For any *wing* interval $I = [d, d+1] \in \mathcal{I}$, $\mathcal{I} \setminus I$ is good iff $\mathcal{I}$ is good.

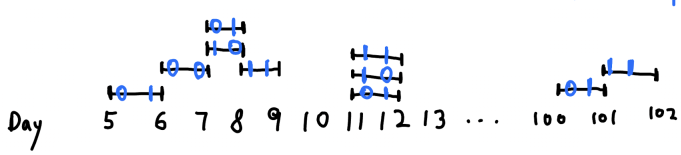
proof. First it's useful to consider another perspective

2-SAT perspective : Given a betting strategy Ω ie. for each student j Vadim's bet on day a_j+1, a_j+2 , we can create a 2-SAT formula in CNF. There's a boolean variable x_d for each day d that intersects an interval. For each interval $I_j = [d, d+1]$, if Vadim bet on bus on day $q \in \{d, d+1\}$, then let $y_q = \bar{x_q}$, else $y_q = x_q$. Then create a clause $C_j = y_d \wedge y_{d+1}$. The CNF formula is just $f_\Omega = \bigvee_{j=1}^{n} C_j$.

Notice Ω is a winning strategy iff f_Ω is always satisfied (ie. satisfied for any setting of variables).

e.g. Given a strategy

bet { 0 bus, 1 telescopic jet bridge



2-SAT CNF formula:

$$(\bar{x}_5 \wedge x_6) \vee (\bar{x}_6 \wedge \bar{x}_7) \vee (\bar{x}_7 \wedge x_8) \vee (x_7 \wedge \bar{x}_8) \vee (x_8 \wedge x_9)$$
$$\vee (x_{11} \wedge x_{12}) \vee (x_{11} \wedge \bar{x}_{12}) \vee (\bar{x}_{11} \wedge x_{12})$$
$$\vee (\bar{x}_{100} \wedge x_{101}) \vee (x_{101} \wedge x_{102})$$

($\Rightarrow$) obvious

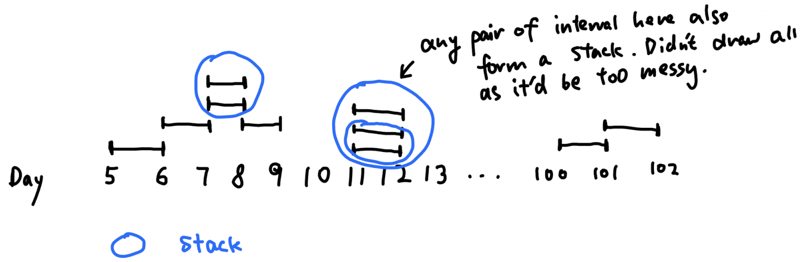
($\Leftarrow$) WLOG suppose d intersects only 1 interval. As $\mathcal{I}$ is good there's a winning strategy Ω for $\mathcal{I}$ from which we can construct a 2-SAT CNF formula f_Ω that is always satisfied. Consider the clause $y_d \wedge y_{d+1}$ corresponding to I in f_Ω, notice removing it gives us another 2-SAT CNF formula f_Θ which corresponds to a strategy Θ for $\mathcal{I} \setminus I$.

It remains to show that Θ is a winning strategy by showing f_Θ is always satisfied. Notice if the variables of f_Θ is V then the variables of f_Ω will be $V \cup \{x_d\}$ where $x_d \notin V$. Suppose there's a setting $\phi : V \to \{0,1\}$ of variables such that f_Θ is not satisfied. Then consider the setting $\psi : V \cup \{x_d\} \to \{0,1\}$ where $\psi(x) = \phi(x)$ for $x \in V$ and

- $\psi(x_d) = 0$ if $y_d = x_d$
- $\psi(x_d) = 1$ if $y_d = \bar{x_d}$

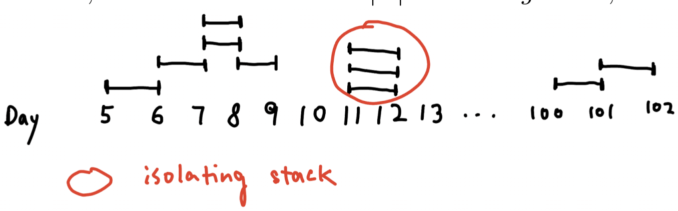
So both the clause $y_d \wedge y_{d+1}$ and f_Θ will not be satisfied under ψ, it follows $f_\Omega = (y_d \wedge y_{d+1}) \vee f_\Theta$ will not be satisfied under ψ, a contradiction. ∎

The other type of interesting structure is what we call a *stack*,



More formally, an $|S| - \textit{stack}$ S is a set of intervals such that every pair intersect on 2 days.

Further, we call a stack S an $|S|-\textit{isolating stack}$, if no interval in S intersects an interval not in S.



Lemma 2. For an *isolating stack* $S \subseteq \mathcal{I}$ where $|S| \leq 3$, $\mathcal{I} \setminus S$ is good iff $\mathcal{I}$ is good.

proof.

($\Rightarrow$) obvious

($\Leftarrow$) We're going to use the **2-SAT perspective** introduced in the proof of lemma 1. As $\mathcal{I}$ is good there's a winning strategy Ω for $\mathcal{I}$ from which we can construct a 2-SAT CNF formula f_Ω that is always satisfied. There'll be < 4 clauses in f_Ω corresponding to the stack $|S|$, notice those clauses will all be of the form (a literal of x_d) $\wedge$ (a literal of x_{d+1}) for some day d. As there're < 4 such clauses, one of the clause type will not be there, suppose it's the clause type $\bar{x_d} \wedge x_{d+1}$ (the reasoning below will be similar for other clause types).

Now remove the $|S|$ clauses corresponding to S in f_Ω, this would yield a 2-SAT CNF formula f_Θ corresponding to a strategy Θ for $\mathcal{I} \setminus S$. It remains to show that Θ is a winning strategy by showing f_Θ is always satisfied. Notice if the variables of f_Θ is V then the variables of f_Ω will be $V \cup \{x_d, x_{d+1}\}$ where $x_d, x_{d+1} \notin V$. Suppose there's a setting $\phi : V \to \{0, 1\}$ of variables such that f_Θ is not satisfied. Then consider the setting $\psi : V \cup \{x_d, x_{d+1}\} \to \{0, 1\}$ where $\psi(x) = \phi(x)$ for $x \in V$ and

- $\psi(x_d) = 0$

- $\psi(x_{d+1}) = 1$

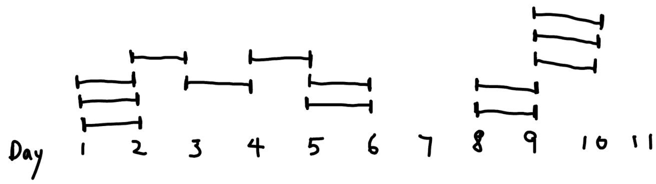
So the $|S|$ clauses corresponding to S in f_Ω and f_Θ will not be satisfied under ψ, it follows f_Ω will not be satisfied under ψ, a contradiction. ∎

Lemma 3. For $\mathcal{I}$ without any *wing interval* and *isolating stack*, it's good iff it's not empty.

proof.

($\Rightarrow$) obvious

($\Leftarrow$) First we're going to create a strategy Ω for $\mathcal{I}$. Consider a timeline and put intervals of $\mathcal{I}$ on the corresponding days ie.



WLOG suppose the earliest day that intersects an interval is day 1. As there's no wing, there must be a 2-stack ie. two intervals $[1,2],[1,2]$, call them group 1, we bet $(1,0)$ for one of them and $(0,0)$ for the other (0 denotes bus and 1 denotes telescopic jet bridge. Also we use parenthesis to denote bet ie. (bet on day d, bet on day d+1) as we already used bracket to denote interval ie. [day d, day d+1]).

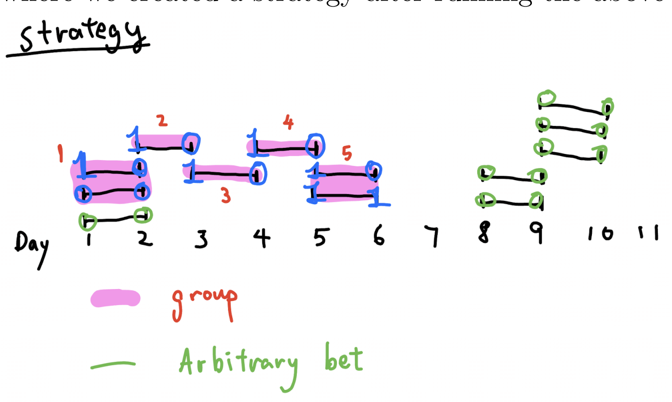
Now

```
If(there's a 2-stack S = {[2,3],[2,3]}){
        call S group 2 and bet (1,0) for one interval in S and (1,1) for the other.
}
else {
        there must be exactly one interval [2,3], otherwise there'll be an isolating stack.
        Let this interval be in group 2 and bet (1,0) for it.

        i = 2
        while(there doesn't exist a 2-stack S = {[i+1,i+2],[i+1,i+2]}){
                        There must be exactly one interval [i+1,i+2],
                        otherwise group i will be a wing.
                        Let this interval be in group i+1 and bet (1,0) for it.
                        ++i
        }
        Clearly the while loop must terminate because we have finite number of intervals.
        When it terminates, it means there's a 2-stack S = {[i+1,i+2],[i+1,i+2]},
        call S group i+1 and bet (1,0) for one interval in S and (1,1) for the other.

}
```

Finally, for the rest of the intervals that's not in any group, we can bet on them arbitrarily. Below is an example where we created a strategy after running the above procedure on a set of intervals:



Suppose we have w groups in total, by construction, group 1 and w will be a 2−stack ie. $\{[1,2],[1,2]\}, \{[w,w+1],[w,w+1]\}$ respectively. While group $g = 2, ..., w-1$ will contain just an interval $[g, g+1]$.

Now we have our strategy Ω for $\mathcal{I}$, which corresponds to a 2-SAT CNF formula f_Ω (recall the **2-SAT perspective**), we show that it'll always be satisfied.

Let C_g denote the clauses corresponding to group g, more specifically :

- $C_1 = (x_1 \wedge \bar{x_2}) \vee (\bar{x_1} \wedge \bar{x_2})$

- $C_w = (x_w \wedge \bar{x}_{w+1}) \vee (x_w \wedge x_{w+1})$

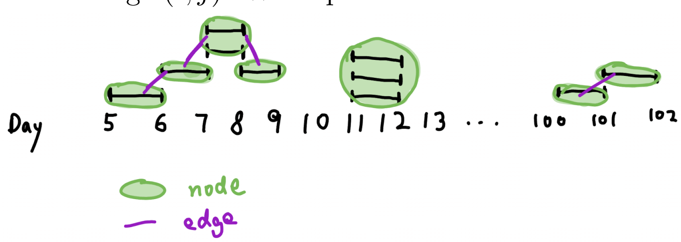
- $C_g = (x_g \wedge \bar{x}_{g+1}) \qquad , g = 2, ..., w-1$

Consider any setting ϕ of the variables. If $\phi(x_2) = 0$ then C_1 is satisfied. Otherwise consider the smallest $i \in \{3, ..., w+1\}$ such that $\phi(x_i) = 0$, as $\phi(x_{i-1}) = 1$ we know C_{i-1} will be satisfied. If no such i exist, then we must have $\phi(x_w) = 1, \phi(x_{w+1}) = 1$ so C_w will be satisfied. Therefore, f_Ω is always satisfied $\implies$ Ω is a winning strategy $\implies$ $\mathcal{I}$ is good . ■

2.2 Algorithm

From our analysis above, it follows any algorithm that first check if there's a 4-stack, if there's one it returns true. Otherwise it removes all wing intervals and isolating stacks, then return true iff the there's ≥ 1 interval left will be correct. It remains to design a fast way to do it.

Given $\mathcal{I}$ we can create a graph $G = (V, E)$ where there's a node for every interval except for intervals that contain the same two days, they will share the same node. Each node will store the information about how many intervals correspond to this node. If two intervals I, J intersect and correspond to two different nodes i, j, add the edge (i, j). We require this to be an undirected and simple graph. e.g.



Observe that every component will be a simple path.

Below is an algorithm that runs in randomized $O(n)$ time.

```
f(A,n){
        //>>>>>>>>>>>>>>>>>>>>STAGE 1: Create G<<<<<<<<<<<<<<<<<<<

        G = an array of n empty lists
        C = an array of size n //store how many intervals corr. to each node
        D = an array of size n //store degree of each node
        num_nodes = 0
        /*
        Each entry (key,value) corresponds to a node of G.
        All intervals corresponding to this node begins at day key.
        The node id is value.
        e.g. intervals [2,3],[2,3],[2,3] all correspond to same entry with key = 2
        */
        H = empty hash table

        for(i = 1;i <= n;++i){
                if(no element in H with key A[i] + 1){
                        //create new node
                        num_nodes++;
                        u = num_nodes; //id of the new node
                        H[A[i] + 1] = u;
                        C[u] = 1;
                        D[u] = 0;

                        //add edges
                        if(there is an element in H with key A[i]){
                                v = H[A[i]];
                                G[v].push(u);
                                G[u].push(v);
                                D[u]++;
                                D[v]++;
```

```
                    }
                    if(there is an element in H with key A[i] + 2){
                            v = H[A[i] + 2];
                            G[v].push(u);
                            G[u].push(v);
                            D[u]++;
                            D[v]++;
                    }
            }else{
                            //the node already exist
                            u = H[A[i] + 1];
                            C[u]++;
                             //clearly there's a winning strategy if there's a 4-stack
                            if(C[u] == 4)    return true;
            }
    }

    //>>>>>>>>>>>>>>>>>>>>STAGE 2: Remove all wing intervals<<<<<<<<<<<<<<<<<<

    Q = empty queue //stores the node id corresponding to wing intervals
    Run DFS on G, whenever we visit a node u with D[u] == 1 and C[u] == 1, push u into Q

    while(!Q.empty()){
                    u = Q.pop();
                    v = G[u].front(); //the only neighbor of u

                    //v becomes wing interval after removal of u ?
                    if(D[v] == 2 && C[v] == 1)       Q.push(v);

                    //every node has degree <= 2 so these are all constant time operations
                    G[u].clear();
                    remove u from G[v];
                    D[v]--;
                    D[u] = -1;
                    num_nodes--;
    }

    //you can easily prove by contradiction that no wing interval remains after this stage

    //>>>>>>>>>>>>>>>>>>>>STAGE 3: Remove all isolating stack with size < 4<<<<<<<<<<<<<<<<<<

    for(i = 1;i <= n;i++){
            if(D[i] == 0){
                    num_nodes--;
            }
    }

    //>>>>>>>>>>>>>>>>>>>>STAGE 4: check if resulting graph is empty<<<<<<<<<<<<<<<<<<

    return num_nodes > 0 ? true : false;
}
```