Maths Circle India

TIFR-STCS Maths Circle Team

Session 9: September 8, 2023

1 Sports day: Continued

We meet again our old friends, Arun, Biron and Kiron (ABK). Last time, we found them considering the problem of scheduling some selection of n games, where the ith game (where $1 \leqslant i \leqslant n$) has exactly c_i conflicts. They were finally able to find a way to schedule at least

$$\sum_{i=1}^{n} \frac{1}{c_i + 1} \tag{1}$$

non-conflicting games in such a situation. To do this, they found helpful the notion of *ordering* games and then looking for *peaking games* in those orderings. To recall, an *ordering* is just a list of games arranged in order. Given an ordering O, we say that the game g is *peaking* in O if it has a higher rank in O than all the games it conflicts with. For an ordering O and game g, we defined $\text{peak}(O, g) = 1$ if g is peaking in O and $\text{peak}(O, g) = 0$ otherwise. We denote the total number of games that are peaking in O as $\text{peak}(O)$. Thus, for every ordering O,

$$\text{peak}(O) = \sum_{i=1}^{n} \text{peak}(O, g_i).$$

We also defined AvgPeak to be the average number of peaking games in a ordering. Thus,

$$\text{AvgPeak} = \frac{1}{n!} \sum_{O} \text{peak}(O),$$

where the sum is over all possible orderings O. Similarly, we defined $\text{AvgPeak}(g)$ to be the *fraction* of orderings in which the game g is peaking. We then proved the following.

- For any game g_i,
$$\text{AvgPeak}(g_i) = \frac{1}{n!} \sum_{O} \text{peak}(O, g_i) = \frac{1}{c_i + 1}, \tag{2}$$
where the sum is over all orderings O.
- We then saw that
$$\text{AvgPeak} = \sum_{i=1}^{n} \text{AvgPeak}(g_i)$$
and that there must exist an ordering Q for which
$$\text{peak}(Q) \geqslant \text{AvgPeak}.$$

- Putting these together, we obtained a proof of there being a non-conflicting set of games of size at least that given in eq. (1). We even found that the ordering which arranges games in increasing order of number of conflicts always has as many peaks as in eq. (1).

[*End of recap.*]

Now, ABK are wondering when is it that the "best" schedule is given by the set of peaking games in some ordering, and also when is it *not* possible to do better than eq. (1).

1. Can you construct a list $\mathcal{B}$ of games for which the "best" non-conflicting schedule is *not* the set of peaking games in *any* ordering of the games in $\mathcal{B}$?
2. Suppose $\mathcal{L}$ is a list of games for which the number of games in the "best" schedule is *not* larger than what is promised by eq. (1). The questions in this part are all in the context of the list $\mathcal{L}$.
   (a) Show that in this case, *every* ordering must have the same number of peaking games.
   (b) Show that in this case, if game g conflicts with games h_1 and h_2, then the games h_1 and h_2 must also conflict with each other.
   (c) Show therefore that, in this case, if we define ConflictSet(g) to be the set of games conflicting with game g, then for every game h in ConflictSet(g), and every game k different from g and h, k is in ConflictSet(g) if and only if k is in ConflictSet(h). If game k_1 is in the set ConflictSet$(g) \cup \{g\}$ and the game k_2 is *not* in the set ConflictSet$(g) \cup \{g\}$, can the games k_1 and k_2 conflict?
   (d) What is the total number of (unordered) conflicting pairs of games among the games in ConflictSet$(g) \cup \{g\}$?
3. Suppose now that $\mathcal{L}$ is a list of games about which we only know the following facts. Below $q \geqslant 3$ is a positive integer.
   - There are qn games in $\mathcal{L}$, where n is a positive integer.
   - The number of (unordered) conflicting pairs of games in $\mathcal{L}$ is $nq(q-1)/2$.

   What is the minimum possible value of the quantity in eq. (1) for such a list? Can you construct an example of such an $\mathcal{L}$ where this minimum value is achieved?
4. Suppose now that $\mathcal{H}$ is *some* list with qn games, $nq(q-1)/2$ unordered conflicting pairs in which the number of games in the "best" schedule is equal to the minimum value in the previous part. What must $\mathcal{H}$ look like?

2 Detecting problems

ABK are now done with the sports day and have a new problem to tackle. They found in a library some very old floppy disks on which are saved some of their favourite story books. The books are stored in a *binary* format: each character (letter, digit, punctuation mark, space, ...) is converted into a unique sequence of 0s and 1s, and these are then stored (in the same sequence as the sequence of chracters in the book) on the floppy disks (each such 0 and 1 is referred to as a *bit*). However, Arun and Barun are worried that over time, the disks might get subtly corrupted, without their knowledge! They want to come up with methods to quickly detect at least some kinds of corruptions.

1. ABK believe that the typical corruption would take the form of either exactly one of the 0s flipping to a 1, or exactly one of the 1s flipping to a 0. Can you devise a method for creating a "summary" of the book, hopefully much smaller than the size of the book itself, and hopefully not too hard to calculate given the book, such that the "summary" will differ for two books that differ exactly by one of the "bits" being flipped?

2. ABK now worry about more subtle corruptions: perhaps two adjacent bits can get swapped! (That is, a pattern of 01 might get transposed to 10). Can you now think of a method for creating a "summary", just like in the previous part, such that the summary would be different for any two books that differ exactly by one such swap of adjacent bits?