University of Wisconsin
PLATTEVILLE
Department of Mathematics

Problem of the Week #9

(Spring 2023)

UWPACS, the UW-Platteville Armageddon Chess Squad, needs to get ready for the new season by ranking its eight players. They don't have much time before they have to submit their rosters, with their players in order from strongest to weakest, and unfortunately they only have one chess clock, so they can't play more than one game at a time. The good news is that Armageddon chess can't end in a draw*, and the games don't take nearly as long as ordinary competitive chess: they're always over in ten minutes or less.
Because they're in a hurry, the coaches have decided to assume that game results really tell you how good the players are — there are never any upsets — and that therefore, if player A beats player B and B beats C, we know for sure that A is better than both B and C (and B is better than C).

1. If the team has a three-hour meeting in which to play games, and the coaches pick the players for each game as they go along, can they get enough information to determine a ranking of all eight players?

2. What if the team only has two and a half hours to get the ranking done?

[You should submit your solution even if you only answer one of these two questions!]

Solution:

1. If we have two sets A and B of n players each, and we already know how the players in each of those sets should be ranked against each other, then we can rank the whole set $A \cup B$ of $2n$ players in the following way. First, have the best player in A play the best player in B. The winner is the best player in $A \cup B$: remove them from the set, and repeat the process. At worst, this can take $2n - 1$ games, removing one of the $2n$ players with each game. (We might finish in as few as n games if the players in A are all better than the players in B or *vice versa*.)

   UWPACS has 8 players. With a single game, we can rank one pair of players. It can take up to 3 games to turn two ranked pairs into a ranked set of 4, and up to 7 games to turn two ranked foursomes into a ranked set of 8. So we can rank the whole squad using $4 \cdot 1 + 2 \cdot 3 + 1 \cdot 7 = 17$ games, which they can finish in two hours and fifty minutes.

2. In 150 minutes, UWPACS can only count on holding 15 ten-minute games, with $2^{15} = 32768$ possible sequences of results. That's not enough different outcomes to tell apart the $8! = 40320$ possible rankings.

Remark. I ~~don't~~ know† whether or not they could rank their 8-player squad within 16 games.

Source: Dennis Shasha, *The Puzzling Adventures of Dr. Ecco.* New York: W.H. Freeman and Company (1988), 23–24, 141–143.

*Stalemate is treated as a win for Black.

†See next page!

Update (4/6/2023): I have learned that the 8-player squad can always be ranked within 16 games, using a sorting algorithm called "merge-insertion sort."[‡] Here's how it goes with eight players:

Step 1 (7 games): Run a standard single-elimination tournament. The winner x_1 is best.

Step 2 (1 game): The player who lost the championship game plays the semifinalist that they haven't played already. The winner x_2 has beaten two semifinalists and lost to x_1.

Step 3 (0–1 game): Those two semifinalists play, if they haven't already: the winner x_3 is better than the loser x_4.

At this point, let y_i be the opponent of x_i in the first round of the tournament. If we write $>$ for "is better than," we know $x_1 > x_2 > x_3 > x_4 > y_4$, as well as $x_1 > y_1$, $x_2 > y_2$, and $x_3 > y_3$.

Step 4 (2 games): Rank y_2 among $\{x_3, x_4, y_4\}$ by binary search. First y_2 should play x_4. If $y_2 > x_4$, then y_2 plays x_3, but if $x_4 > y_2$ then y_2 plays y_4.

Step 5 (2 games): Rank y_3 among $\{x_4, y_2, y_4\}$ by binary search. First y_3 should play the middle-ranked member of $\{x_4, y_2, y_4\}$. If they win, they play the top-ranked member next, and if they lose they face the bottom-ranked member.

Step 6 (3 games): Rank y_1 among $\{x_2, x_3, x_4, y_2, y_3, y_4\}$ by binary search. That is, match y_1 against a middle-ranked member of that set to eliminate half of the possible rankings for y_1. Repeat twice to find the right ranking for y_1.

[‡] `https://en.wikipedia.org/wiki/Merge-insertion_sort`