UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DIEGO HOMMERDING AMORIM

Assignment 3
"Winning a Tournament" to Flow Problem

Porto Alegre
2025

1.1 Task

The goal of this implementation is to solve the problem of winning a tournament via reduction to a flow problem. Specifically, conduct experiments to analyze the following:

- The relationship between time and the number of teams.
- The probability of winning the tournament as a function of the bias of the first team winning and the fraction of games that have already occurred.

1.2 Solution

The "Winning a Tournament"reduction was implemented using a customized graph data structure from the previous assignment [4]. This implementation extends that structure by introducing a child class that, instead of reading a DIMACS graph from standard input, parses a "Winning a Tournament"instance produced by the custom generator [6]. The reduction logic is applied incrementally as the instance is read, directly transforming the tournament configuration into a max-flow problem.

Two key validation checks are incorporated into the process:

- **After the reduction and before solving the max-flow problem**: During the reduction, a flag is set if any team already has at least as many wins as team 1 can theoretically achieve. This flag is only evaluated after the full reduction is complete. Delaying the check in this way ensures that the entire graph is always read and processed uniformly, avoiding early exits and thus preserving consistent benchmarking times across all instances.
- **After computing the max-flow problem**: A check verifies that all edges outgoing from the source node are saturated, confirming that every undecided match was allocated in a way that allows team 1 to win.

After the reduction, the algorithm uses the fattest-path strategy in the Ford–Fulkerson method to compute the maximum flow. This variant was selected based on the conclusion of the previous assignment [4]. Further details about the reduction process can be found in the Advanced Algorithms class notes, page102 [5].

1.3 Test environment

The results were obtained using a *Lenovo Ideapad S145-15IWL* notebook with an **Intel(R) Core(TM) i7-8565U** processor running at **1.80 GHz** and **20 GB of RAM**. The system was running **Ubuntu 22.04 LTS** with Linux kernel 6.8.0. The code was compiled using **g++ 13.1.0** with optimization flags `-O3`.

1.4 Results

This section presents the experimental results, divided into two subsections: *Win Rate Heatmap* and *Time Measurement*.

1.4.1 Win Rate Heatmap

Figure 1.1 shows the heatmap of “yes” outcomes for different values of α and β, where α represents the fraction of games already played in a tournament instance, and β controls the bias toward team 1 winning those played games. For each (α, β) pair, 100

tournament instances were generated and evaluated. The heatmap reflects the percentage of instances in which team 1 could still win the tournament after all computations. In all cases, the number of teams was fixed at 20, and the number of games per pairing was fixed at 2.

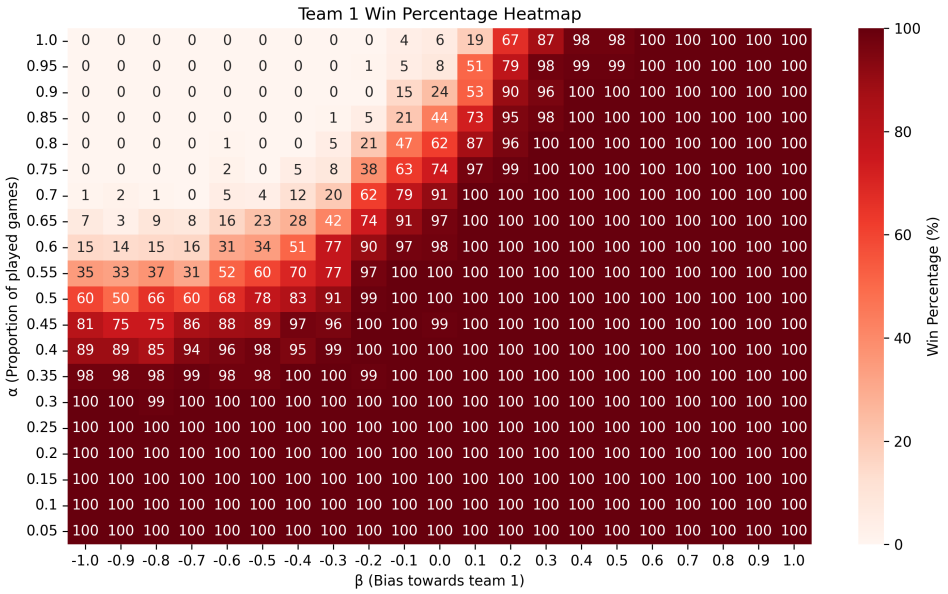

Figure 1.1 – Team 1 win percentage heatmap for varied (α, β) pairs

1.4.2 Time Measurement

Figure 1.2 presents the average execution time per instance (in microseconds) for an increasing number of teams. The plot uses a log-log scale to clearly show how the performance of the solution scales with the number of participating teams. For each fixed team count, 100 instances were generated using $\alpha = 0.5$, $\beta = 0.0$, and two games per pairing. The number of teams was incremented using powers of $\sqrt{2}$, specifically, $\sqrt{2}^i$ for $i = 2 \ldots 17$.

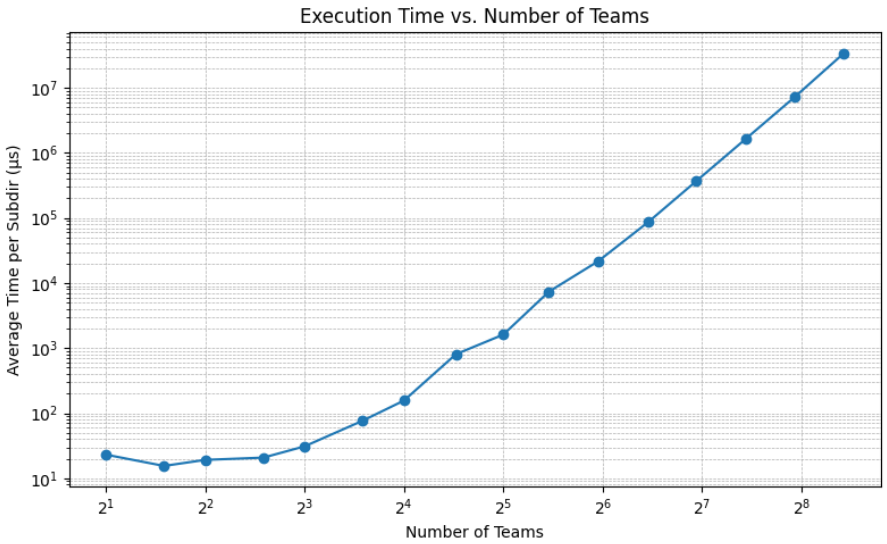

Figure 1.2 – Execution time in microseconds for incremental number of teams

1.5 Conclusion

Analyzing the results, it is possible to formulate some hypotheses and conjectures:

- **Hypothesis 1:** As observed in Section 1.4.1 and Figure 1.1, the impact of bias becomes more significant as the proportion of games already played increases. This suggests that the fraction of games completed directly enhances the effect of bias on team 1's ability to still win. When $\alpha \leq 0.35$, team 1 is consistently able to win, regardless of bias, likely because a substantial number of matches remain. This results in a "border"effect in the heatmap, where the shift in win probabilities creates a triangular pattern, particularly along the transition between areas where team 1 is able or not able to win.

- **Hypothesis 2:** As observed in Section 1.4.2 and Figure 1.2, the near-linear appearance of the log-log plot suggests that the execution time grows polynomially with the number of teams. This implies a relationship of the form $T(\mu s) = a \cdot (\text{team\_size})^k$, where a and k are constants. Empirically, as the team size increases, the execution time exhibits a noticeably aggressive growth trend.

Based on the findings, it can be concluded that the *Winning a Tournament* problem can be efficiently reduced to a max-flow formulation in realistic scenarios, enabling the determination of whether a given team can still win the tournament.

REFERÊNCIAS

Robert Sedgewick, *Algorithms for the Masses*, ANALCO'11, San Francisco, January 2011. Note: Modi Memorial Lecture, Drexel University, April 2012.

David S. Johnson, *A theoretician's guide to the experimental analysis of algorithms*, Proceedings of the 5th and 6th DIMACS Implementation Challenges, 2002.

Robert Sedgewick and Kevin Wayne, *Algorithms*, 4th edition, Addison-Wesley, 2011.

Diego Amorim, *Maxflow Algorithms*. GitHub repository. Available at: https://github.com/diegohommer/maxflow-algorithms. Accessed May 2025.

Marcus Ritt, *Notas de Aula INF05016*. Available at: https://www.inf.ufrgs.br/ mrpritt/lib/exe/fetch.php?media=inf05016:notas-0f24280.pdf. Accessed May 2025, page 102.

Marcus Ritt, *Vencendo Torneios*. GitHub repository. Available at: https://github.com/mrpritt/vencendo-torneios. Accessed April 2025.