Algorithms and Data Structures
Practicum 2 - Actor Actress Matching

Elianne Heuer, s4320514
Jordy Aaldering, s1004292

January 9, 2020

Algorithm

We start by reading in the input and creating an adjacency list. First we read in the actresses and actors and put them in a dictionary containing their name, gender, and index in the graph. We then iterate through all movies to fill the graph. An edge is then added between two stars of a movie if they have a different gender.

Next we find the number of edges of the maximal matching of the graph. We do this by applying the depth first search function bpm to each actress to find a matching actor. bpm looks for an optimal edge between an actress and actor to add to the maximal matching. It does this by first checking if that vertex has been visited, if it hasn't been visited we check whether that vertex hasn't been assigned or if it can be connected to another vertex via a recursive step. To make sure that the vertex isn't connected to the same vertex again in the recursive step we mark the vertex as visited. bmpMax simply loops through all actresses and calls on bpm for each actress to see if its assignment can be added to the maximal matching. bpm then returns true if an edge was added and false otherwise. If bpm returned true we add one to the result.

Finally we compare number of edges in the maximal matching to the number of actresses. If these are the same we know that every edge is in the maximal matching, and thus that we have a perfect matching. If this is the case Mark wins, otherwise Veronique wins. This result is then printed to the screen.

Correctness

Proof that we get the correct number of edges in the maximal matching: We iterate over all actresses and check if it is possible to add an edge containing this actress to the maximal matching. Because the existing edges are only moved and only a single edge is added containing the current actress, we know that the result is increased by exactly one. We do not have to check the effects of removing an edge completely from the maximal matching because the resulting maximal matching would have the exact same number of edges and thus would be a useless operation. So it follows that we indeed get the correct number of edges in the maximal matching.

Proof that Mark wins if there is a perfect matching P: Let Mark currently be at vertex v. Since P is a perfect matching we know that edge $P(v)$ has not been taken. So Mark will always choose edge $P(v)$. Since the edges chosen by Veronique never belong to the perfect matching Mark never gets stuck. So Mark always wins if there is a perfect matching.

Proof that Veronique wins if there is no perfect matching: Let M be a maximal matching, and let v be a vertex that is not in M. It then follows that an edge (v, w) chosen by Mark is not in M. Veronique then chooses edge $M(w)$ if possible, and again Mark can never choose an edge in M. So Veronique can choose an edge in M as long as there is one adjacent to the vertex that Mark chose. Now if we assume that Veronique gets stuck, it must follow that the edges chosen by the players trace out an alternating path with Mark having one extra edge, showing that M was not a maximal matching. But we know that M is a maximal matching, so by contradiction we have proven that Veronique always wins if there is no perfect matching.

Complexity

Reading in the actors and actresses both have a time complexity of $\mathcal{O}(n)$. There are m movies, each movie can have at most $n + n$ stars that have to be connected. Stars are connected by comparing them to the already added stars of that movie. So reading in the movies has a time complexity of: $\mathcal{O}(m \cdot (n + n) \cdot (n + n)) = \mathcal{O}(m \cdot n^2)$. Making the total input of reading in the data and creating the graph equal to $\mathcal{O}(m \cdot n^2)$.

Making a bipartite matching for a vertex uses a depth first search approach, and thus has a time complexity of $\mathcal{O}(|V| + |E|)$, which in our case results in a time complexity of $\mathcal{O}((n + n) + (n \cdot n)) = \mathcal{O}(n^2)$. This function is called in a loop that is executed $n + n$ times to make a maximal matching, combining this gives us a time complexity of $\mathcal{O}((n + n) \cdot n^2) = \mathcal{O}(n^3)$.

Combining all this we get that the time complexity will be $\mathcal{O}(m \cdot n^2 + n^3)$. In the case that there are more movies than actors or actresses, the time complexity of the algorithm will be $\mathcal{O}(m \cdot n^2)$ and in the case that there are less movies than actors or actresses, the time complexity will be $\mathcal{O}(n^3)$.