Analysis of Algorithms 2 Homework 3

Ahmed Burak Gulhan
150160903

A)
<u>Question:</u>
Let the current team with the highest points in the league T, and its points P. Instead of using a maximum-flow based method, could not we just use a simple criterion such
that "If current points of a team plus the number of remaining matches a team has >= P, that
team still has a chance to win the league".

<u>Answer:</u>
No. We cannot say for certain that team P will win if there are still some remaining matches. Lets assume there are 3 teams A, B and C. Let's assume that team A will win all of it's matches and it's score will be 3. Assume that B and C have initial scores of 0 and there are 7 C-B matches remaining. There is no circumstance that A will win the league. The 7 points from the matches remaining will have to go to a team. If C and B win and lose 3 matches each their scores will be 3 and 3 respectively, which is the same as team A's maximum score. The last remaining point from the C-B match-up will go to either B or C, which will have a score of 4 which is larger than A's.

B)
compile with ./g++ -std=c++11 main.c
run using ./a.out <input file> <output file>
or
./a.out <input file> (output saved to output1.txt and result is printed to screen)

C)
To solve the problem given in the homework we must convert the problem into a network flow problem. This can be better explained using an example.

Assume we have 4 teams with scores 33 29 28 27. Lets call these teams a,b,c and d respectively. These teams have the following match matrix which shows the remaining matches between any two teams:

0 1 6 1
1 0 0 3
6 0 0 1
1 3 1 0

We have 1 a-b match, 6 a-c matches, 1 a-d match, 3 b-d matches and 1 c-d match.

Let's check if <u>team b</u> has a chance of winning the league.

We assume that team b will win all of it's matches. Therefore the maximum score W of team b is equal to 29 + 4 = 33.

Therefore team b can win the league *if and only if* the following conditions hold true (being tied with first place counts as winning):

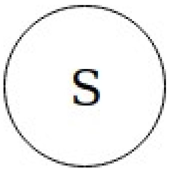
- Team a has no more than R(a) = 33 - 33 = 0 wins in the remaining games.
- Team c has no more than R(c) = 33 - 28 = 5 wins in the remaining games.
- Team d has no more than R(d) = 33 – 27 = 6 wins in the remaining games.

Let P be a set of all teams except for team b
P = {a, c, d}
Let Q be a set of all possible unique pairs of matches that does not include team b (since we assumed team b has won every game)
Q = {{a,c}, {a,d}, {c,d}}

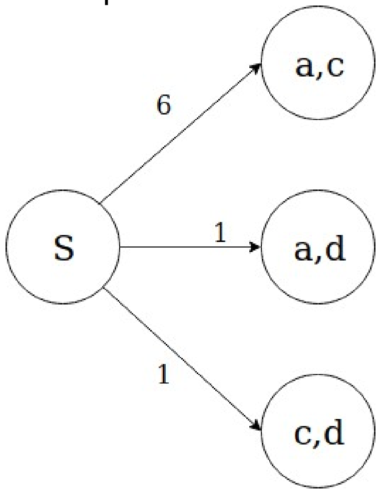
Let G be the total number of matches remaining between the teams in set P
G = 6+1+1 = 8

This problem can be solved by making and solving a flow graph.

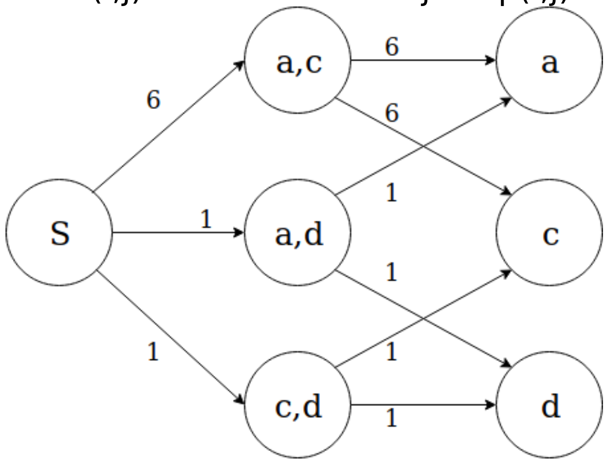
1. We make our source node s. This is where all games originate



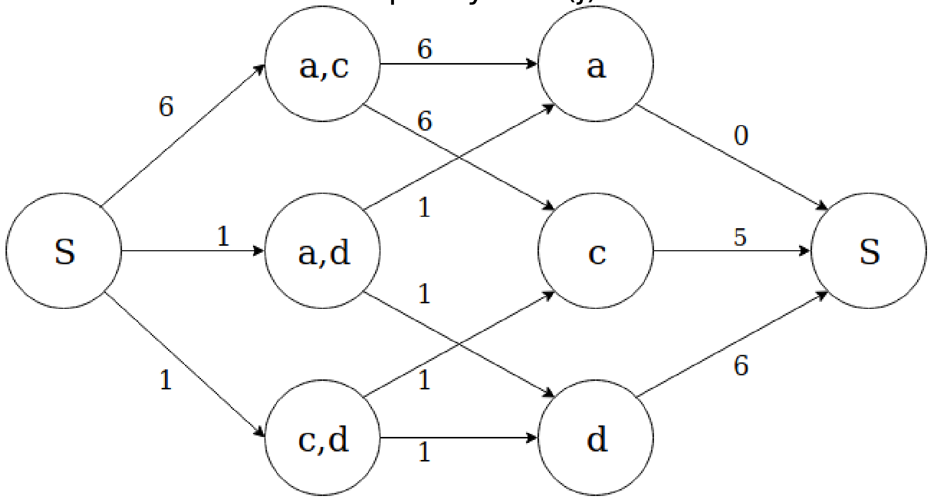
2. Make a node for each pair in Q. Draw a connection form s to each pair in Q and set the connection capacity as the number of matches left for each pair.



3. Make a node for each team in P. For each Q node (i,j) add connections from (i,j) to P nodes i and j. cap(i,j) → i = cap(i,j) → j = cap(S) → (i,j)



4. Make a sink node T. Make a connection from any P node j to T and set the connection capacity as R(j).



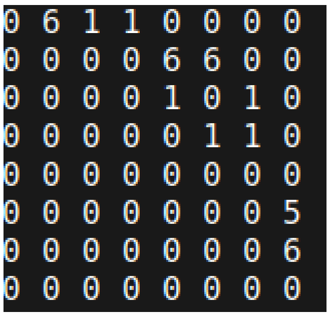
5. Find the max flow in this graph.
   If the max flow == G, then team b has a chance of winning the league.
   If not, then team b has no chance of winning the league.

We can use Ford Fulkerson algorithm to calculate the max flow.
The max flow is found as 5. Since 5 != G = 6, team b has nor chance of winning the league.

In summary what we did was for team x we found the maximum possible score W by assuming team x has won all games. We then calculated the constraints for the remaining teams so that if these constraints R(j) hold true after playing all the remaining matches team x has a chance of winning. We then built a flow graph using this information. If our flow graph's maximum flow is equal to G (total number of remaining matches) then this means that the constraints we found hold true for at least 1 combination of wins and loses. If the maximum flow if not equal to G then this means that the constraints we set are impossible to be met after playing all remaining matches.

By applying this algorithm for each team we can solve the problem for this homework.

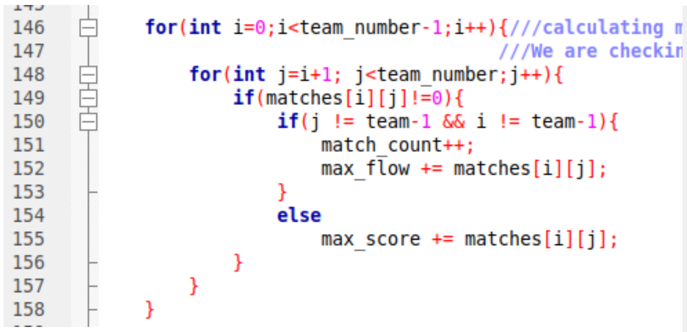
In my code I made a function that generates a flow graph in matrix form using the above algorithm. For this example the flow graph matrix for team b is:



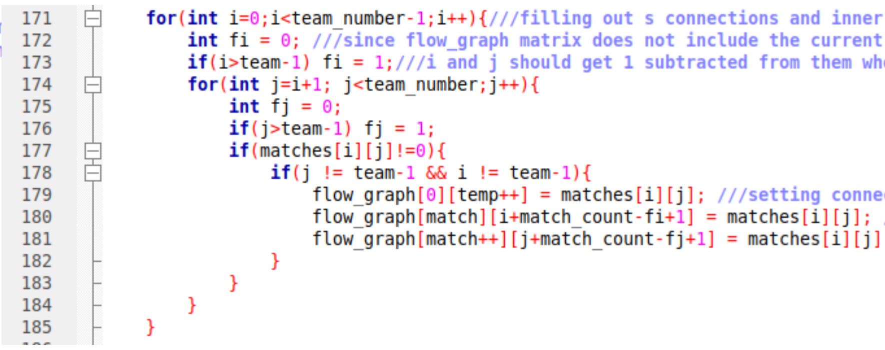
```
0 6 1 1 0 0 0 0
0 0 0 0 6 6 0 0
0 0 0 0 1 0 1 0
0 0 0 0 0 1 1 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 5
0 0 0 0 0 0 0 6
0 0 0 0 0 0 0 0
```

Complexity analysis:

In my code the function check_win() is used to calculate whether or not a given team has a chance to win the league. In this function there are 2 iterations in the upper half of the nxn score matrix (n teams). The code segments are shown below.



```
for(int i=0;i<team_number-1;i++){///calculating m
                                 ///We are checkin
    for(int j=i+1; j<team_number;j++){
        if(matches[i][j]!=0){
            if(j != team-1 && i != team-1){
                match_count++;
                max_flow += matches[i][j];
            }
            else
                max_score += matches[i][j];
        }
    }
}
```



```
for(int i=0;i<team_number-1;i++){///filling out s connections and inner
    int fi = 0; ///since flow_graph matrix does not include the current
    if(i>team-1) fi = 1;///i and j should get 1 subtracted from them wh
    for(int j=i+1; j<team_number;j++){
        int fj = 0;
        if(j>team-1) fj = 1;
        if(matches[i][j]!=0){
            if(j != team-1 && i != team-1){
                flow_graph[0][temp++] = matches[i][j]; ///setting conne
                flow_graph[match][i+match_count-fi+1] = matches[i][j];
                flow_graph[match++][j+match_count-fj+1] = matches[i][j]
            }
        }
    }
}
```

The time complexity for an iteration in the upper half of an nxn matrix is $O((n-1)^2)2)$ which is $O(n^2)$

This function then generates a flow matrix, which then has the Ford Fulkerson algorithm applied to it.
The size of the flow matrix = size of P + size of Q + 2 (s and t nodes)

- P and Q are defined above in the algorithm explanation

The size of P = n-1
The maximum size of Q = $(n-1)^2 – (n-1)$

- $n>=2$

Therefore size of flow matrix = $n-1 + (n-1)^2 – (n-1) + 2 = n^2 – 2n + 2$

We already know that the time complexity of the Ford Fulkerson algorithm is O(M*f) where M is the number of edges and f is the maximum flow.

We know that the number of edges in the flow graph = size of P (edges from S to set P) + size of P *2 (edges from set P to set Q) + size of Q (edges from Q to t) = 3 * size of P + size of Q.
We found that that the size of P = n-1 and the size of ! = $(n-1)^2 – (n-1)$.
Therefore we can say that the number of edges in flow graph =
M = $n-1 + (n-1)^2 – (n-1) = n^2 – 2n + 2$

Therefore the time complexity of Ford Fulkerson algorithm for this problem is $O((n^2 – 2n + 2) * f) = O(f*n^2)$ where n is the number of teams in our league.

So to check one team we have $O(n^2) + O(f*n^2) = O(f*n^2)$

Since we have to check all n teams the total time complexity of this problem is:
$n*O(f*n^2) = O(f*n^3)$
where f is the maximum match count for a single type of match and n is the number of teams.

For the space complexity we found that our flow graph matrix size = $n^2 – 2n + 2 = O(n^2)$
In Ford Fulkerson we have a residual graph the size of the input graph which is $O(n^2)$ we also have 2 arrays of size n, which is O(n).
So the space complexity of our problem is $O(n^2) + 2*O(n) = O(n^2)$
This might be made smaller by using a different representation for the flow graph.