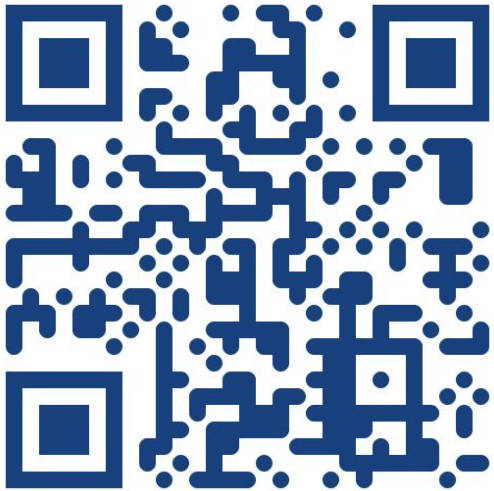
Algorithms and Probability

Exercise Session 11



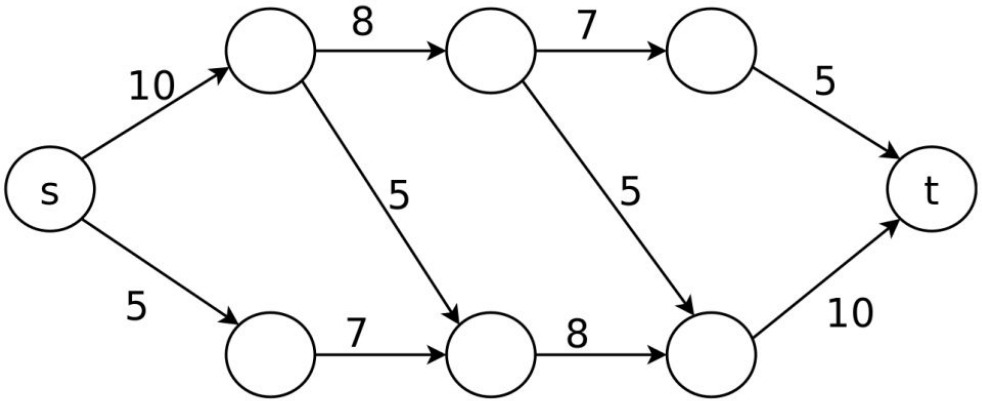
https://n.ethz.ch/~ahmala/anw

Ford Fulkerson

```
1  f(e) ← 0 for all e ∈ A
2  while there is an s—t path P in the residual network do
3        increase flow f along P
4  return f
```

Ford Fulkerson

```
1  f(e) ← 0 for all e ∈ A
2  while there is an s—t path P in the residual network do
3        increase flow f along P
4  return f
```



Time Complexity

- O(val(f_max) * |E|)
- O(U * |V| * |E|)

```
1  f(e) ← 0 for all e ∈ A
2  while there is an s—t path P in the residual network do
3        increase flow f along P
4  return f
```

Better Algorithm

(Dynamic Trees, Sleator-Tarjan, 1983): $O(mn \log n)$

Max Flow in Almost Linear Time

https://www.quantamagazine.org/researchers-achieve-absurdly-fast-algorithm-for-network-flow-20220608/

The build is basically dynamic trees adapted to tree-based graph approximators. Brief guide to technical details from an OI data structures perspective(source https://codeforces.com/blog/entry/100510?#comment-892347)

- Write flows in linear algebraic notation, bash some inequalities, turn the problem into supporting about m approximate minimum-ratio cycles on a dynamically changing graph. (Section 4 & 9)
- Observe that routing along low stretch trees give such approximate solutions to the min-ratio cycle problem. (Dijkstra running in mlogn time is OP) (Section 2.2. for static, dynamic version in Section 7)
- Start with one such tree embedding, throw HLD/separators/contractions at it to maintain partial routings to end points of vertices involved in updates. (Section 6)
- Problem on these vertices involved in updates becomes maintaining spanners under vertex splits. In static cases, this can be done by expanders, which are structureless graphs. Maintain such graphs with another round (or two) of segtrees... (Section 5)
- Things can still break because the costs of the partial routing significantly decreased. Do some coordinated rebuild of the k layers of recursion (Section 8).
- Carefully analyze the interactions of the randomness against the future updates, using stability of the optimal update step. (start of Sections 6, then Section 9)

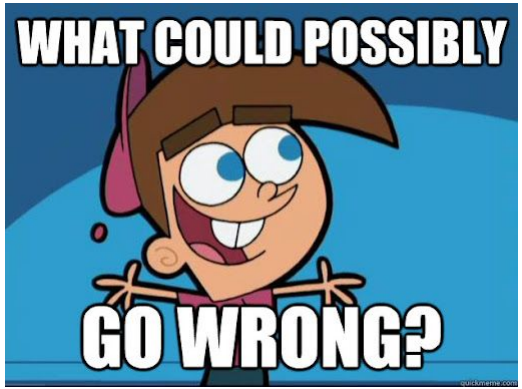
Max Flow in Almost Linear Time

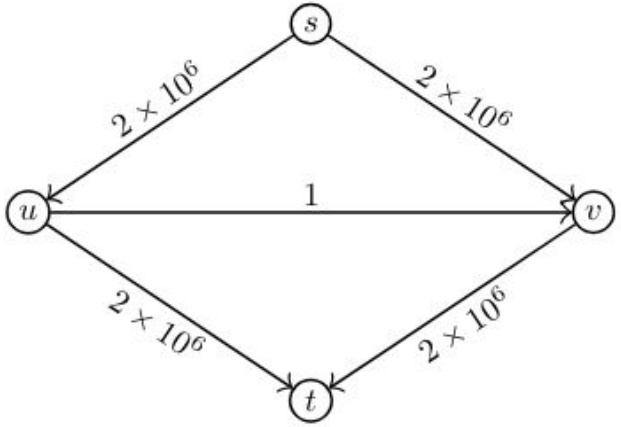
https://www.quantamagazine.org/researchers-achieve-absurdly-fast-algorithm-for-network-flow-20220606/

The build is basically dynamic trees adapted to tree-based graph approximators. Brief guide to technical details from an OI data structures perspective(source: https://codeforces.com/blog/entry/100510?#comment-892347)

- Write flows in linear algebraic notation, bash some inequalities, turn the problem into supporting about approximate minimum-ratio cycles on a dynamically changing graph. (Section 4 & 9)
- Observe that routing along low stretch trees give such approximate solutions to the min ratio cycle problem. (Dijkstra running in mlogn time is OP) (Section 2.2. for static, dynamic version in Section 7)
- Start with one such tree embedding, throw HLD/separators/contraction at it to maintain partial routings to end points of vertices involved in updates. (Section 6)
- Problem on these vertices involved in updates become maintaining spanners under vertex splits. In static cases, this can be done by expanders, which are structureless graphs. Maintain such graphs with clusters rooted (or whatever) segtrees... (Section 5)
- Things can still break b[illegible] the costs of the partial routing significantly decreased. Do some coordinated rebuild of the k layers of recursion (Section 8).
- Carefully analyze the interaction of the randomness against the future updates, using stability of the optimal update step. (start of Sections 6, then Section 9)

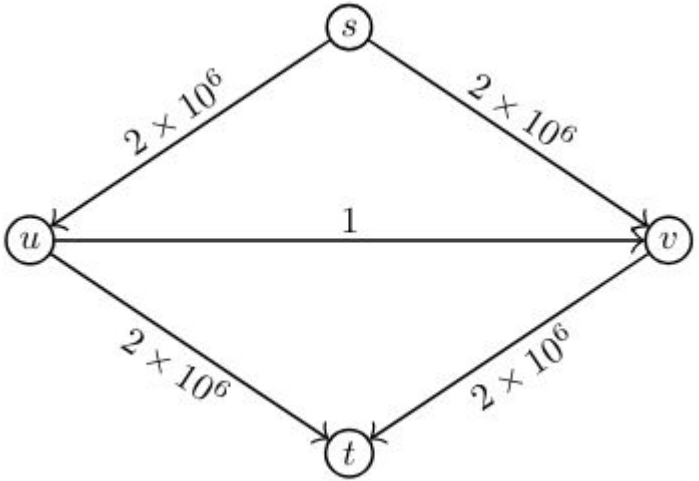
How worse is Ford Fulkerson?





How worse is Ford-Fulkerson?

- maximum value of a flow is 4 million
- Ford-Fulkerson algorithm might always choose a path that uses the edge uv by alternately choosing suvt and svut.
- This way the value of the flow increases by 1 in every step, so it takes 4 million improvements to reach a maximum flow.

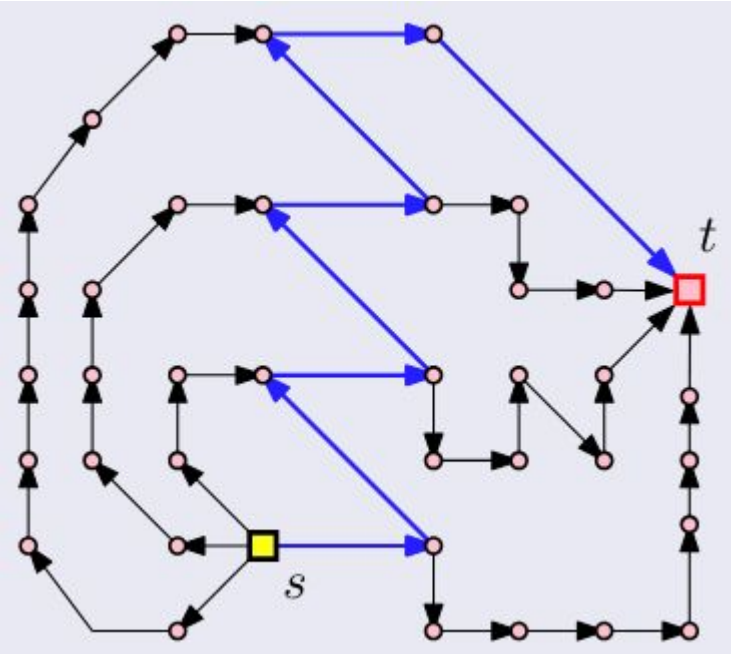


Is this Correct?

Removing the shortest path from source to target decreases the max-flow either by 0 or 1.

NOPE

~~Removing the shortest path from source to target decreases the flow either by 0 or 1.~~



Integrality Theorem

If the all capacities are integers, then there is an integer maximum flow.

Prove or Disprove!

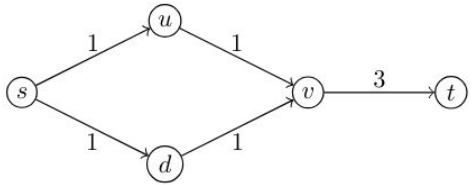
Let G be a network with source s, sink t, and integer capacities.

a) If all capacities are even then there is a maximal flow f such that f(e) is even for all edges e.
b) If all capacities are odd then there is a maximal flow f such that f(e) is odd for all edges e.

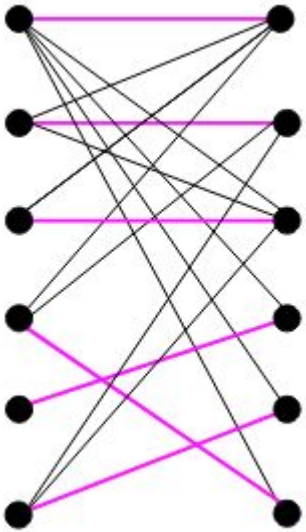
Solution

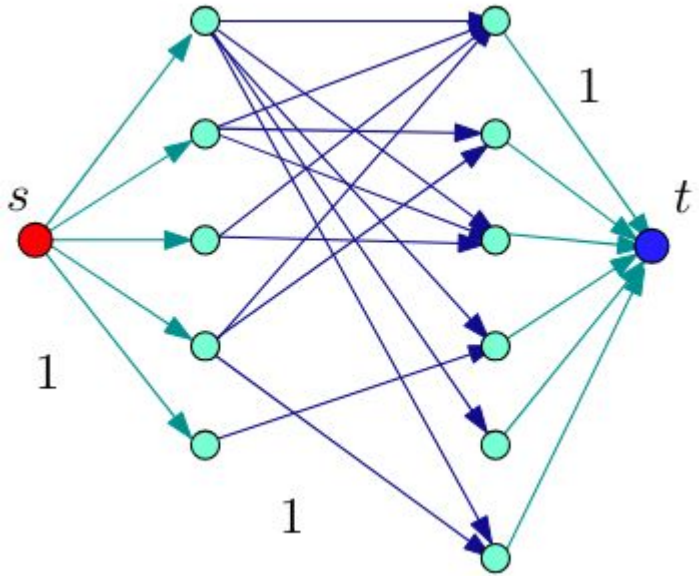
(a) Dividing all capacities by two, we obtain a network with integral capacities. Therefore, it has an integral maximum flow. Multiplying this flow by 2, we get an even maximum flow on the original network.

(b) Counterexample:



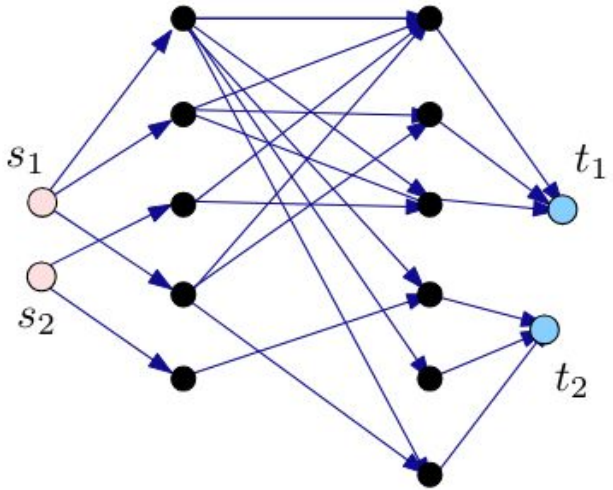
Bipartite Matching in O(nm)




s
t
1
1
1

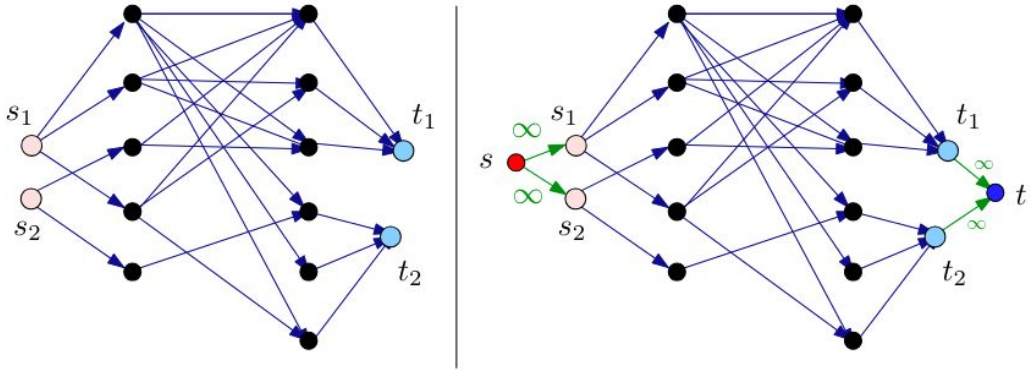
Multiple Sources and Sinks

Given a flow network with several sources and sinks, how can we compute maximum flow on such a network?



Multiple Sources and Sinks

The idea is to create a super source, that send all its flow to the old sources and similarly create a super sink that receives all the flow. Clearly, computing flow in both networks in equivalent.



Minimum Cut with terminal nodes(sink & source)

Apply Max-Flow Algorithm (nm log(n))

Minimum Cut with and without terminal nodes(sink & source)

With Max-Flow Algorithm $O(n^4 * \log(n))$

Minimum Cut with and without terminal nodes(sink & source)

With Randomized Approach in O(n^2):

```
1  while G has more than two vertices do
2      e ← u.a.r. edge from G
3      G ← G/e
4  return number of edges left over
```

Repeat it n^2 * log(n) times, take the minimum of the returned values.

End up with O(n^4 * log(n))

Minimum Cut with and without terminal nodes(sink & source)

$$\Pr[\mu(G) = \mu(G/e)] \geq \Pr[e \notin C] = 1 - \frac{|C|}{|E|} \geq 1 - \frac{k}{kn/2} = 1 - \frac{2}{n},$$

For n small, we have large failure rate.

Bootstrapping

- When a constant number of vertices left, use a deterministic&correct algorithm (e.g. Max-Flow)
- With bootstrapping, you get O(n^2 * poly(log(n))).

In Class Exercise

Aufgabe 1 – *Meisterschaft*

Die Mannschaften aus *Bedigliora*, *Caslano*, *Novaggio*, *Pura* und *Sessa* spielen um die Meisterschaft. Während der Saison muss jede Mannschaft sechs mal gegen jede andere Mannschaft antreten, und bei jedem Spiel wird genau ein Punkt an den Sieger der Begegnung vergeben (es gibt immer einen Sieger und einen Verlierer). Die aktuelle Tabelle ist rechts angegeben. Dabei hat Bedigliora noch 5 ausstehende Spiele gegen Caslano, 3 gegen Sessa und 6 gegen Novaggio. Caslano hat noch 2 ausstehende Spiele gegen Sessa und 5 gegen Novaggio. Sessa muss noch 3 mal gegen Novaggio antreten.

Mannschaft	Punkte
Caslano	8
Novaggio	7
Bedigliora	6
Sessa	3
Pura	0

Pura verpflichtet mit sofortiger Wirkung einen Superstar, um doch noch die Meisterschaft zu gewinnen. Aber können sie das überhaupt noch? *Gibt es einen weiteren Saisonverlauf, bei dem Pura am Ende auf einem (womöglich geteilten) ersten Platz landet?*

Modellieren Sie das Problem als Flussproblem in einem Netzwerk.

Hinweis: Wie viele Spiele müsste Pura noch gewinnen, und wie viele dürfen die anderen Mannschaften noch gewinnen? Benutzen Sie diese Informationen, um die Netzwerkkapazitäten geeignet festzulegen.

In Class Exercise

Total Number Of Games?

How Many Matches of Pura left?

How many points other teams can get max so Pura gets 1.place?

Aufgabe 1 – *Meisterschaft*

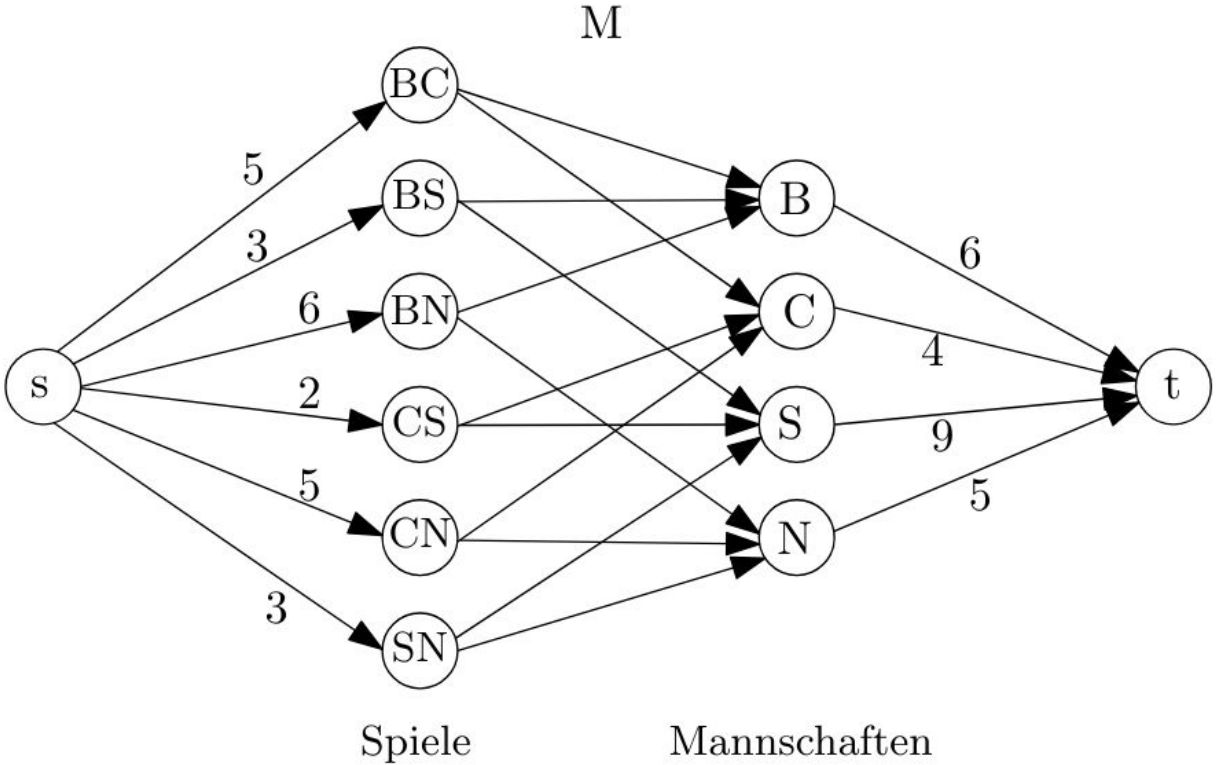
Die Mannschaften aus *Bedigliora*, *Caslano*, *Novaggio*, *Pura* und *Sessa* spielen um die Meisterschaft. Während der Saison muss jede Mannschaft sechs mal gegen jede andere Mannschaft antreten, und bei jedem Spiel wird genau ein Punkt an den Sieger der Begegnung vergeben (es gibt immer einen Sieger und einen Verlierer). Die aktuelle Tabelle ist rechts angegeben. Dabei hat Bedigliora noch 5 ausstehende Spiele gegen Caslano, 3 gegen Sessa und 6 gegen Novaggio. Caslano hat noch 2 ausstehende Spiele gegen Sessa und 5 gegen Novaggio. Sessa muss noch 3 mal gegen Novaggio antreten.

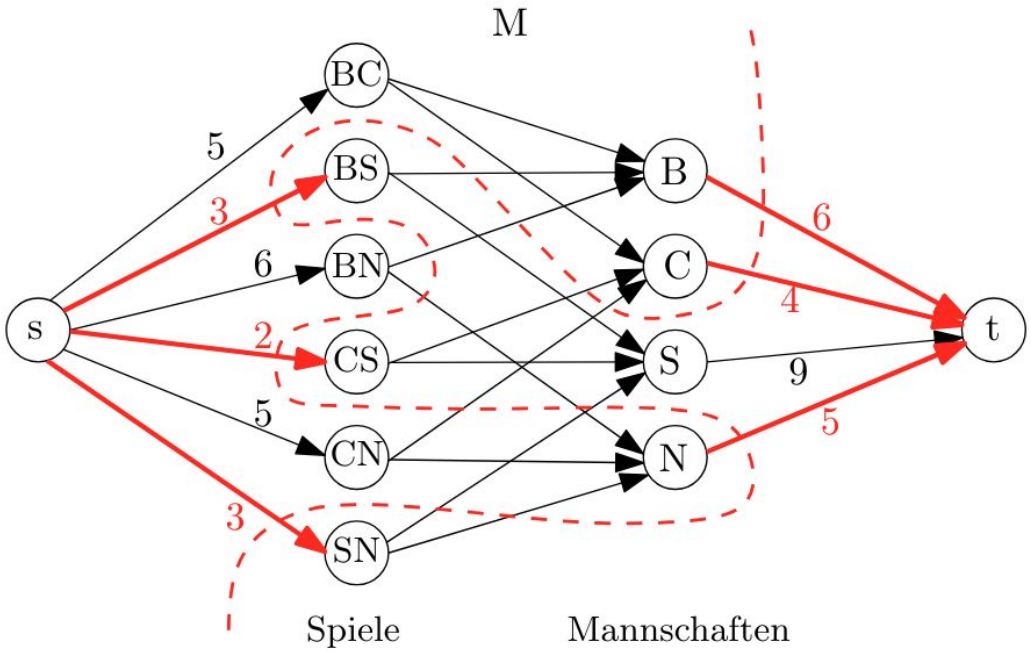
Mannschaft	Punkte
Caslano	8
Novaggio	7
Bedigliora	6
Sessa	3
Pura	0

Pura verpflichtet mit sofortiger Wirkung einen Superstar, um doch noch die Meisterschaft zu gewinnen. Aber können sie das überhaupt noch? *Gibt es einen weiteren Saisonverlauf, bei dem Pura am Ende auf einem (womöglich geteilten) ersten Platz landet?*

Modellieren Sie das Problem als Flussproblem in einem Netzwerk.

Hinweis: Wie viele Spiele müsste Pura noch gewinnen, und wie viele dürfen die anderen Mannschaften noch gewinnen? Benutzen Sie diese Informationen, um die Netzwerkkapazitäten geeignet festzulegen.


M
BC
BS
BN
CS
CN
SN
B
C
S
N
s
t
5
3
6
2
5
3
6
4
9
5
Spiele
Mannschaften


M
BC
BS
BN
CS
CN
SN
B
C
S
N
s
t
5
3
6
2
5
3
6
4
9
5
Spiele
Mannschaften