Maximum Matchings

Maximum Matchings

problem definition

Maximum Matchings
problem definition

- We define

$$M \subseteq E \text{ is a matching of } G = (V, E)$$
$$\text{iff } \forall e_1, e_2 \in M : e_1 \cap e_2 = \emptyset$$

Maximum Matchings
problem definition

- We define

$$M \subseteq E \text{ is a matching of } G = (V, E)$$
$$\text{iff } \forall e_1, e_2 \in M : e_1 \cap e_2 = \emptyset$$

- In an unweighted graph, a maximum matching is a matching of maximum cardinality

Maximum Matchings
problem definition

- We define

$$M \subseteq E \text{ is a matching of } G = (V, E)$$
$$\text{iff } \forall e_1, e_2 \in M : e_1 \cap e_2 = \emptyset$$

- In an unweighted graph, a maximum matching is a matching of maximum cardinality

- In a weighted graph, a maximum matching is a matching such that the sum over the included edges is maximum

Maximum Matchings
problem definition

- We define

$$M \subseteq E \text{ is a matching of } G = (V, E)$$
$$\text{iff } \forall e_1, e_2 \in M : e_1 \cap e_2 = \emptyset$$

- In an unweighted graph, a maximum matching is a matching of maximum cardinality
- In a weighted graph, a maximum matching is a matching such that the sum over the included edges is maximum
- BGL does not provide weighted matching algorithms

Maximum Matchings

general unweighted case

- A vertex that is not covered by M is called *free*

Maximum Matchings

general unweighted case

- A vertex that is not covered by M is called *free*
- A path that alternates between edges in $E \setminus M$ and edges in M is called *alternating path*

Maximum Matchings

general unweighted case

- A vertex that is not covered by M is called *free*
- A path that alternates between edges in $E \setminus M$ and edges in M is called *alternating path*
- An *augmenting path* is an alternating path that starts and ends at a free vertex

Maximum Matchings

general unweighted case

- A vertex that is not covered by M is called *free*
- A path that alternates between edges in $E \setminus M$ and edges in M is called *alternating path*
- An *augmenting path* is an alternating path that starts and ends at a free vertex
- We can improve any matching by "flipping" the edges along an augmenting path

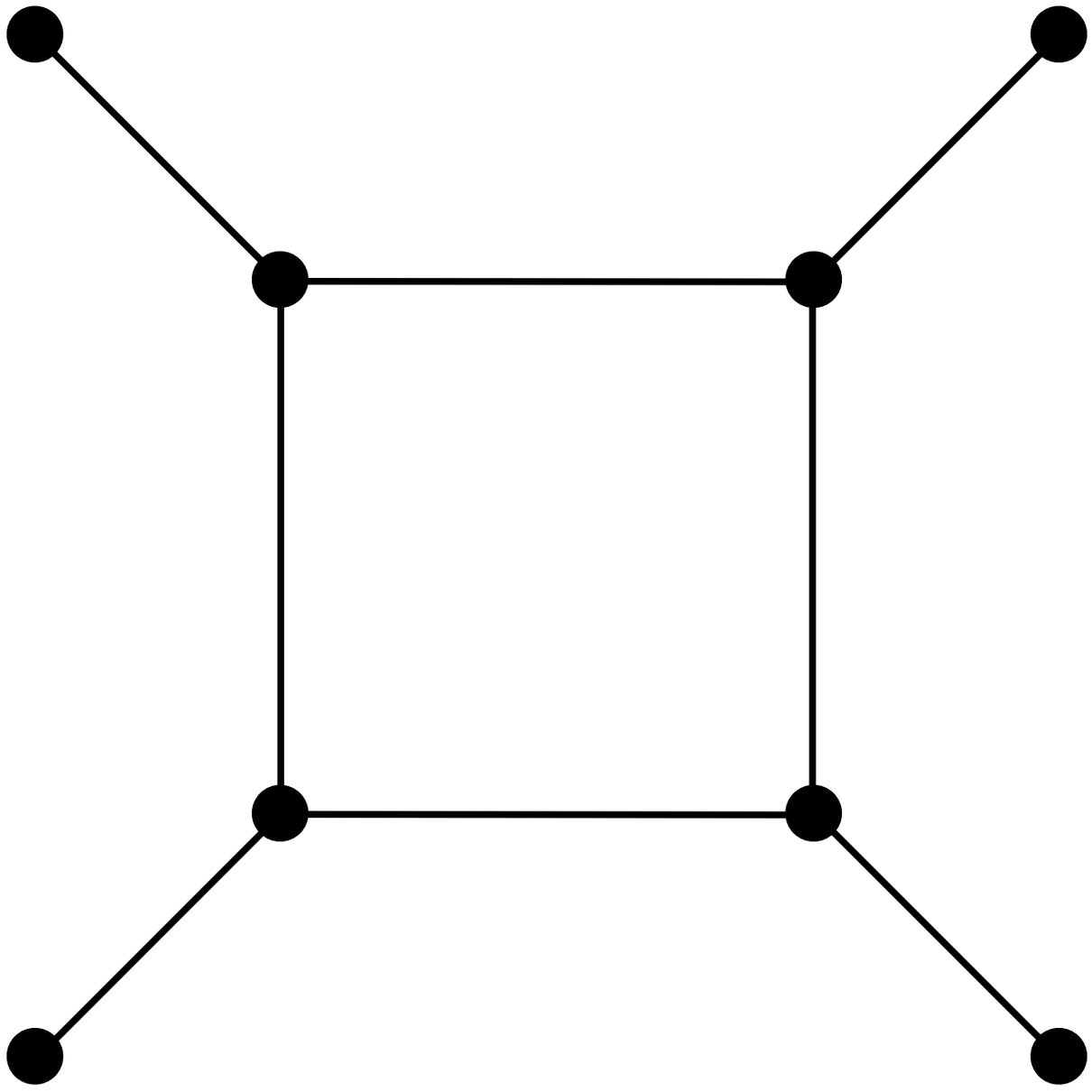
Maximum Matchings

general unweighted case

- A vertex that is not covered by M is called *free*
- A path that alternates between edges in $E \setminus M$ and edges in M is called *alternating path*
- An *augmenting path* is an alternating path that starts and ends at a free vertex
- We can improve any matching by "flipping" the edges along an augmenting path
- A matching is a maximum (not only maximal) matching, iff there is no augmenting path!

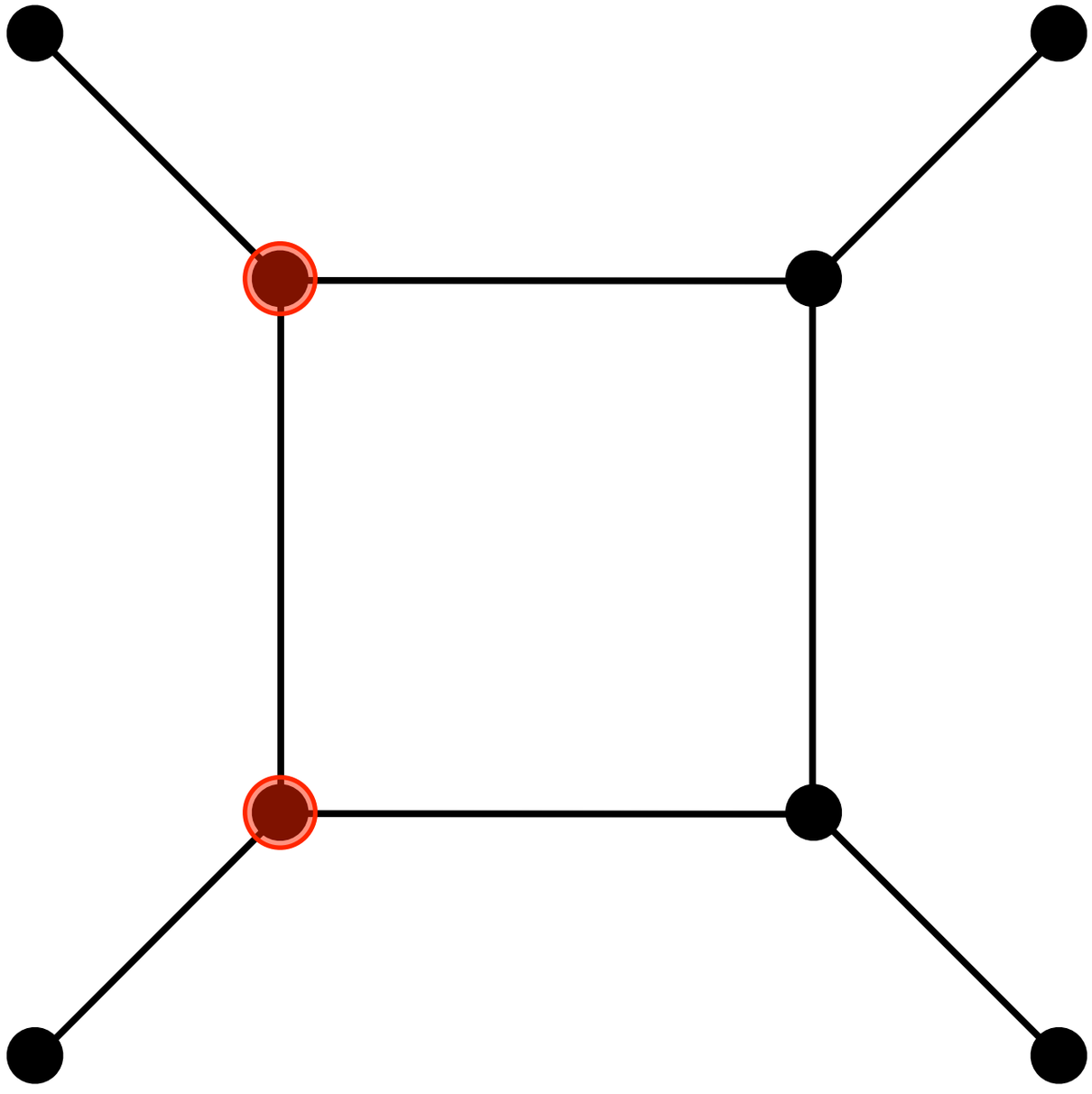
Maximum Matchings

general unweighted case example



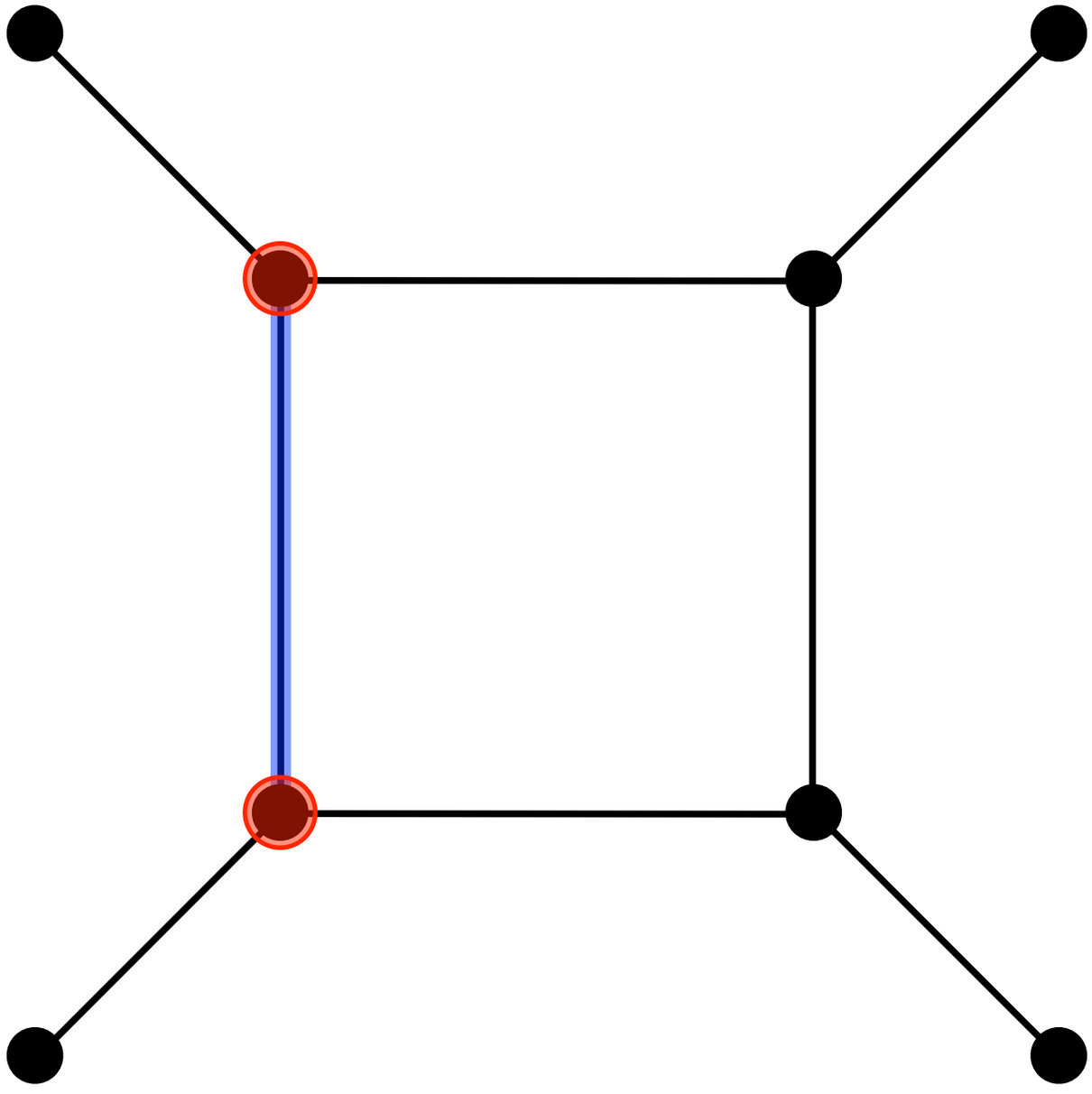
Maximum Matchings

general unweighted case example



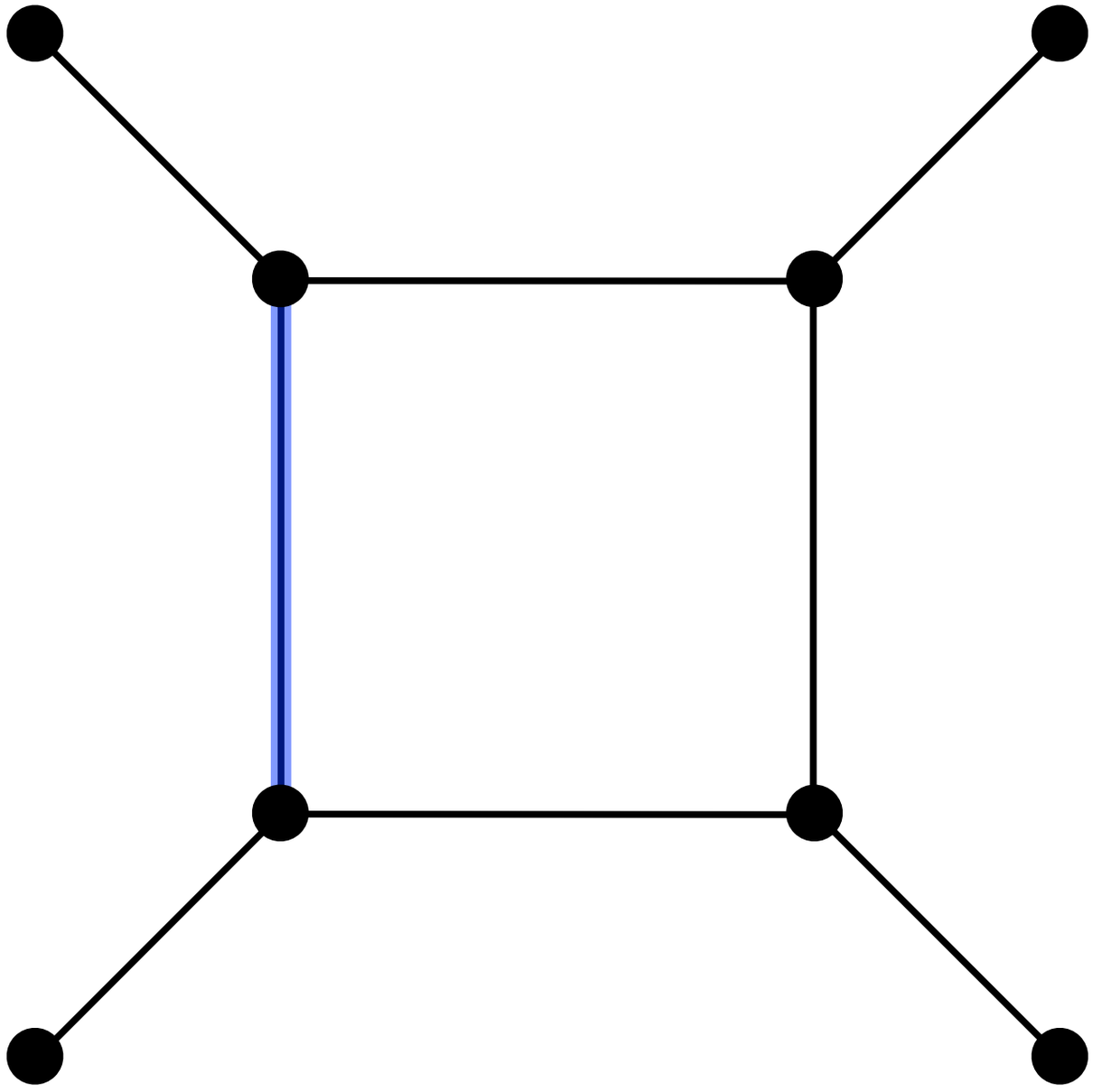
Maximum Matchings

general unweighted case example



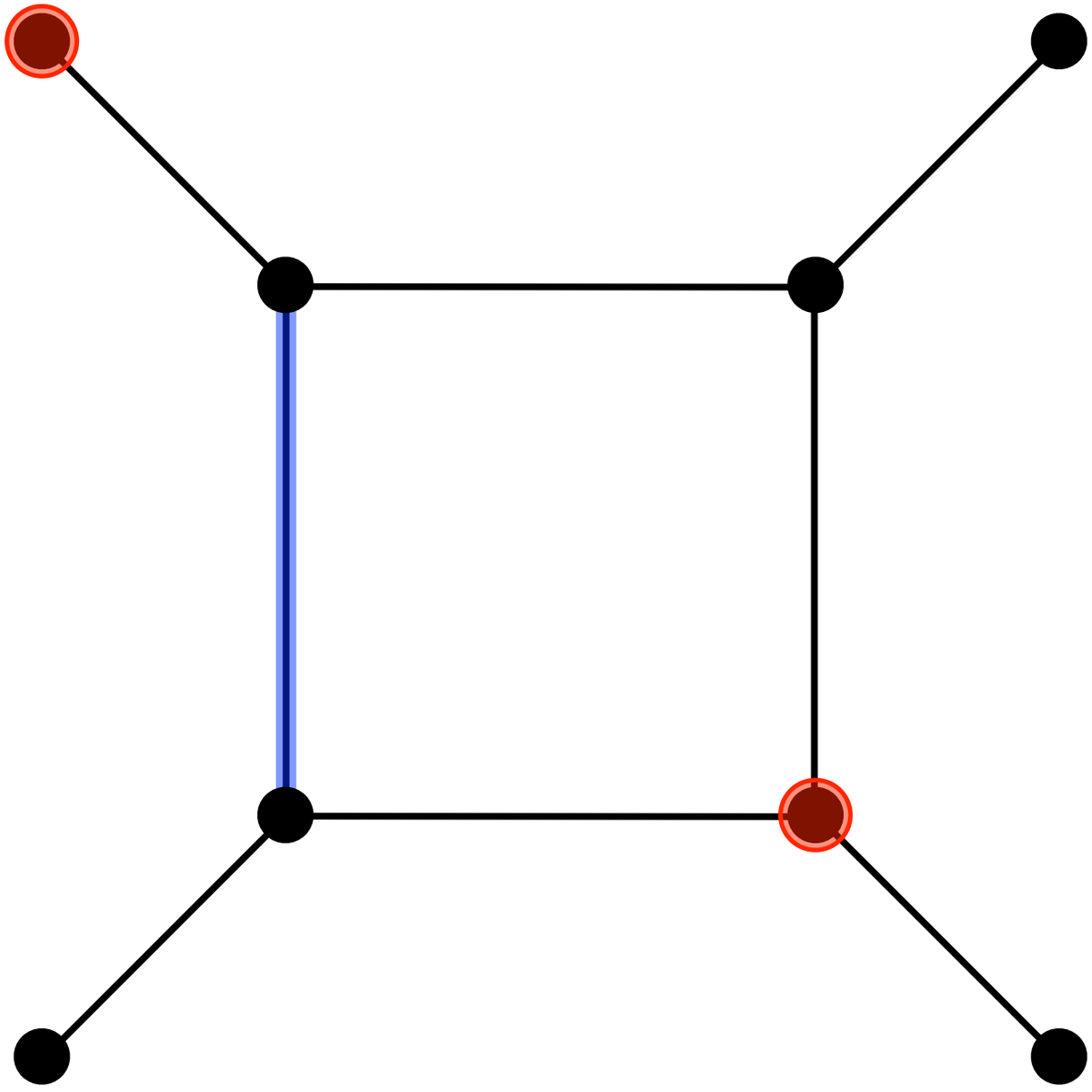
Maximum Matchings

general unweighted case example



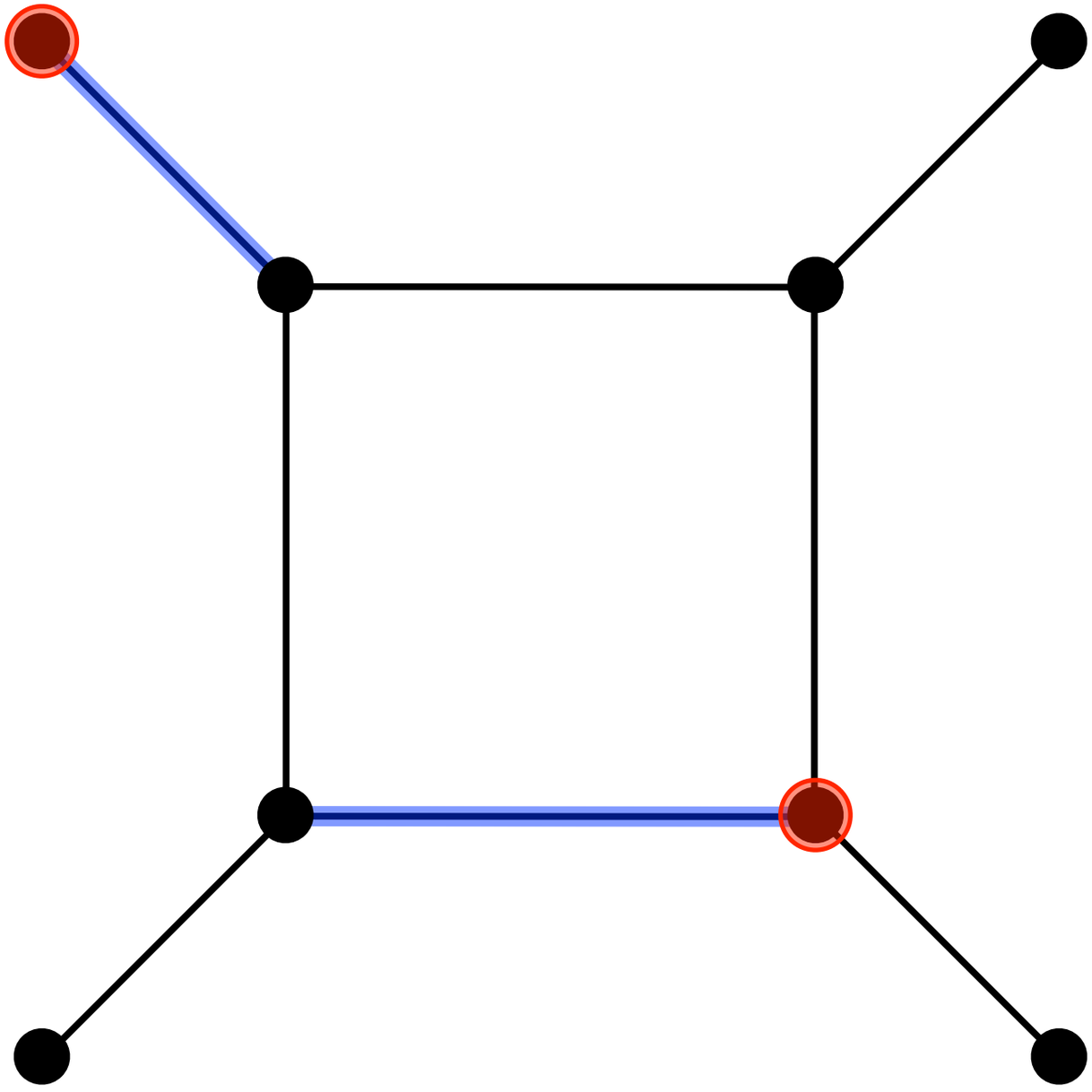
Maximum Matchings

general unweighted case example



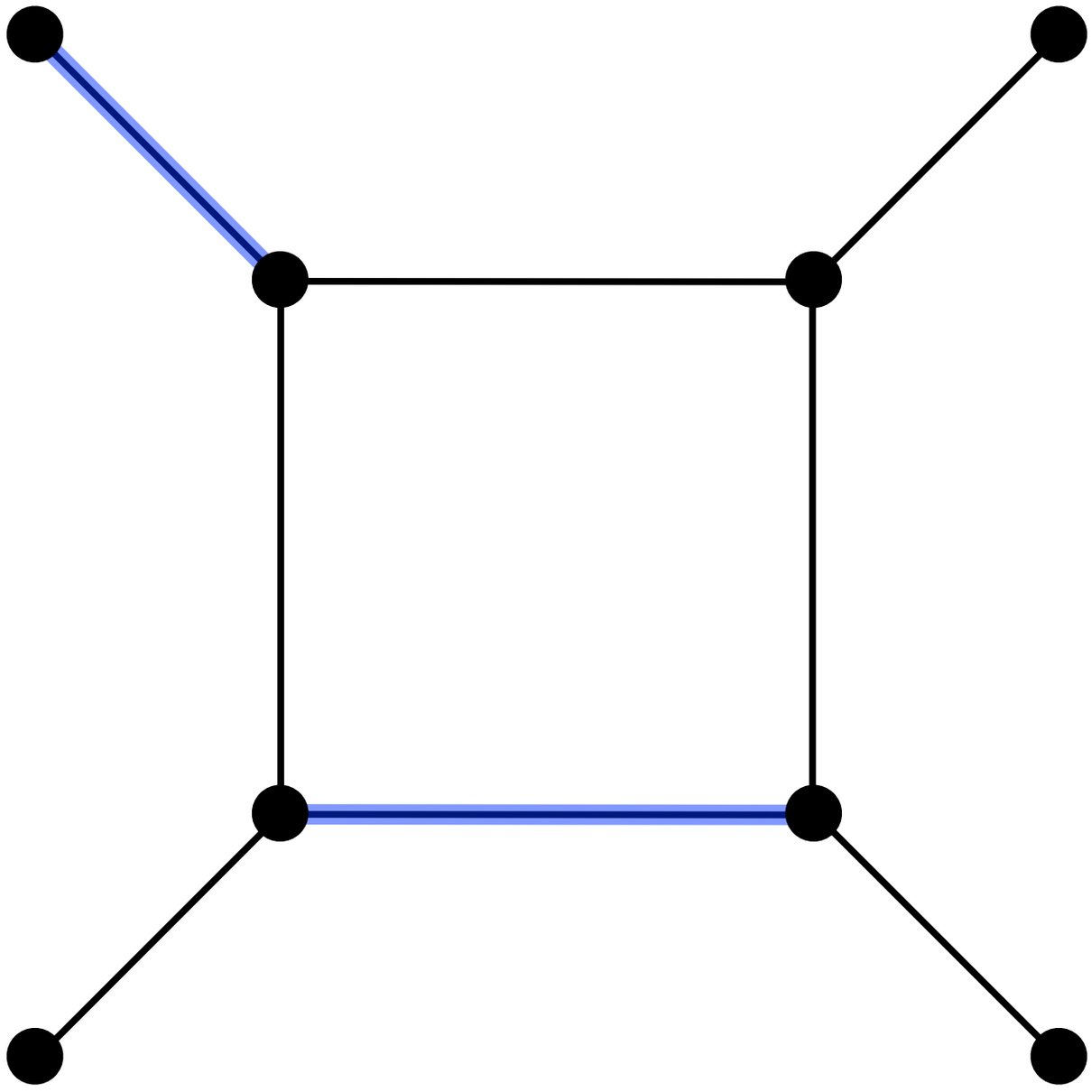
Maximum Matchings

general unweighted case example



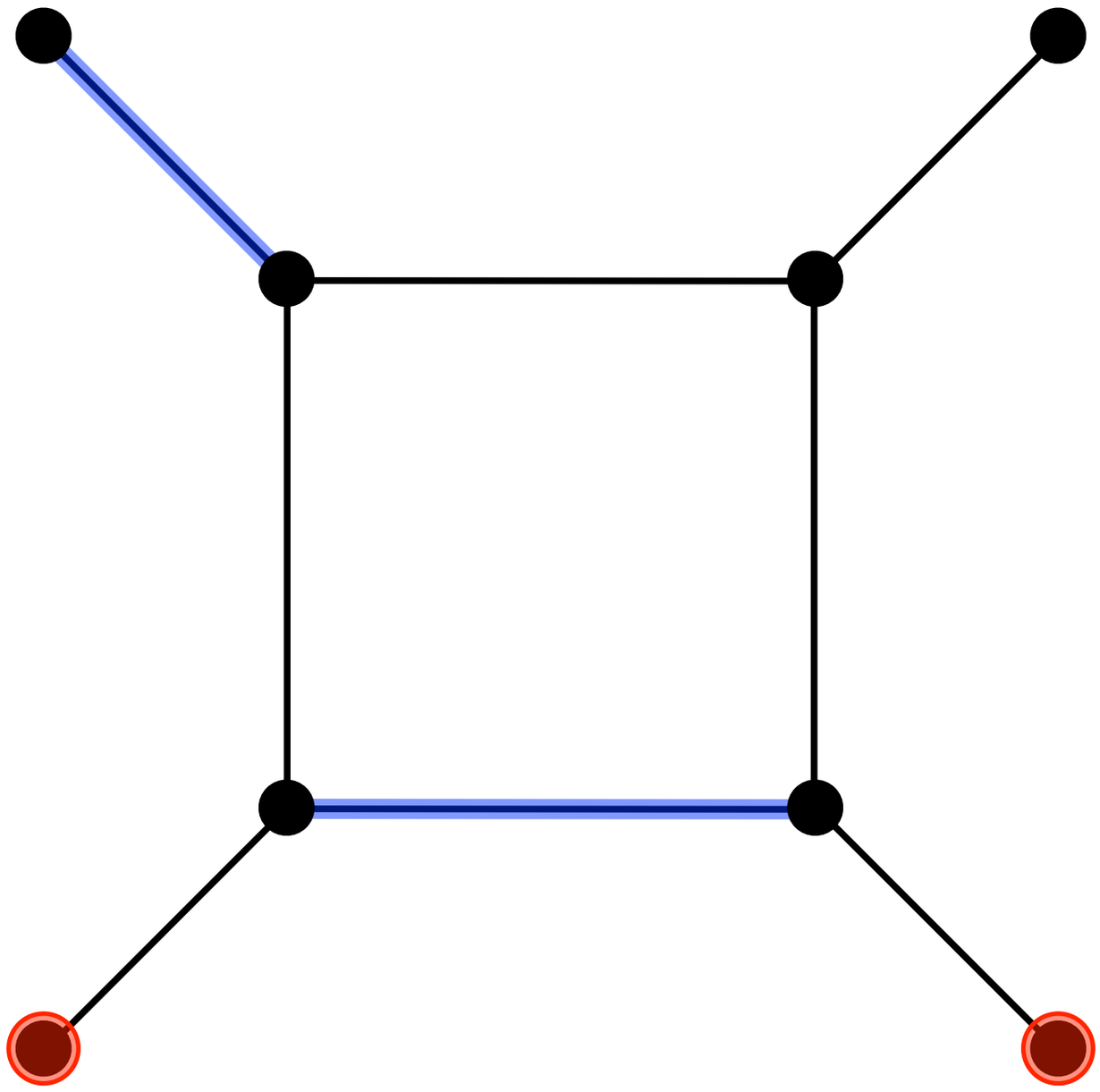
Maximum Matchings

general unweighted case example



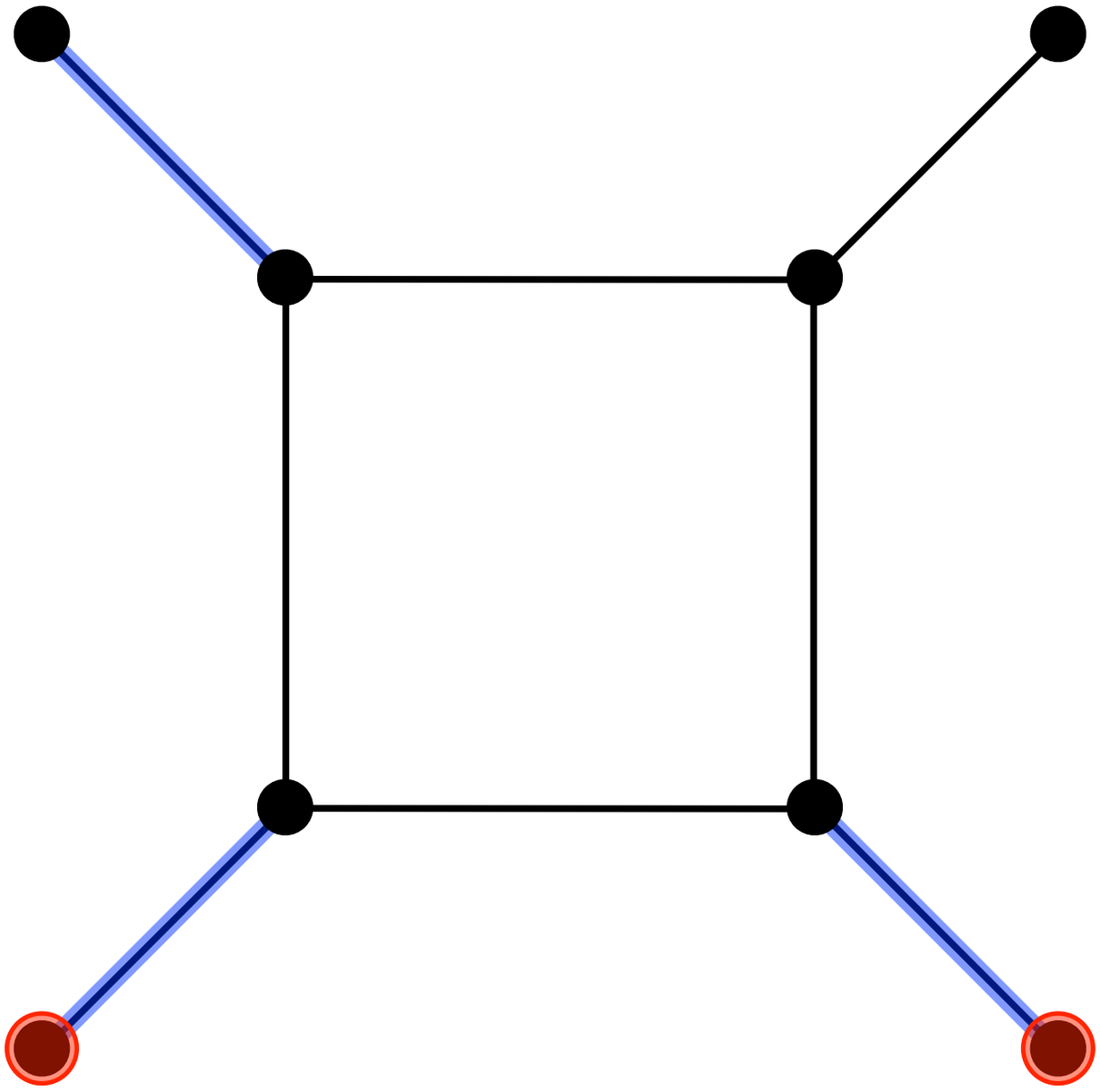
Maximum Matchings

general unweighted case example



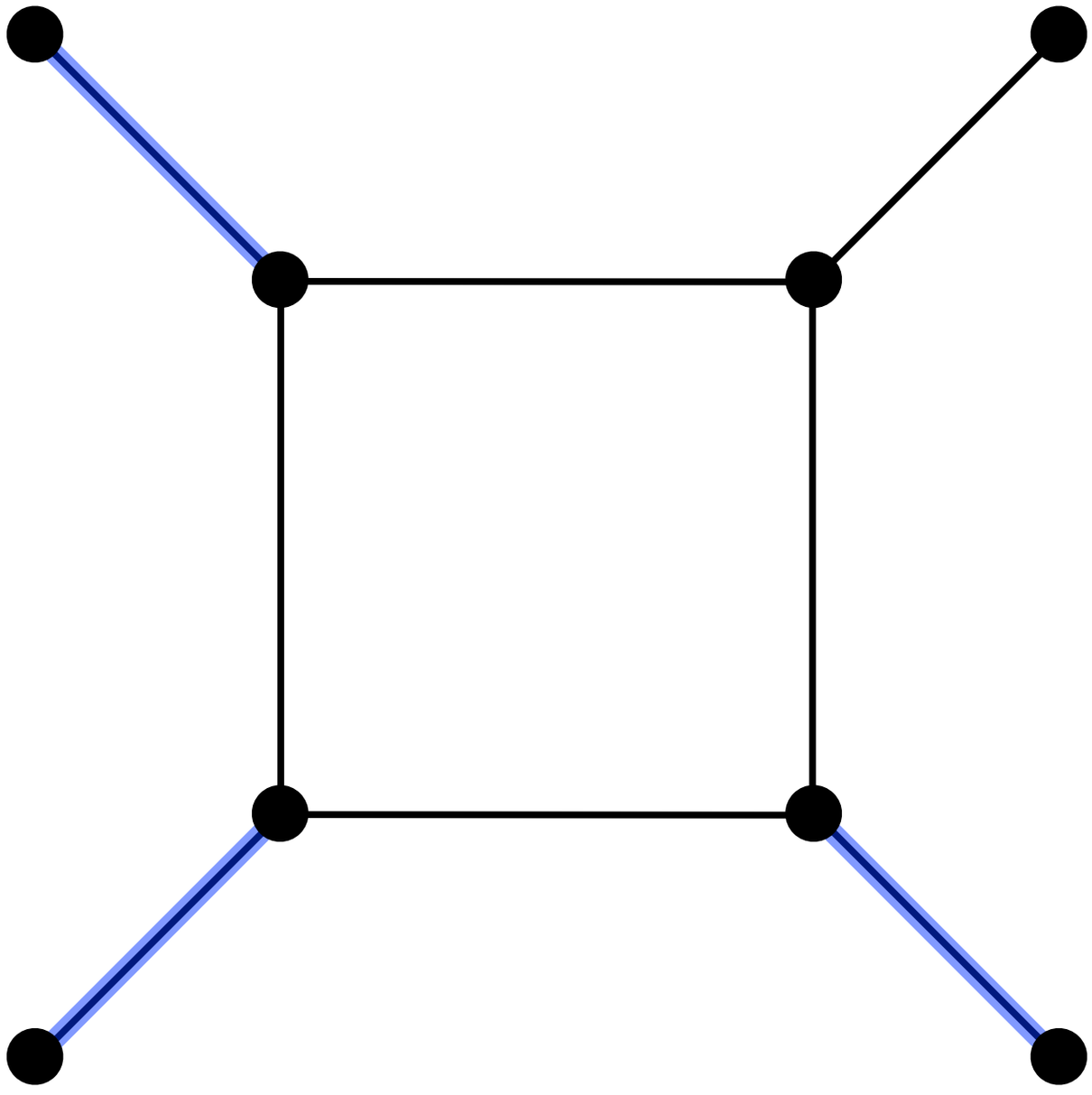
Maximum Matchings

general unweighted case example



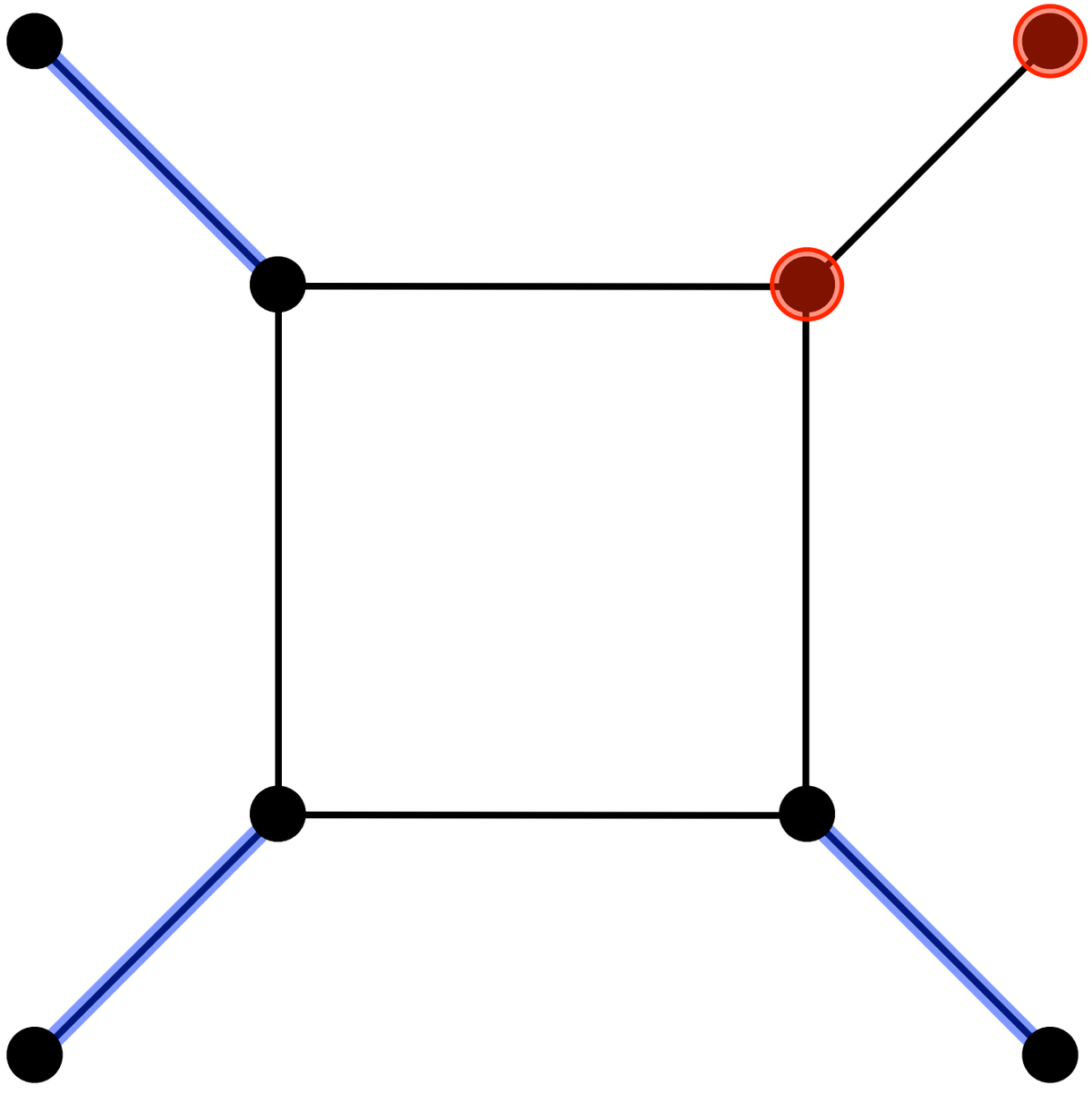
Maximum Matchings

general unweighted case example



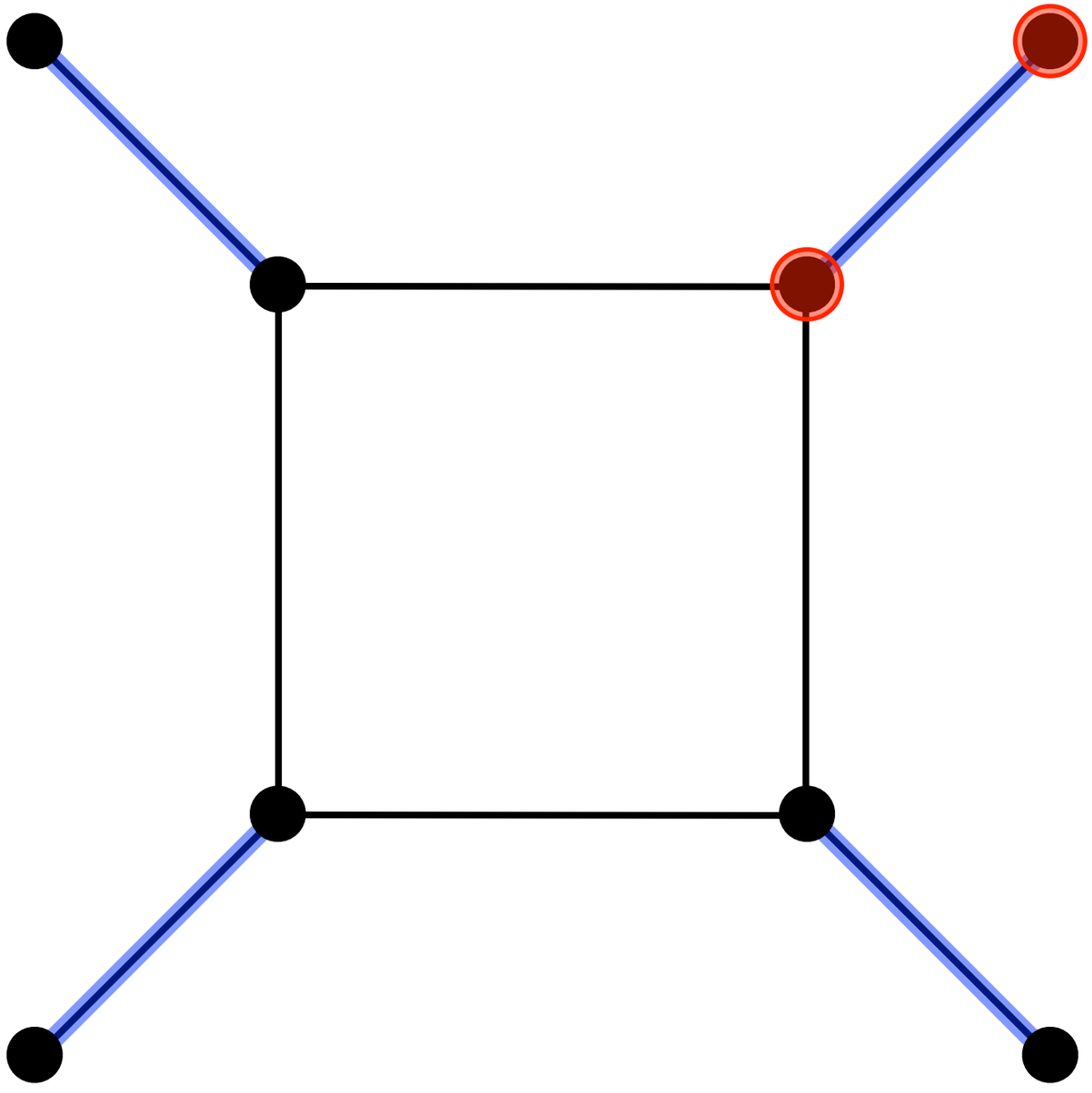
Maximum Matchings

general unweighted case example



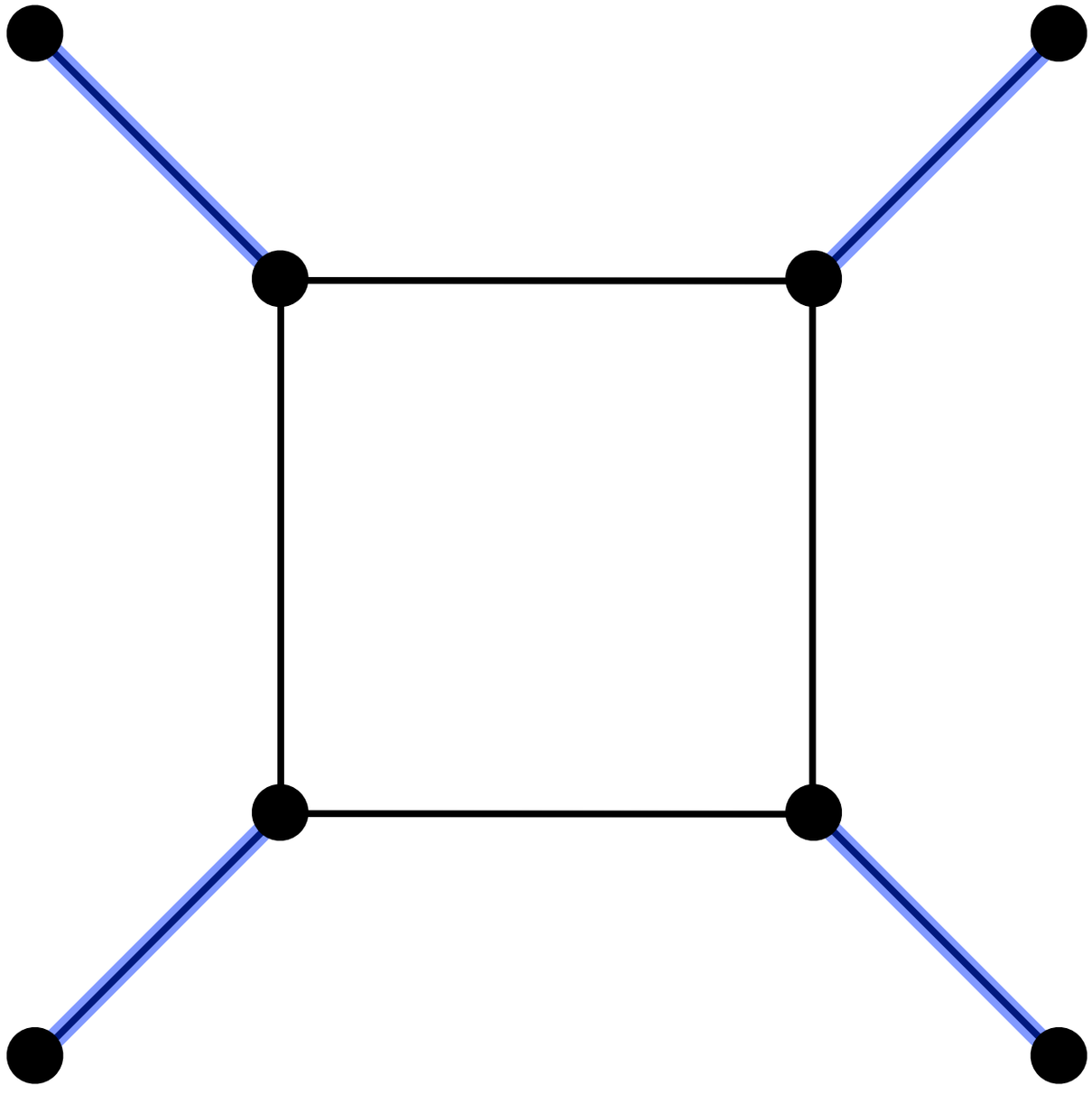
Maximum Matchings

general unweighted case example



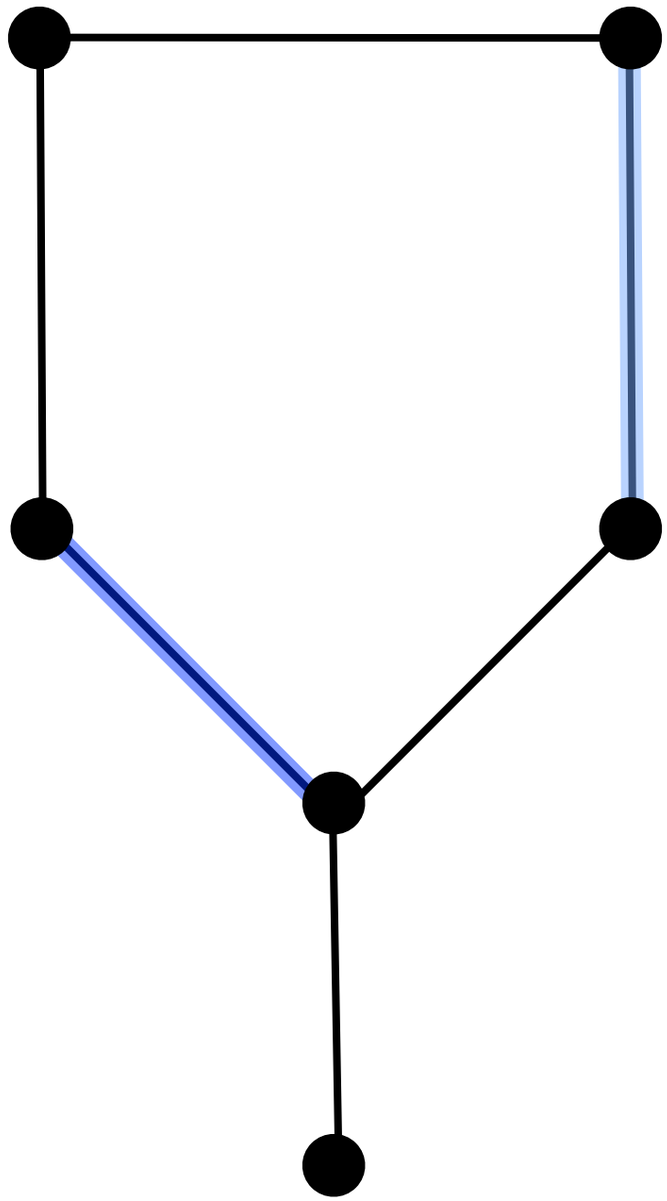
Maximum Matchings

general unweighted case example



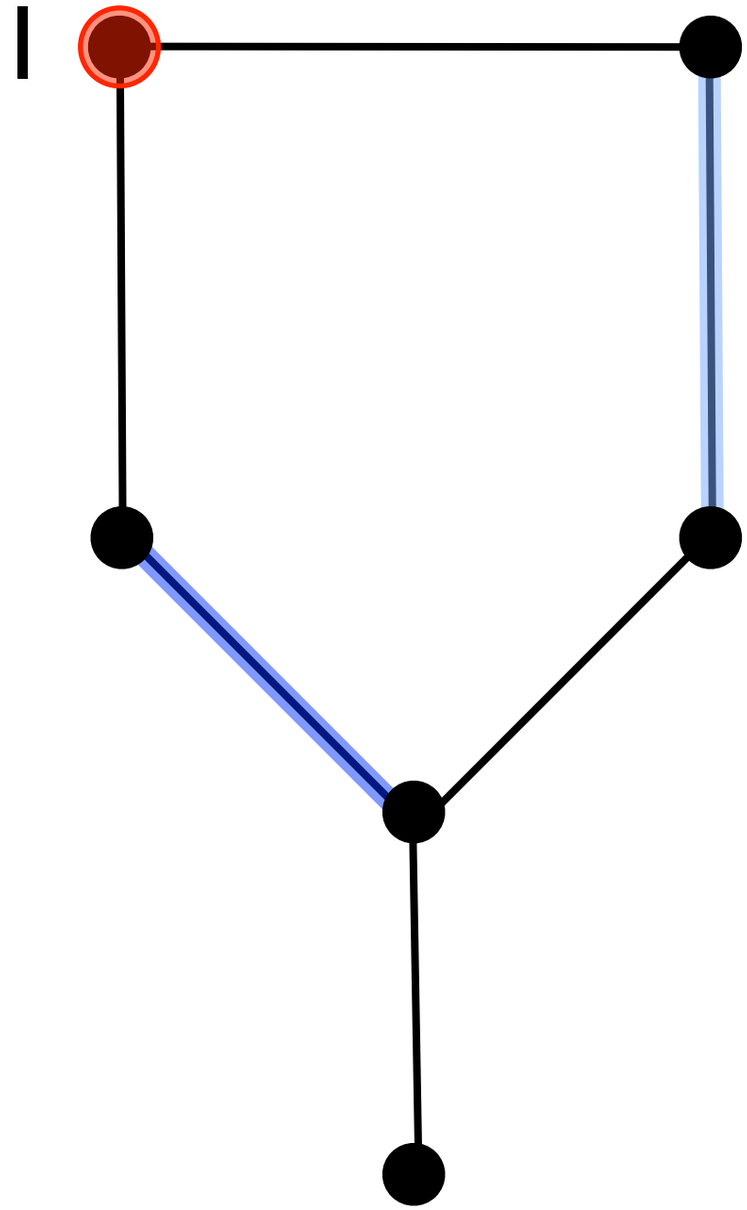
Maximum Matchings

but...



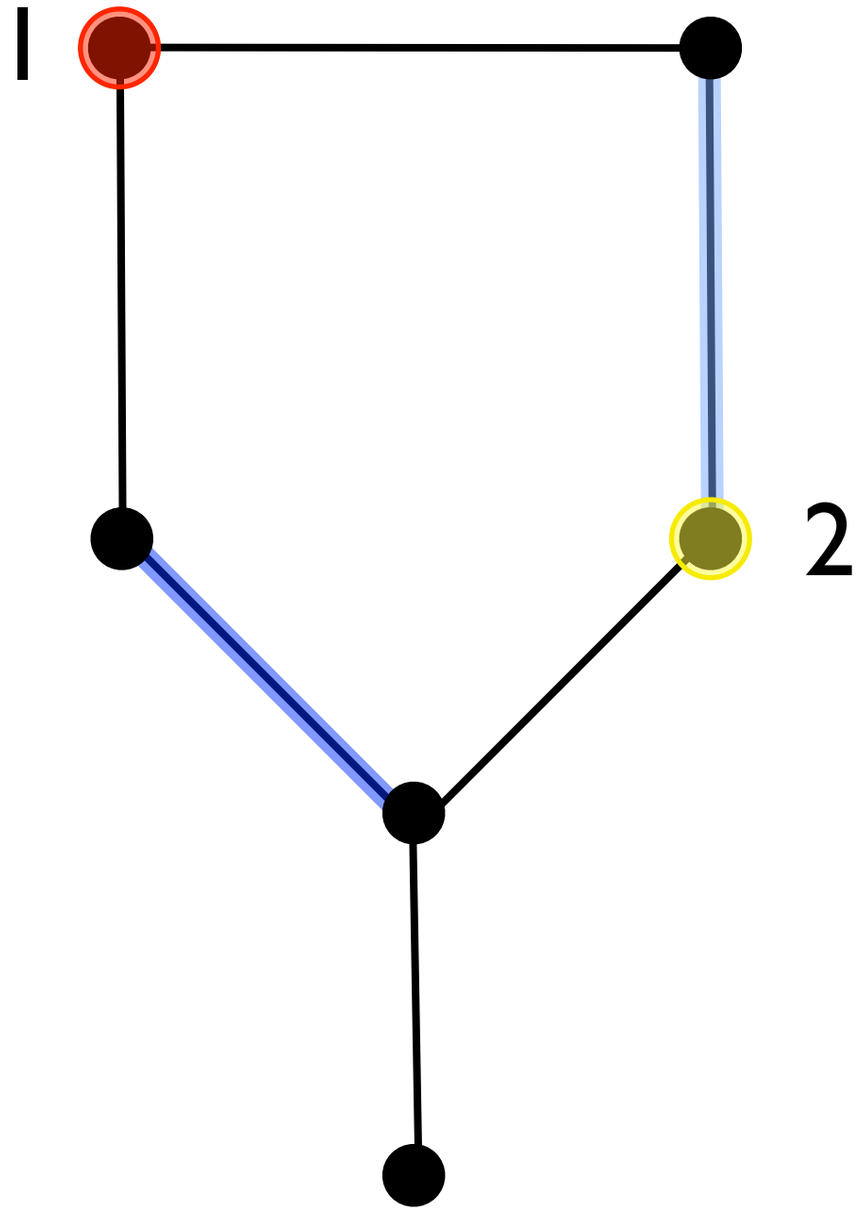
Maximum Matchings

but...



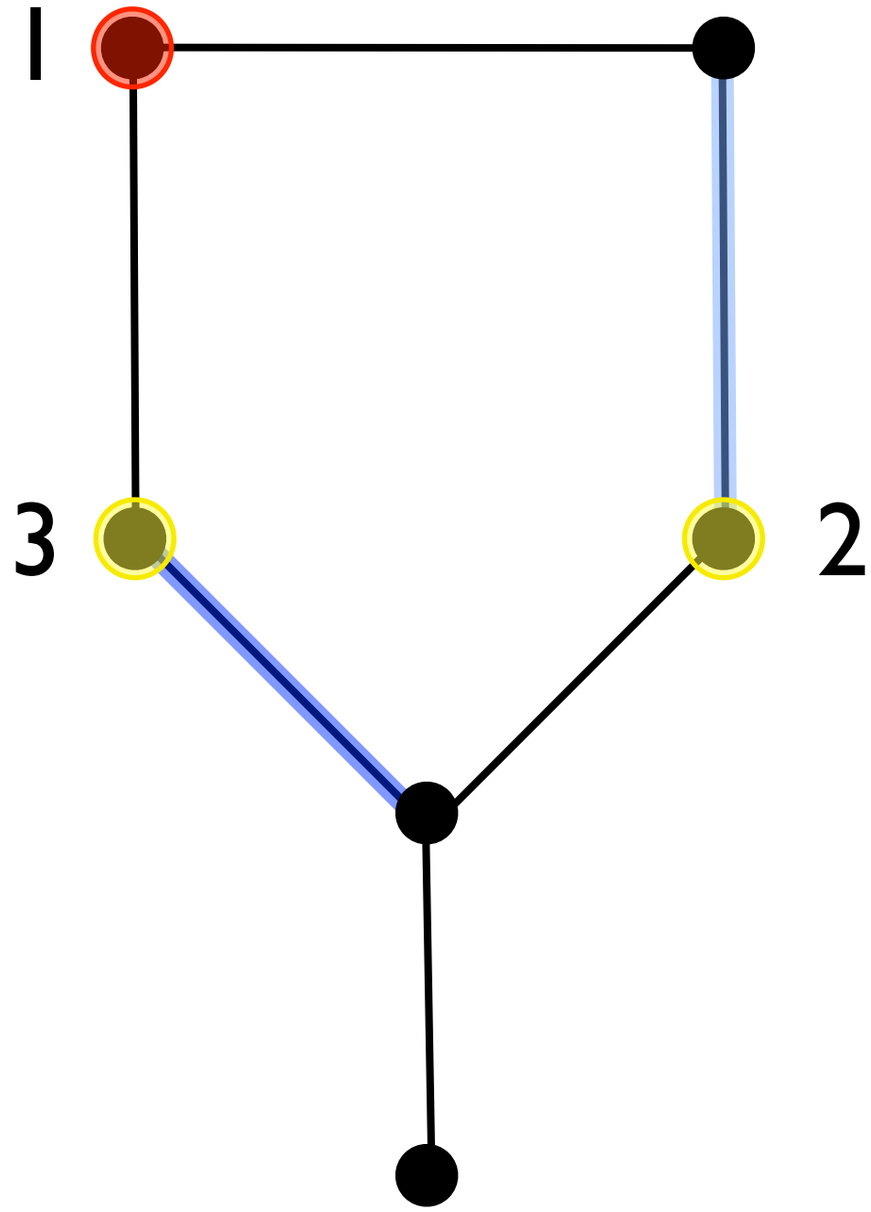
Maximum Matchings

but...



Maximum Matchings

but...



Maximum Matchings
general unweighted case

- We need to take care of odd cycles (*blossoms*) by compressing them

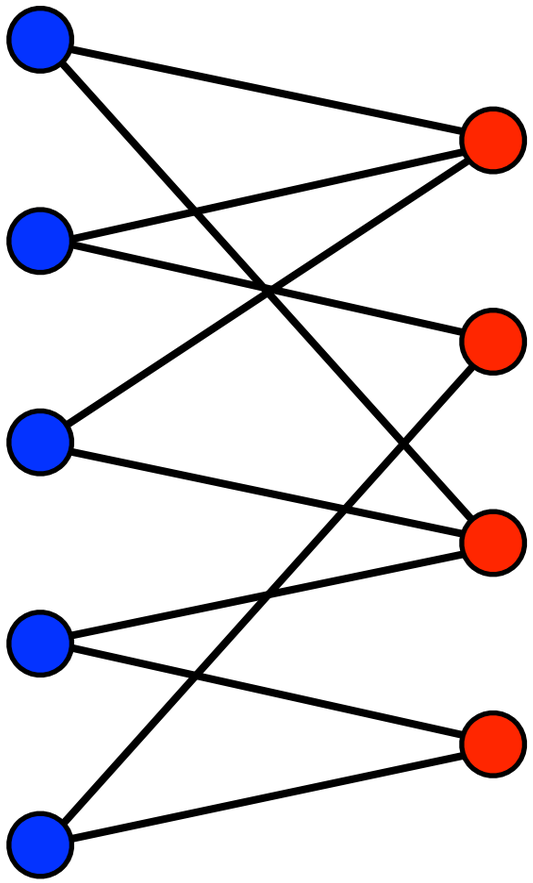
Maximum Matchings

general unweighted case

- We need to take care of odd cycles (*blossoms*) by compressing them
- Edmonds' algorithm does that efficiently

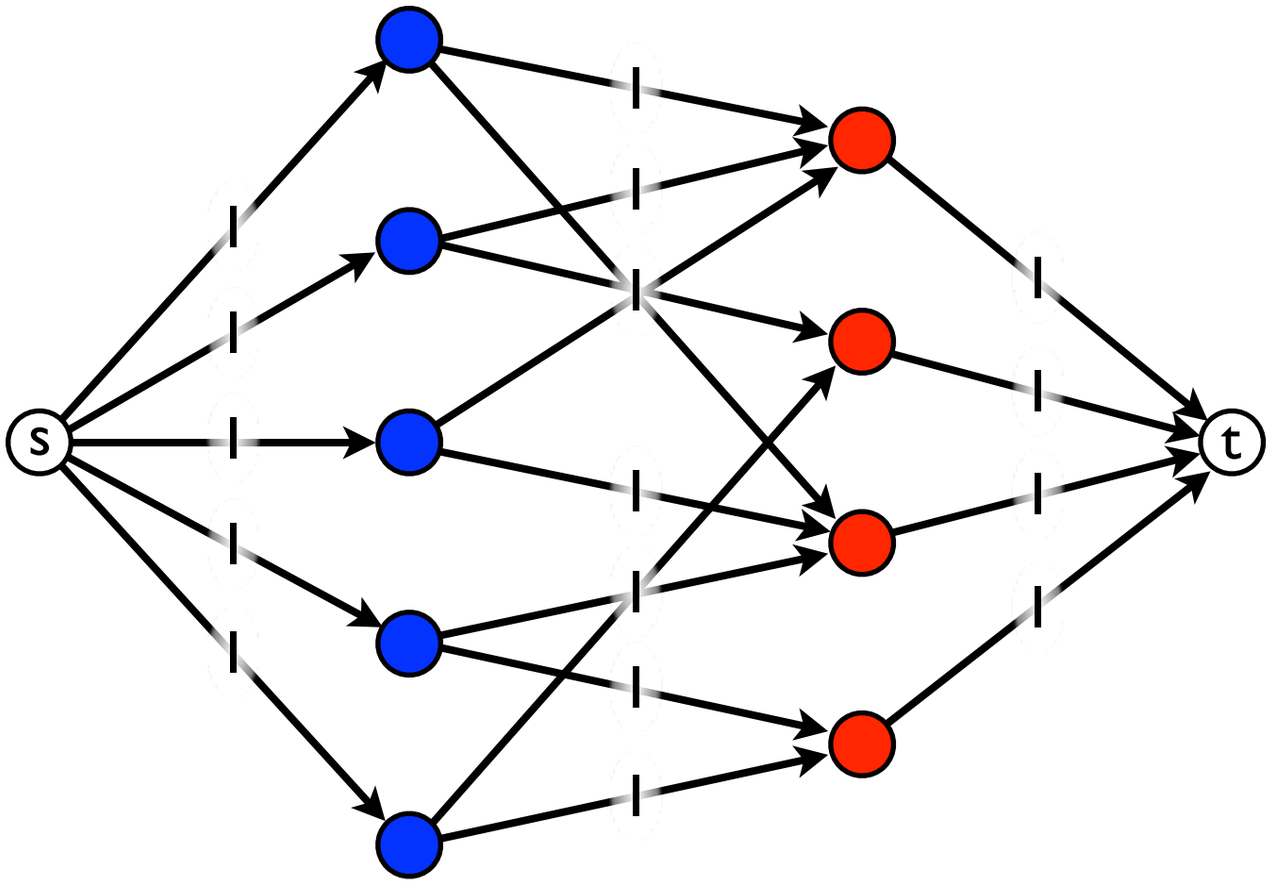
Maximum Matchings

unweighted bipartite case



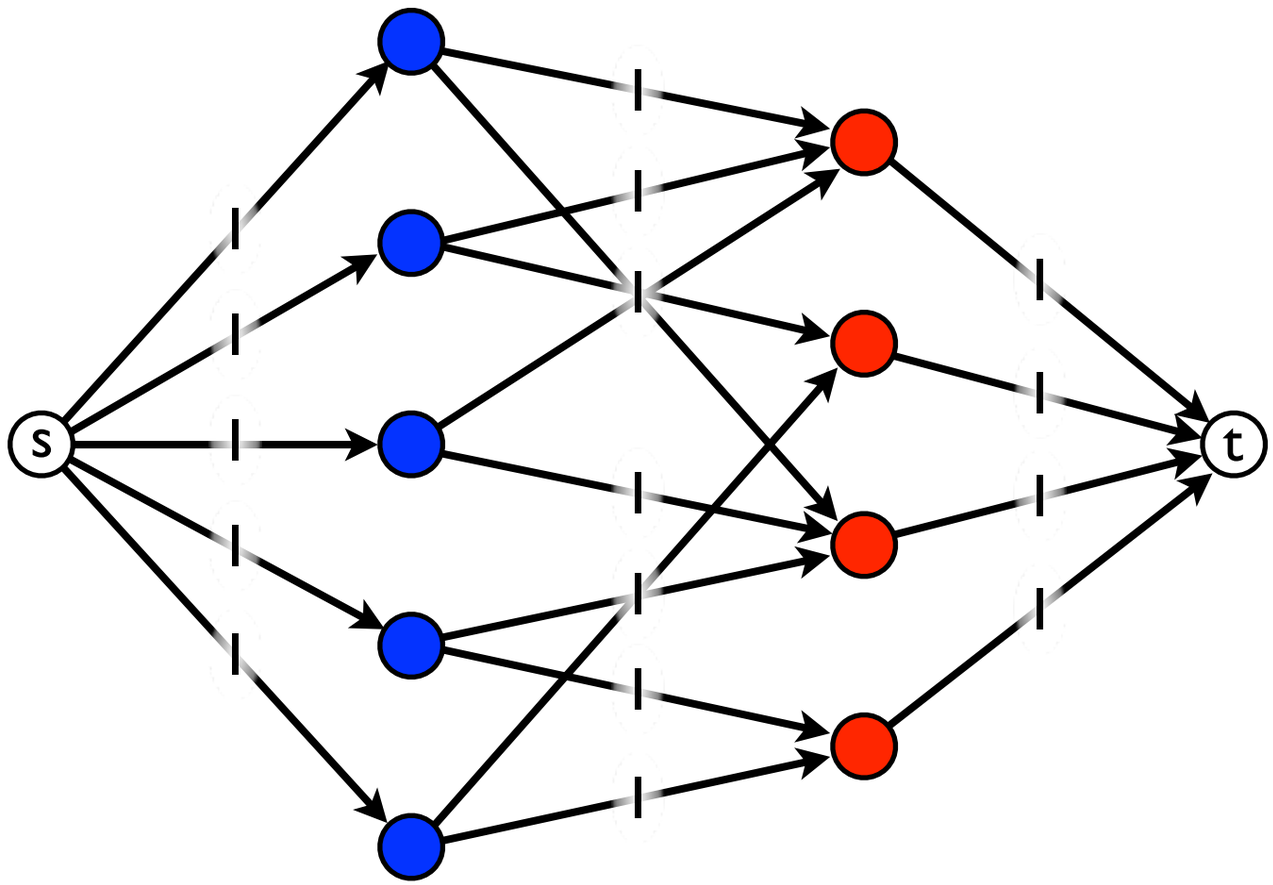
Maximum Matchings

unweighted bipartite case



Maximum Matchings

unweighted bipartite case



⇒ maximum flow = cardinality of maximum matching

Maximum Matchings

unweighted bipartite case

method	graph type	runtime
flow	bipartite unweighted	$O(VE)$
BGL Edmonds	any unweighted	$O(VE \cdot \alpha(V, E))$
Edmonds	any	$O(\sqrt{V}E)$
Mucha, Sankowski	any	$O(V^{2.376})$

Matchings in BGL

Matchings in BGL

invoking algorithm

Matchings in BGL

invoking algorithm

```
#include <boost/graph/max_cardinality_matching.hpp>
```

Matchings in BGL

invoking algorithm

```
#include <boost/graph/max_cardinality_matching.hpp>
...
```

Matchings in BGL

invoking algorithm

```
#include <boost/graph/max_cardinality_matching.hpp>
...
vector<VertexDescriptor> mate(NVERTICES);
edmonds_maximum_cardinality_matching(g, &mate[0]);
```

Matchings in BGL

invoking algorithm

```
#include <boost/graph/max_cardinality_matching.hpp>
...
vector<VertexDescriptor> mate(NVERTICES);
edmonds_maximum_cardinality_matching(g, &mate[0]);
```

Matchings in BGL

invoking algorithm

```
#include <boost/graph/max_cardinality_matching.hpp>
...
vector<VertexDescriptor> mate(NVERTICES);
edmonds_maximum_cardinality_matching(g, &mate[0]);

const VertexDescriptor NULL_VERTEX = graph_traits<Graph>::null_vertex();
```

Matchings in BGL

invoking algorithm

```
#include <boost/graph/max_cardinality_matching.hpp>
...
vector<VertexDescriptor> mate(NVERTICES);
edmonds_maximum_cardinality_matching(g, &mate[0]);


const VertexDescriptor NULL_VERTEX = graph_traits<Graph>::null_vertex();
...
```

Matchings in BGL

invoking algorithm

```
#include <boost/graph/max_cardinality_matching.hpp>
...
vector<VertexDescriptor> mate(NVERTICES);
edmonds_maximum_cardinality_matching(g, &mate[0]);


const VertexDescriptor NULL_VERTEX = graph_traits<Graph>::null_vertex();
...
for(int i = 0; i < NVERTICES; ++i)
```

Matchings in BGL

invoking algorithm

```
#include <boost/graph/max_cardinality_matching.hpp>
...
vector<VertexDescriptor> mate(NVERTICES);
edmonds_maximum_cardinality_matching(g, &mate[0]);


const VertexDescriptor NULL_VERTEX = graph_traits<Graph>::null_vertex();
...
for(int i = 0; i < NVERTICES; ++i)
  if(mate[i] != NULL_VERTEX && i < mate[i])
```

Matchings in BGL

invoking algorithm

```
#include <boost/graph/max_cardinality_matching.hpp>
...
vector<VertexDescriptor> mate(NVERTICES);
edmonds_maximum_cardinality_matching(g, &mate[0]);


const VertexDescriptor NULL_VERTEX = graph_traits<Graph>::null_vertex();
...
for(int i = 0; i < NVERTICES; ++i)
  if(mate[i] != NULL_VERTEX && i < mate[i])
    cout << i << " -- " << mate[i] << endl;
```