NP-Completeness: The Equivalence of the Decision and Optimization Independent Set Problems

Coloring Problem: Optimization Version

Let $G = (V, E)$ be an undirected graph. An independent set of G is a subset of the vertices such that so that no two vertices in the subset are neighbors (in G).

Coloring Problem: Optimization Version

Let $G = (V, E)$ be an undirected graph. An independent set of G is a subset of the vertices such that so that no two vertices in the subset are neighbors (in G).

Example

Coloring Problem: Optimization Version

Coloring Problem: Optimization Version

Independent set problem: Given an undirected graph $G = (V, E)$, find a maximum independent set.

Coloring Problem: Optimization Version

Independent set problem: Given an undirected graph $G = (V, E)$, find a maximum independent set.

Is the independent set problem NP-complete?

Coloring Problem: Optimization Version

Independent set problem: Given an undirected graph $G = (V, E)$, find a maximum independent set.

Is the independent set problem NP-complete?

Technically it cannot be NP-complete since it is not a decision problem.

Coloring Problem: Optimization Version

Independent set problem: Given an undirected graph $G = (V, E)$, find a maximum independent set.

Is the independent set problem NP-complete?

Technically it cannot be NP-complete since it is not a decision problem.

What do we need to do?

Coloring Problem: Optimization Version

Independent set problem: Given an undirected graph $G = (V, E)$, find a maximum independent set.

Is the independent set problem NP-complete?

Technically it cannot be NP-complete since it is not a decision problem.

What do we need to do?

Find an analogous decision problem, and prove that the analogous decision problem is NP-complete.

Decision Version

Decision Version

Decision version of the independent set problem:

Decision Version

Decision version of the independent set problem:
Given an undirected graph $G = (V, E)$ and an integer k, does G have an independent of size k (or larger)?

Decision Version

Decision version of the independent set problem:
Given an undirected graph $G = (V, E)$ and an integer k, does G have an independent of size k (or larger)?

Is the decision version analogous?

Decision Version

Decision version of the independent set problem:
Given an undirected graph $G = (V, E)$ and an integer k, does G have an independent of size k (or larger)?

Is the decision version analogous?

Two requirements:

Decision Version

Decision version of the independent set problem:
Given an undirected graph $G = (V, E)$ and an integer k, does G have an independent of size k (or larger)?

Is the decision version analogous?

Two requirements:

- The decision problem should feel like the optimization problem.

Decision Version

Decision version of the independent set problem:
Given an undirected graph $G = (V, E)$ and an integer k, does G have an independent of size k (or larger)?

Is the decision version analogous?

Two requirements:

- The decision problem should feel like the optimization problem. Yes.

Decision Version

Decision version of the independent set problem:
Given an undirected graph $G = (V, E)$ and an integer k, does G have an independent of size k (or larger)?

Is the decision version analogous?

Two requirements:

- The decision problem should feel like the optimization problem. Yes.
- The decision and optimization problems should be equivalent up to polynomial time.

Decision Version

Decision version of the independent set problem:
Given an undirected graph $G = (V, E)$ and an integer k, does G have an independent of size k (or larger)?

Is the decision version analogous?

Two requirements:

- The decision problem should feel like the optimization problem. Yes.
- The decision and optimization problems should be equivalent up to polynomial time. Will show.

Decision Version

Decision Version

Is the decison version NP-complete?

Decision Version

Is the decison version NP-complete? Yes.

Decision Version

Is the decison version NP-complete? Yes.

How do we know?

Decision Version

Is the decison version NP-complete? Yes.

How do we know?

Karp proved it in 1972.

Proof of equivalence

Proof of equivalence

Theorem

The optimization version of independent set is in $\boldsymbol{P}$ *if and only if the decision version of independent set is in* $\boldsymbol{P}$.

Proof of equivalence

Theorem

The optimization version of independent set is in $\boldsymbol{P}$ *if and only if the decision version of independent set is in* $\boldsymbol{P}$.

Proof.

$(\implies)$ Assume that the optimization version of independent set is in $\boldsymbol{P}$ with time $O(n^r)$.

Proof of equivalence

Theorem

The optimization version of independent set is in $\boldsymbol{P}$ *if and only if the decision version of independent set is in* $\boldsymbol{P}$.

Proof.

$(\implies)$ Assume that the optimization version of independent set is in $\boldsymbol{P}$ with time $O(n^r)$. Given decision problem for undirected graph $G = (V, E)$ with integer k:

Proof of equivalence

Theorem

*The optimization version of independent set is in **P** if and only if the decision version of independent set is in **P**.*

Proof.

($\Longrightarrow$) Assume that the optimization version of independent set is in **P** with time $O(n^r)$. Given decision problem for undirected graph $G = (V, E)$ with integer k:

(1) Run the the optimization version on G.

Proof of equivalence

Theorem

The optimization version of independent set is in $\boldsymbol{P}$ *if and only if the decision version of independent set is in* $\boldsymbol{P}$.

Proof.

$(\implies)$ Assume that the optimization version of independent set is in $\boldsymbol{P}$ with time $O(n^r)$. Given decision problem for undirected graph $G = (V, E)$ with integer k:

(1) Run the the optimization version on G.

(2) Count the number of vertices.

Proof of equivalence

Theorem

The optimization version of independent set is in $\boldsymbol{P}$ *if and only if the decision version of independent set is in* $\boldsymbol{P}$*.*

Proof.

$(\implies)$ Assume that the optimization version of independent set is in $\boldsymbol{P}$ with time $O(n^r)$. Given decision problem for undirected graph $G = (V, E)$ with integer k:

(1) Run the the optimization version on G.

(2) Count the number of vertices.

(3) Compare to k.

Proof of equivalence

Theorem

The optimization version of independent set is in $\boldsymbol{P}$ *if and only if the decision version of independent set is in* $\boldsymbol{P}$*.*

Proof.

$(\implies)$ Assume that the optimization version of independent set is in $\boldsymbol{P}$ with time $O(n^r)$. Given decision problem for undirected graph $G = (V, E)$ with integer k:

(1) Run the the optimization version on G.

(2) Count the number of vertices.

(3) Compare to k.

Time $O(n^r) + O(n) = O(n^r)$. □

Proof of equivalence

Proof of equivalence

Proof.

($\Longleftarrow$) Assume that the decision version of independent set is in $\boldsymbol{P}$ with time $O(n^s)$, using method

`IS_Decision(G,k)`

Proof of equivalence

Proof.

($\Longleftarrow$) Assume that the decision version of independent set is in $\boldsymbol{P}$ with time $O(n^s)$, using method

```
IS_Decision(G,k)
```

Given optimization problem for undirected graph $G = (V, E)$:

Proof of equivalence

Proof.

($\Longleftarrow$) Assume that the decision version of independent set is in $\boldsymbol{P}$ with time $O(n^s)$, using method

`IS_Decision(G,k)`

Given optimization problem for undirected graph $G = (V, E)$:

Some magic.

□

Algorithm

Algorithm

```
/* Find optimal number of vertices */
k ← n
while not IS_Decision(G,k) do
    k ← k-1
end while
```

Algorithm continued

Algorithm continued

```
        /* Find the vertices */
Independent_Set ← ∅
for i = 1 to n do
    G' ← G
    delete vertex i and its neighbors from G'
    if IS_Decision(G',k-1) then
        Independent_Set ← Independent_Set ∪ {i}
        k ← k-1
        G ← G'
    end if
end for
```

Analysis

Analysis

$$T(n) \;=\; O(nn^s)$$

Analysis

$$
\begin{aligned}
T(n) &= O(nn^s) \\
&= O(n^{s+1})
\end{aligned}
$$

NP-Completeness: The Equivalence of the Decision and Optimization Coloring Problems

Coloring Problem: Optimization Version

Coloring problem: Given an undirected graph $G = (V, E)$, color the vertices of G with as few colors as possible so that no two neighboring vertices have the same color.

Coloring Problem: Optimization Version

Coloring problem: Given an undirected graph $G = (V, E)$, color the vertices of G with as few colors as possible so that no two neighboring vertices have the same color.

Is the coloring problem NP-complete?

Coloring Problem: Optimization Version

Coloring problem: Given an undirected graph $G = (V, E)$, color the vertices of G with as few colors as possible so that no two neighboring vertices have the same color.

Is the coloring problem NP-complete?

Technically it cannot be NP-complete since it is not a decision problem.

Coloring Problem: Optimization Version

Coloring problem: Given an undirected graph $G = (V, E)$, color the vertices of G with as few colors as possible so that no two neighboring vertices have the same color.

Is the coloring problem NP-complete?

Technically it cannot be NP-complete since it is not a decision problem.

What do we need to do?

Coloring Problem: Optimization Version

Coloring problem: Given an undirected graph $G = (V, E)$, color the vertices of G with as few colors as possible so that no two neighboring vertices have the same color.

Is the coloring problem NP-complete?

Technically it cannot be NP-complete since it is not a decision problem.

What do we need to do?

Find an analogous decision problem, and prove that the analogous decision problem is NP-complete.

Decision Version

Decision Version

Decision version of coloring problem:

Given an undirected graph $G = (V, E)$ and an integer k, can the vertices of G be colored with (at most) k colors so that no two neighboring vertices have the same color?

Decision Version

Decision version of coloring problem:

Given an undirected graph $G = (V, E)$ and an integer k, can the vertices of G be colored with (at most) k colors so that no two neighboring vertices have the same color?

Is the decision version analogous?

Decision Version

Decision version of coloring problem:
Given an undirected graph $G = (V, E)$ and an integer k, can the vertices of G be colored with (at most) k colors so that no two neighboring vertices have the same color?

Is the decision version analogous?

Two requirements:

Decision Version

Decision version of coloring problem:

Given an undirected graph $G = (V, E)$ and an integer k, can the vertices of G be colored with (at most) k colors so that no two neighboring vertices have the same color?

Is the decision version analogous?

Two requirements:

- The decision problem should feel like the optimization problem.

Decision Version

Decision version of coloring problem:
Given an undirected graph $G = (V, E)$ and an integer k, can the vertices of G be colored with (at most) k colors so that no two neighboring vertices have the same color?

Is the decision version analogous?

Two requirements:

- The decision problem should feel like the optimization problem. Yes.

Decision Version

Decision version of coloring problem:
Given an undirected graph $G = (V, E)$ and an integer k, can the vertices of G be colored with (at most) k colors so that no two neighboring vertices have the same color?

Is the decision version analogous?

Two requirements:

- The decision problem should feel like the optimization problem. Yes.
- The decision and optimization problems should be equivalent up to polynomial time.

Decision Version

Decision version of coloring problem:
Given an undirected graph $G = (V, E)$ and an integer k, can the vertices of G be colored with (at most) k colors so that no two neighboring vertices have the same color?

Is the decision version analogous?

Two requirements:

- The decision problem should feel like the optimization problem. Yes.
- The decision and optimization problems should be equivalent up to polynomial time. Will show.

Decision Version

Decision Version

Is the decison version NP-complete?

Decision Version

Is the decison version NP-complete? Yes.

Decision Version

Is the decison version NP-complete? Yes.

How do we know?

Decision Version

Is the decison version NP-complete? Yes.

How do we know?

Karp proved it in 1972.

Proof of equivalence

Proof of equivalence

Theorem

The optimization version of coloring is in $\boldsymbol{P}$ *if and only if the decision version of coloring is in* $\boldsymbol{P}$.

Proof of equivalence

Theorem

The optimization version of coloring is in $\boldsymbol{P}$ *if and only if the decision version of coloring is in* $\boldsymbol{P}$.

Proof.

$(\implies)$ Assume that the optimization version of coloring is in $\boldsymbol{P}$ with time $O(n^r)$.

Proof of equivalence

Theorem

The optimization version of coloring is in $\boldsymbol{P}$ *if and only if the decision version of coloring is in* $\boldsymbol{P}$*.*

Proof.

$(\implies)$ Assume that the optimization version of coloring is in $\boldsymbol{P}$ with time $O(n^r)$. Given decision problem for undirected graph $G = (V, E)$ with integer k:

Proof of equivalence

Theorem

*The optimization version of coloring is in **P** if and only if the decision version of coloring is in **P**.*

Proof.

($\Longrightarrow$) Assume that the optimization version of coloring is in ***P*** with time $O(n^r)$. Given decision problem for undirected graph $G = (V, E)$ with integer k:

(1) Run the the optimization version on G.

Proof of equivalence

Theorem

*The optimization version of coloring is in **P** if and only if the decision version of coloring is in **P**.*

Proof.

($\Longrightarrow$) Assume that the optimization version of coloring is in **P** with time $O(n^r)$. Given decision problem for undirected graph $G = (V, E)$ with integer k:

(1) Run the the optimization version on G.

(2) Count the number of colors.

Proof of equivalence

Theorem

The optimization version of coloring is in ***P*** *if and only if the decision version of coloring is in* ***P****.*

Proof.

($\implies$) Assume that the optimization version of coloring is in ***P*** with time $O(n^r)$. Given decision problem for undirected graph $G = (V, E)$ with integer k:

(1) Run the the optimization version on G.

(2) Count the number of colors.

(3) Compare to k.

Proof of equivalence

Theorem

The optimization version of coloring is in $\boldsymbol{P}$ *if and only if the decision version of coloring is in* $\boldsymbol{P}$.

Proof.

($\Longrightarrow$) Assume that the optimization version of coloring is in $\boldsymbol{P}$ with time $O(n^r)$. Given decision problem for undirected graph $G = (V, E)$ with integer k:

(1) Run the the optimization version on G.

(2) Count the number of colors.

(3) Compare to k.

Time $O(n^r) + O(n) \; = \; O(n^r)$. □

The other way around.

Idea for algorithm

Assume that given $G = (V, E)$,
`partial_colorable(G,k,p)` decides if the partial coloring p of G can be extended to a k-coloring of G in time $O(n^s)$.

Idea for algorithm

Idea for algorithm

```
/* Find optimal number of colors */
p ← empty coloring
k ← 1
while not partial_colorable(G,k,p) do
    k ← k+1
end while
```

Idea for algorithm

```
/* Find optimal number of colors */
p ← empty coloring
k ← 1
while not partial_colorable(G,k,p) do
    k ← k+1
end while
/* Color the vertices */
for i = 1 to n do
    j ← 0
    repeat
        j ← j+1
        color vertex i with color j in p
    until partial_colorable(G,k,p)
end for
```

Idea for algorithm

```
/* Find optimal number of colors */
p ← empty coloring
k ← 1
while not partial_colorable(G,k,p) do
    k ← k+1
end while
/* Color the vertices */
for i = 1 to n do
    j ← 0
    repeat
        j ← j+1
        color vertex i with color j in p
    until partial_colorable(G,k,p)
end for
```

Proof of equivalence

Proof of equivalence

`partial_colorable is not the decision version!`

Proof of equivalence

Proof of equivalence

Proof.

$(\Longleftarrow)$ Assume that the decision version of coloring is in $\boldsymbol{P}$ with time $O(n^s)$, using method

```
colorable(G,k)
```

Proof of equivalence

Proof.

($\Longleftarrow$) Assume that the decision version of coloring is in $\boldsymbol{P}$ with time $O(n^s)$, using method

```
colorable(G,k)
```

Given optimization problem for undirected graph $G = (V, E)$:

Proof of equivalence

Proof.

$(\Longleftarrow)$ Assume that the decision version of coloring is in $\boldsymbol{P}$ with time $O(n^s)$, using method

```
colorable(G,k)
```

Given optimization problem for undirected graph $G = (V, E)$:

Some magic.

□

Proof of equivalence

Proof of equivalence

Donald Rumsfeld (Secretary of Defense) said:

> *... you go to war with the army you have, not the army you might want or wish to have ...*

Proof of equivalence

Donald Rumsfeld (Secretary of Defense) said:

> *... you go to war with the army you have, not the army you might want or wish to have ...*

With apologies to Rumsfeld:

> *You program with the library routine you have, not the library routine you wish you had.*

Proof of equivalence

Donald Rumsfeld (Secretary of Defense) said:

> *... you go to war with the army you have, not the army you might want or wish to have ...*

With apologies to Rumsfeld:

> *You program with the library routine you have, not the library routine you wish you had.*

Another great Rumsfeld quote:

> *There are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns ...*

Proof of equivalence

Proof of equivalence

Find optimal number of colors, k

Proof of equivalence

Find optimal number of colors, k

Create a k-clique (complete graph of size k)

Proof of equivalence

Find optimal number of colors, k

Create a k-clique (complete graph of size k)

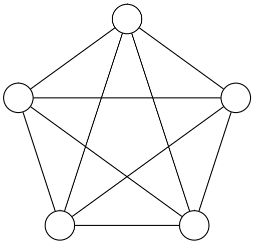
Example: $k = 5$

Proof of equivalence

Find optimal number of colors, k

Create a k-clique (complete graph of size k)

Example: $k = 5$

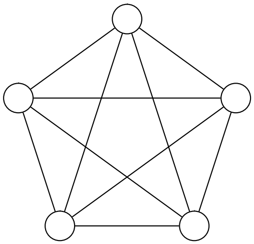


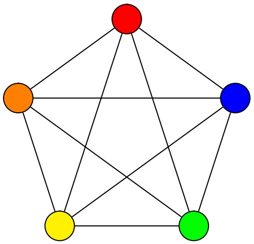
Proof of equivalence

Find optimal number of colors, k

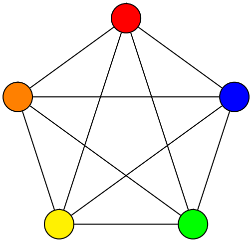
Create a k-clique (complete graph of size k)

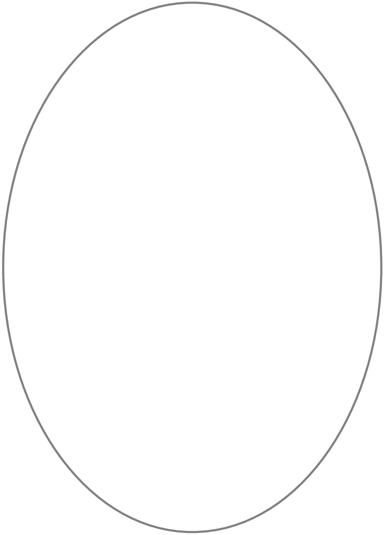
Example: $k = 5$



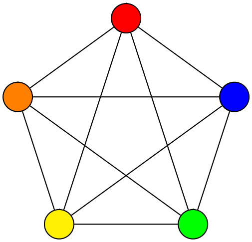


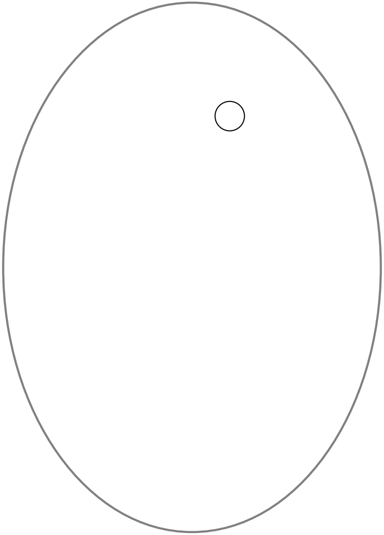
Proof of equivalence, $k = 5$



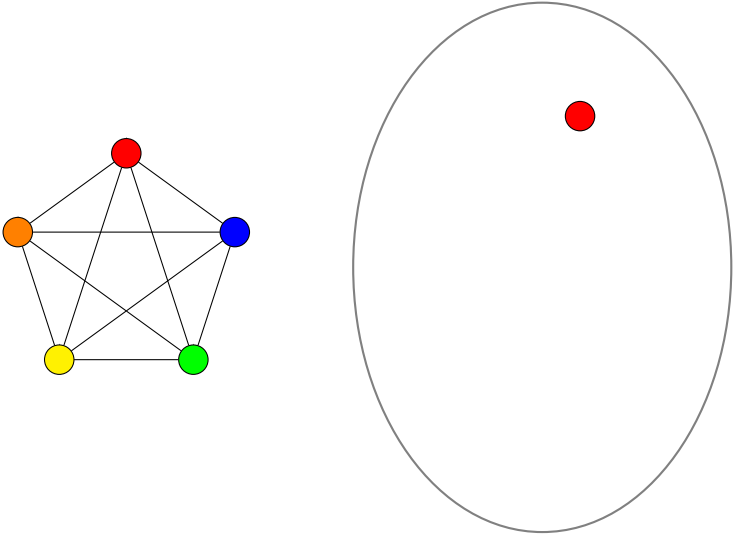


Proof of equivalence, $k = 5$

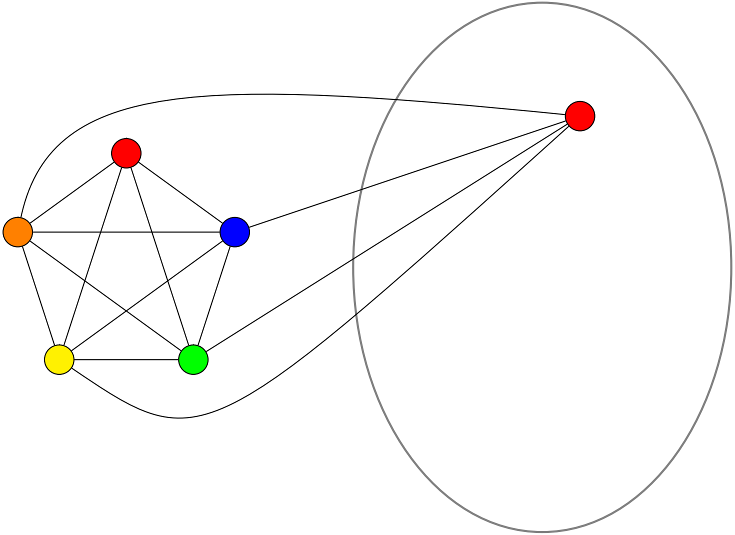




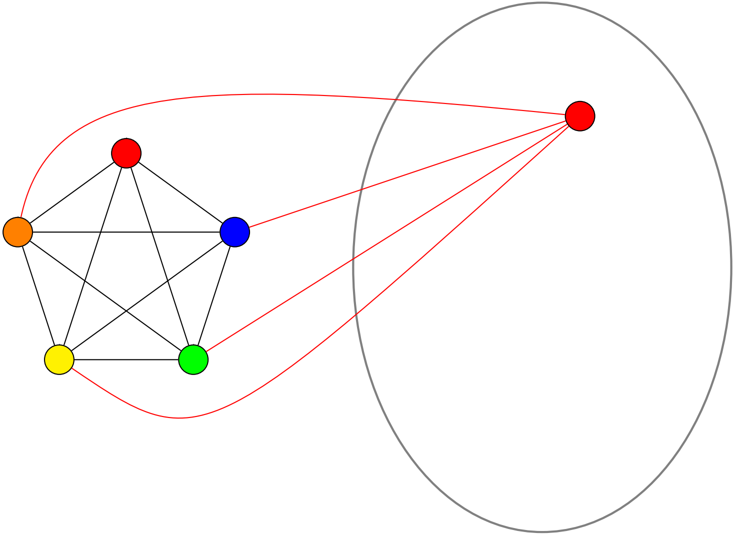
Proof of equivalence, $k = 5$



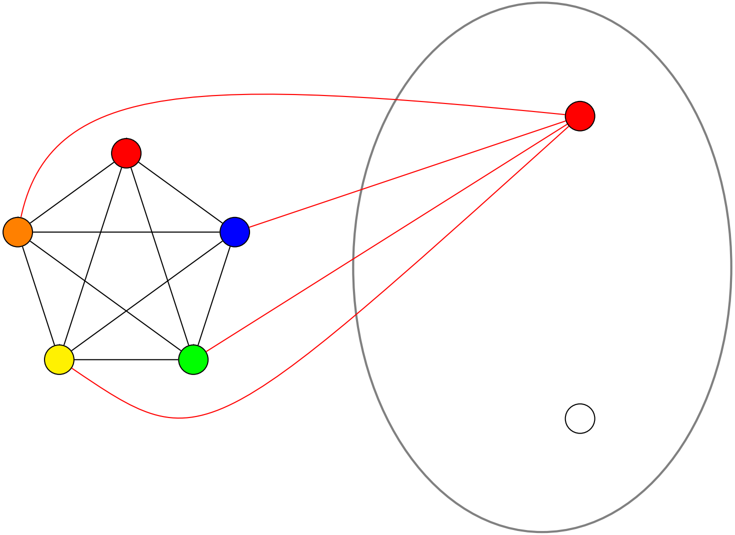
Proof of equivalence, $k = 5$



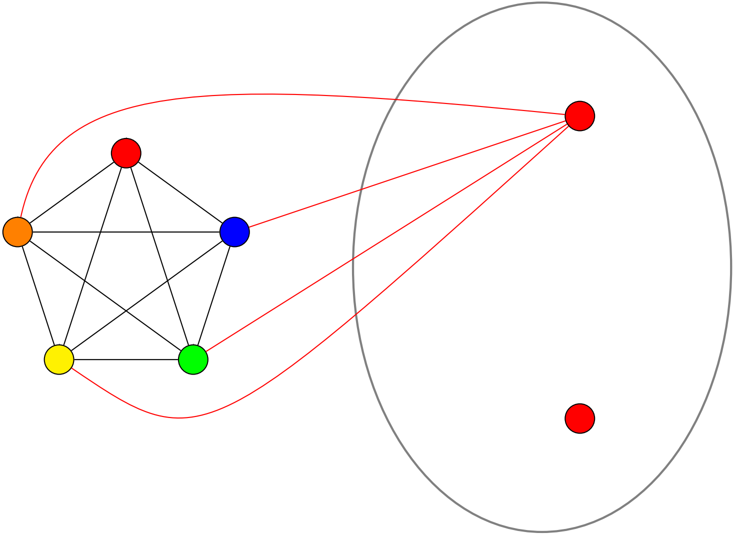
Proof of equivalence, $k = 5$



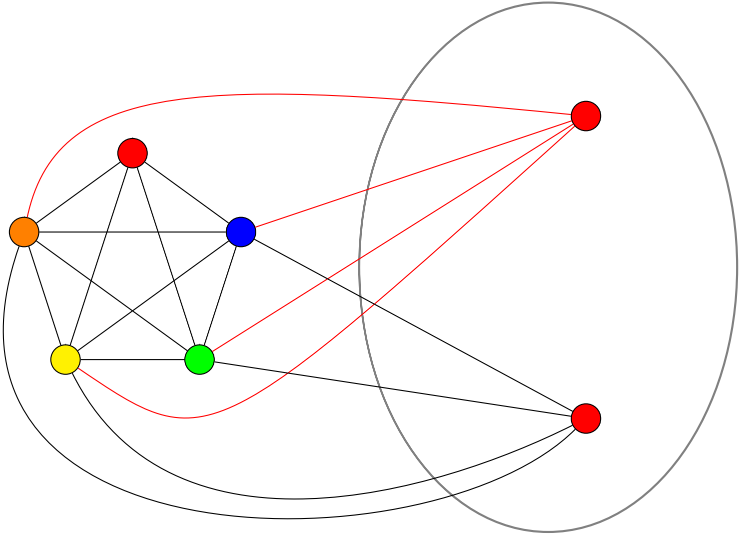
Proof of equivalence, $k = 5$



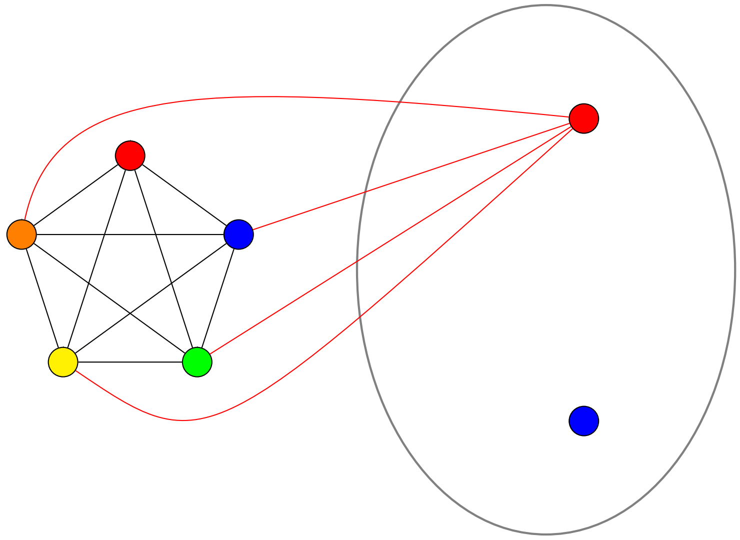
Proof of equivalence, $k = 5$



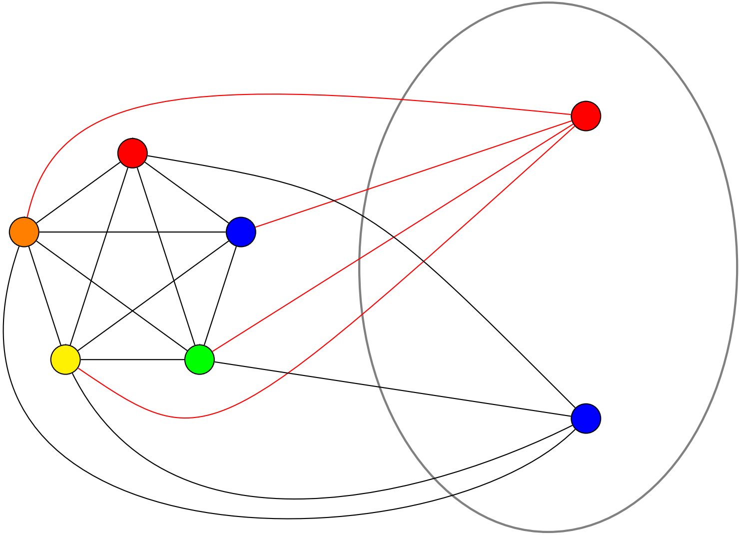
Proof of equivalence, $k = 5$



Proof of equivalence, $k = 5$



Proof of equivalence, $k = 5$



Algorithm

Algorithm

```
/* Find optimal number of colors */
```

Algorithm

```
/* Find optimal number of colors */
k ← 1
while not colorable(G,k) do
    k ← k+1
end while
```

Algorithm

```
/* Find optimal number of colors */
k ← 1
while not colorable(G,k) do
    k ← k+1
end while
```

$O(kn^s)$

Algorithm

```
/* Find optimal number of colors */
k ← 1
while not colorable(G,k) do
    k ← k+1
end while

/* Create a k-clique */
```

$O(kn^s)$

Algorithm

```
/* Find optimal number of colors */    O(kn^s)
k ← 1
while not colorable(G,k) do
    k ← k+1
end while

/* Create a k-clique */
Create a k-clique
```

Algorithm

```
/* Find optimal number of colors */   O(kn^s)
k ← 1
while not colorable(G,k) do
    k ← k+1
end while

/* Create a k-clique */   O(k^2)
Create a k-clique
```

Algorithm (continued)

Algorithm (continued)

```
/* Color the vertices */
```

Algorithm (continued)

```
/* Color the vertices */
for i = 1 to n do
    j ← 0
    repeat
        j ← j+1
        connect vertex i to every vertex
                in clique except vertex j
    until colorable(G,k)
    color[i] ← j
end for
```

Algorithm (continued)

```
/* Color the vertices */
for i = 1 to n do
    j ← 0
    repeat
        j ← j+1
        connect vertex i to every vertex
                in clique except vertex j
    until colorable(G,k)
    color[i] ← j
end for
```

Algorithm (continued)

```
/* Color the vertices */
for i = 1 to n do
    j ← 0
    repeat
        j ← j+1
        connect vertex i to every vertex
                in clique except vertex j
    until colorable(G,k)
    color[i] ← j
end for
```

$O(nk(k + (n + k)^s))$

Analysis

Analysis

$$T(n) \; = \; O(kn^s) \; + \; O(k^2) \; + \; O(nk(k+(n+k)^s))$$

Analysis

$$\begin{aligned} T(n) &= O(kn^s) + O(k^2) + O(nk(k + (n+k)^s)) \\ &= O(nn^s) + O(n^2) + O(nn(n + (n+n)^s)) \end{aligned}$$

Analysis

$$
\begin{aligned}
T(n) &= O(kn^s) + O(k^2) + O(nk(k + (n+k)^s)) \\
&= O(nn^s) + O(n^2) + O(nn(n + (n+n)^s)) \\
&= O(n^{s+1}) + O(n^2) + O(n^3 + n^2(2n)^s))
\end{aligned}
$$

Analysis

$$
\begin{aligned}
T(n) &= O(kn^s) + O(k^2) + O(nk(k + (n+k)^s)) \\
&= O(nn^s) + O(n^2) + O(nn(n + (n+n)^s)) \\
&= O(n^{s+1}) + O(n^2) + O(n^3 + n^2(2n)^s)) \\
&= O(n^{s+1}) + O(n^2) + O(n^3 + n^2 2^s n^s)
\end{aligned}
$$

Analysis

$$
\begin{aligned}
T(n) &= O(kn^s) + O(k^2) + O(nk(k + (n+k)^s)) \\
&= O(nn^s) + O(n^2) + O(nn(n + (n+n)^s)) \\
&= O(n^{s+1}) + O(n^2) + O(n^3 + n^2(2n)^s)) \\
&= O(n^{s+1}) + O(n^2) + O(n^3 + n^2 2^s n^s) \\
&= O(n^{s+1}) + O(n^2) + O(n^3 + n^{s+2})
\end{aligned}
$$

Analysis

$$
\begin{aligned}
T(n) &= O(kn^s) + O(k^2) + O(nk(k + (n+k)^s)) \\
&= O(nn^s) + O(n^2) + O(nn(n + (n+n)^s)) \\
&= O(n^{s+1}) + O(n^2) + O(n^3 + n^2(2n)^s)) \\
&= O(n^{s+1}) + O(n^2) + O(n^3 + n^2 2^s n^s) \\
&= O(n^{s+1}) + O(n^2) + O(n^3 + n^{s+2}) \\
&= O(n^{s+2})
\end{aligned}
$$