Flood-filling Games on Graphs

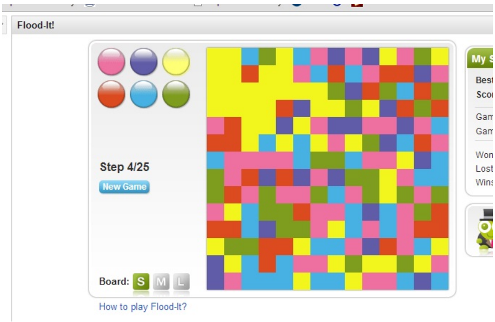
Kitty Meeks

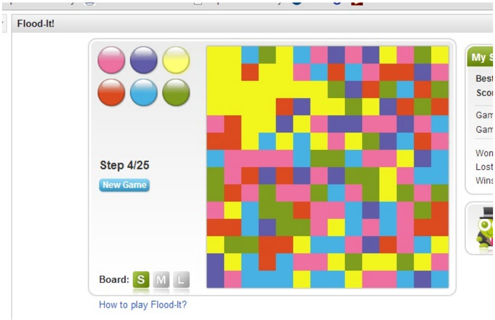
School of Mathematical Sciences
Queen Mary, University of London

Joint work with Alex Scott (University of Oxford)

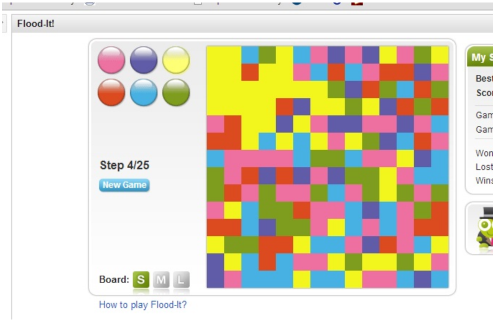
This is officially FUN

- *The complexity of flood filling games*, Arthur, Clifford, Jalsenius, Montanaro & Sach, **Fun with Algorithms 2010**
- *An algorithmic analysis of the honey-bee game*, Fleischer & Woeginger, **Fun with Algorithms 2010**
- *Spanning trees and the complexity of flood filling games*, M. & Scott, **Fun with Algorithms 2012**


Flood-It!
Step 4/25
New Game
Board: S M L
How to play Flood-It?

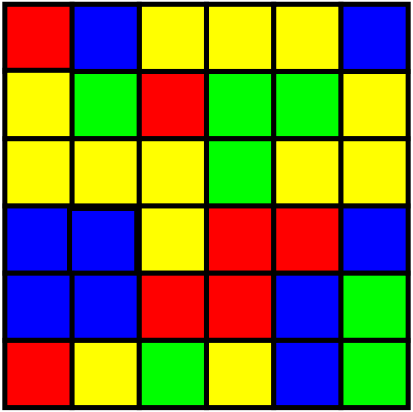


- http://floodit.appspot.com
- Smartphone apps: Flood-It! 2 (iPhone), Flood-It! (Android)

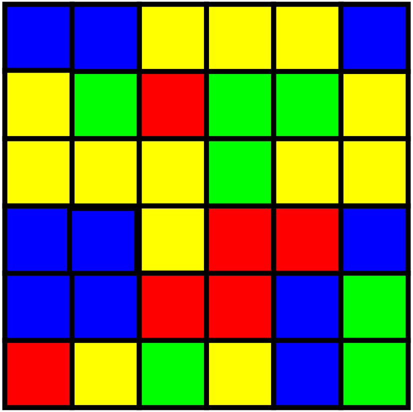


- http://floodit.appspot.com
- Smartphone apps: Flood-It! 2 (iPhone), Flood-It! (Android)
- 2-player version: Honey-bee game

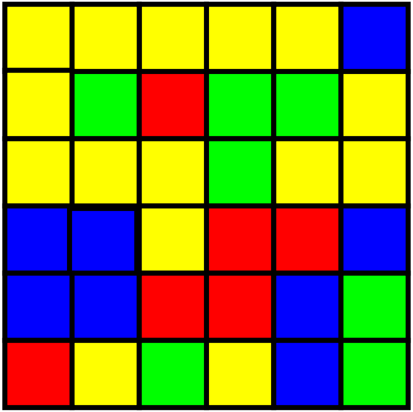
The original Flood-It game



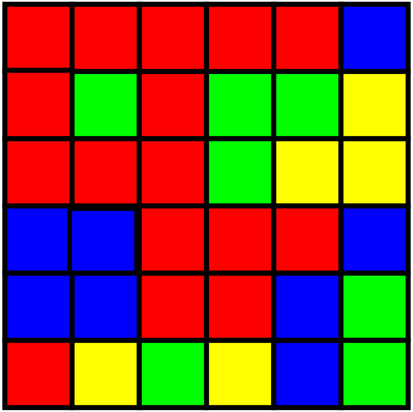
The original Flood-It game



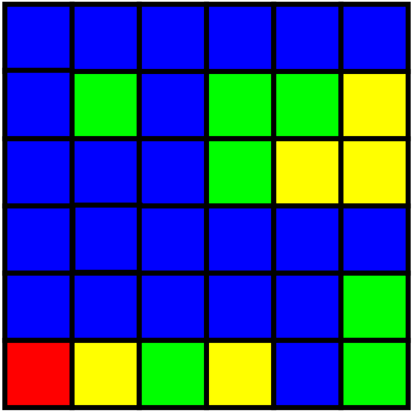
The original Flood-It game



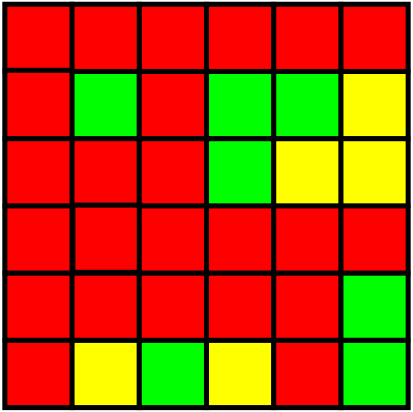
The original Flood-It game



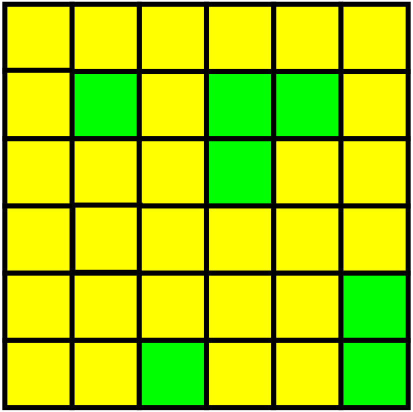
The original Flood-It game



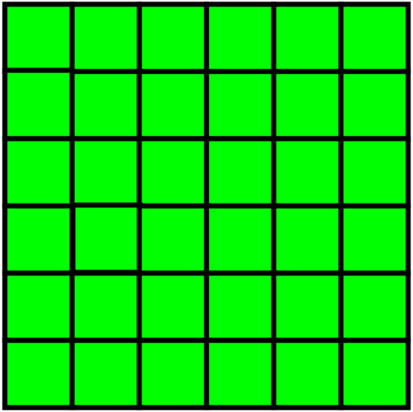
The original Flood-It game



The original Flood-It game



The original Flood-It game



A good strategy?

- Choose the colour that will add the most squares to the flooded area?

A good strategy?

- Choose the colour that will add the most squares to the flooded area?
- **NO!**

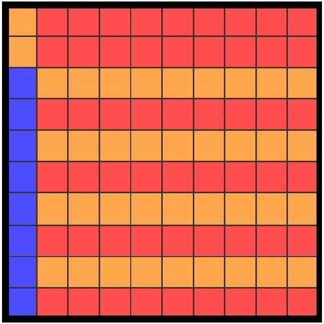


Figure: Example due to Clifford, Jalsenius, Montanaro and Sach

Generalising the game

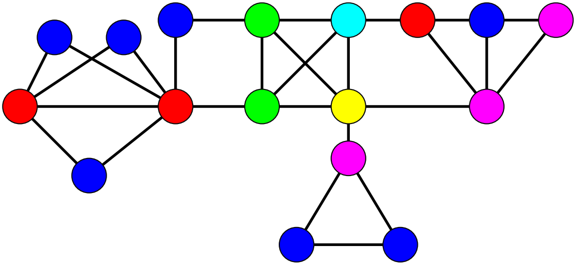
1. Play on arbitrary, coloured (connected) graphs
2. Allow moves to change the colour of **any** monochromatic component

So a move (v, d) involves picking a vertex v and a colour d and giving every vertex in the same monochromatic component as v colour d.

Generalising the game

1. Play on arbitrary, coloured (connected) graphs
2. Allow moves to change the colour of **any** monochromatic component

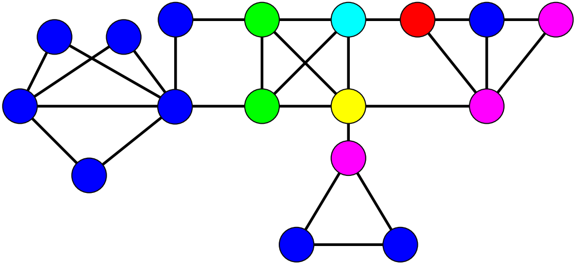
So a move (v, d) involves picking a vertex v and a colour d and giving every vertex in the same monochromatic component as v colour d.



Generalising the game

1. Play on arbitrary, coloured (connected) graphs
2. Allow moves to change the colour of **any** monochromatic component

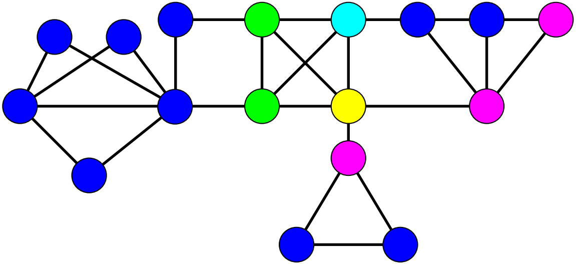
So a move (v, d) involves picking a vertex v and a colour d and giving every vertex in the same monochromatic component as v colour d.



Generalising the game

1. Play on arbitrary, coloured (connected) graphs
2. Allow moves to change the colour of **any** monochromatic component

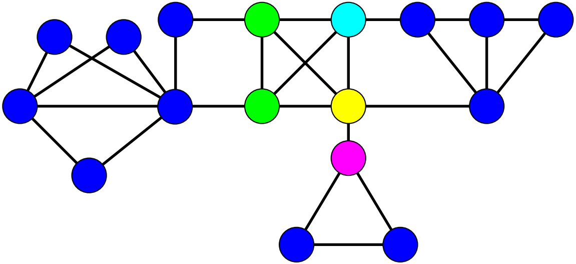
So a move (v, d) involves picking a vertex v and a colour d and giving every vertex in the same monochromatic component as v colour d.



Generalising the game

1. Play on arbitrary, coloured (connected) graphs
2. Allow moves to change the colour of **any** monochromatic component

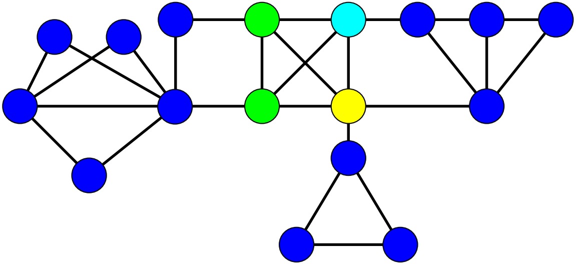
So a move (v, d) involves picking a vertex v and a colour d and giving every vertex in the same monochromatic component as v colour d.



Generalising the game

1. Play on arbitrary, coloured (connected) graphs
2. Allow moves to change the colour of **any** monochromatic component

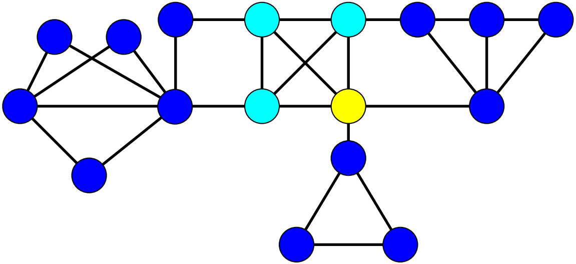
So a move (v, d) involves picking a vertex v and a colour d and giving every vertex in the same monochromatic component as v colour d.



Generalising the game

1. Play on arbitrary, coloured (connected) graphs
2. Allow moves to change the colour of **any** monochromatic component

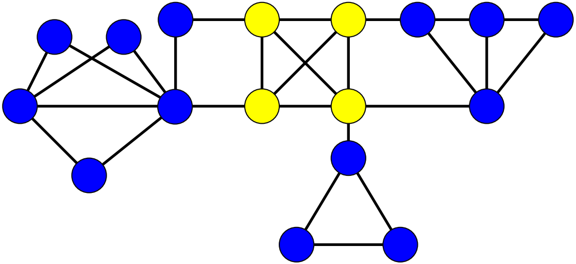
So a move (v, d) involves picking a vertex v and a colour d and giving every vertex in the same monochromatic component as v colour d.



Generalising the game

1. Play on arbitrary, coloured (connected) graphs
2. Allow moves to change the colour of **any** monochromatic component

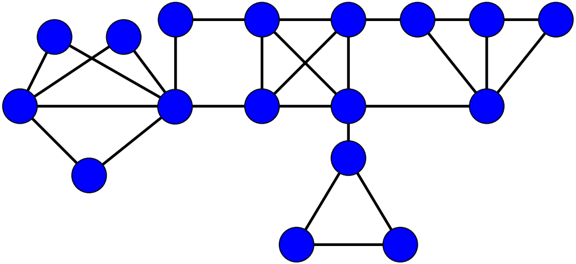
So a move (v, d) involves picking a vertex v and a colour d and giving every vertex in the same monochromatic component as v colour d.



Generalising the game

1. Play on arbitrary, coloured (connected) graphs
2. Allow moves to change the colour of **any** monochromatic component

So a move (v, d) involves picking a vertex v and a colour d and giving every vertex in the same monochromatic component as v colour d.



Problems considered

FREE-FLOOD-IT

Given a coloured connected graph G, what is the minimum number of moves required to flood G (when moves can be played at any vertex)?

Problems considered

FREE-FLOOD-IT
: Given a coloured connected graph G, what is the minimum number of moves required to flood G (when moves can be played at any vertex)?

FIXED-FLOOD-IT
: Given a coloured connected graph G and a vertex $v \in V(G)$, what is the minimum number of moves that must be played at v to flood G?

Problems considered

FREE-FLOOD-IT

Given a coloured connected graph G, what is the minimum number of moves required to flood G (when moves can be played at any vertex)?

FIXED-FLOOD-IT

Given a coloured connected graph G and a vertex $v \in V(G)$, what is the minimum number of moves that must be played at v to flood G?

c-FREE-FLOOD-IT and c-FIXED-FLOOD-IT

The same two questions, except that the initial colouring uses only colours from some fixed set of size c.

Problems considered

FREE-FLOOD-IT

Given a coloured connected graph G, what is the minimum number of moves required to flood G (when moves can be played at any vertex)?

FIXED-FLOOD-IT

Given a coloured connected graph G and a vertex $v \in V(G)$, what is the minimum number of moves that must be played at v to flood G?

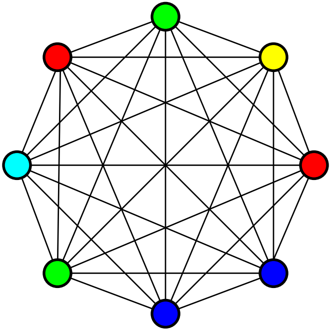
c-FREE-FLOOD-IT and c-FIXED-FLOOD-IT

The same two questions, except that the initial colouring uses only colours from some fixed set of size c.

- If G has n vertices, $n - 1$ moves are always enough (even in the fixed case).
- If there are c colours, at least $c - 1$ moves are needed (even in the free version).

An easy case

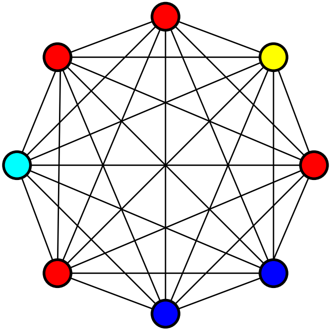
- Cliques



Exactly $c - 1$ moves are required.

An easy case

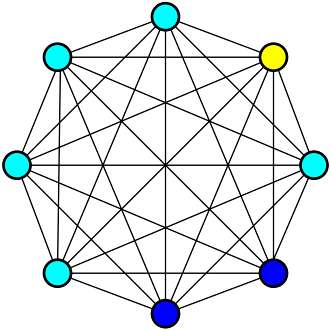
- Cliques



Exactly $c - 1$ moves are required.

An easy case

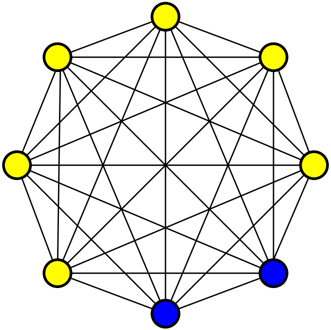
- Cliques



Exactly $c - 1$ moves are required.

An easy case

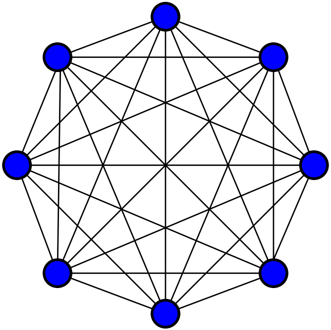
- Cliques



Exactly $c-1$ moves are required.

An easy case

- Cliques



Exactly $c - 1$ moves are required.

Another easy(ish) case...

- How many moves are required for a complete bipartite graph coloured with c colours?

2 colours is easy

(Proved independently by M. and Scott; Clifford, Jalsenius, Montanaro and Sach; Lagoutte, Noual and Thierry.)

2 colours is easy

(Proved independently by M. and Scott; Clifford, Jalsenius, Montanaro and Sach; Lagoutte, Noual and Thierry.)

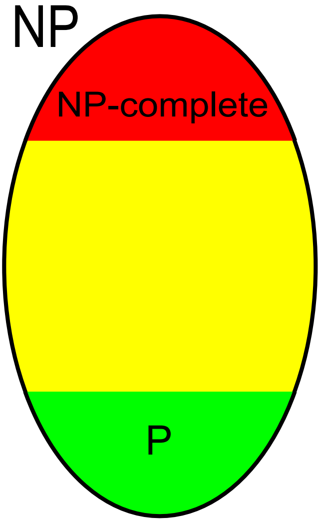
- The fixed version is trivial with two colours.

2 colours is easy

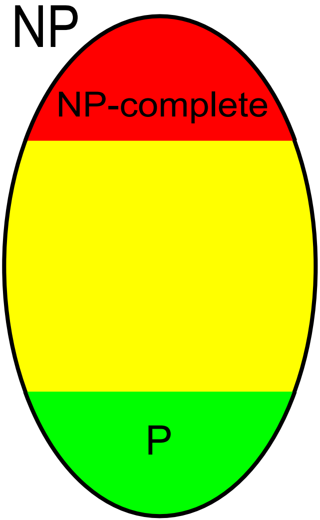
(Proved independently by M. and Scott; Clifford, Jalsenius, Montanaro and Sach; Lagoutte, Noual and Thierry.)

- The fixed version is trivial with two colours.
- In the free version, there is always an optimal strategy that involves playing all moves at the same vertex.
- This means that the minimum number of moves required is equal to the radius of the graph, which is easily computed in polynomial time.

A very brief introduction to computational complexity



A very brief introduction to computational complexity



- NP-complete problems are the “hardest” in NP.
- A polynomial-time algorithm for an NP-complete problem would mean P=NP.
- Showing a problem is NP-complete means that it “almost certainly” cannot be solved in polynomial time.

The original game is hard

Theorem (Arthur, Clifford, Jalsenius, Montanaro, Sach (2010))

*3-*FIXED-FLOOD-IT *and 3-*FREE-FLOOD-IT *are NP-hard on* $n \times n$ *boards.*

The original game is hard

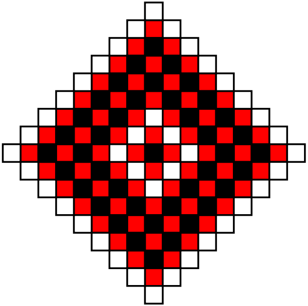
Theorem (Arthur, Clifford, Jalsenius, Montanaro, Sach (2010))

*3-*FIXED-FLOOD-IT *and 3-*FREE-FLOOD-IT *are NP-hard on $n \times n$ boards.*

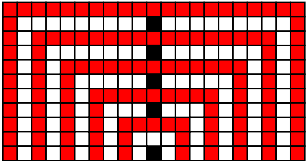
- Reduction from:
  SHORTEST COMMON SUPERSEQUENCE (SCS)
  Input: Strings $s_1, \ldots, s_k$ (of length at most w) over a binary alphabet $\Sigma = \{0, 1\}$, and an integer l.
  Question: Do $s_1, \ldots, s_k$ have a common supersequence of length at most l?
- Shown to be NP-complete by Räihä and Ukkonen.

The original game is hard (fixed version)

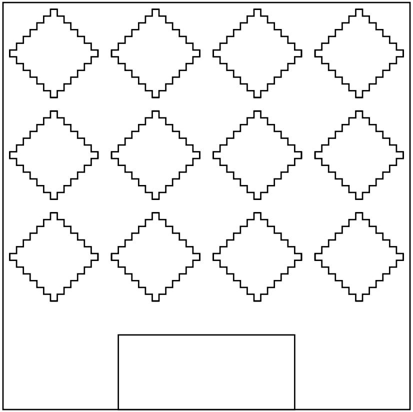
- Representation of the sequence 10010



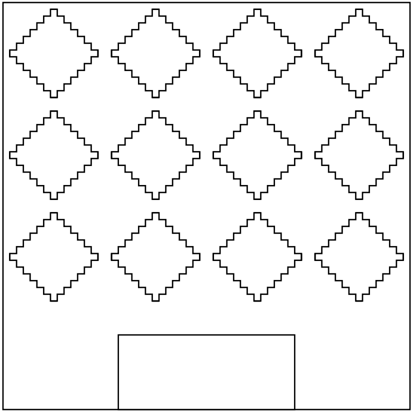
- Gadget to ensure red moves alternate with black or white moves



The original game is hard (fixed version)



The original game is hard (fixed version)



- The proof also implies that there is no constant factor (independent of the number of colours) polynomial-time approximation algorithm.

Rectangular boards of fixed height

	$1 \times n$	$2 \times n$	$3 \times n$	$n \times n$
$c = 2$				
$c = 3$				NP-h
$c = 4$				NP-h
c unbounded				NP-h

Complexity status of c-Free-Flood-It and c-Fixed-Flood-It on rectangular boards.

Rectangular boards of fixed height

	$1 \times n$	$2 \times n$	$3 \times n$	$n \times n$
$c = 2$	P	P	P	P
$c = 3$				NP-h
$c = 4$				NP-h
c unbounded				NP-h

Complexity status of c-Free-Flood-It and c-Fixed-Flood-It on rectangular boards.

Rectangular boards of fixed height

	$1 \times n$	$2 \times n$	$3 \times n$	$n \times n$
$c = 2$	P	P	P	P
$c = 3$	P			NP-h
$c = 4$	P			NP-h
c unbounded	P			NP-h

Complexity status of c-Free-Flood-It and c-Fixed-Flood-It on rectangular boards.

Rectangular boards of fixed height

	$1 \times n$	$2 \times n$	$3 \times n$	$n \times n$
$c = 2$	P	P	P	P
$c = 3$	P		?	NP-h
$c = 4$	P		NP-h	NP-h
c unbounded	P		NP-h	NP-h

Complexity status of c-Free-Flood-It and c-Fixed-Flood-It on rectangular boards.

2xn boards

	c-FIXED-FLOOD-IT	c-FREE-FLOOD-IT
c fixed	P (Clifford et. al., 2012)	P (M.,Scott)
c unbounded	P (Clifford et. al., 2012)	NP-h (M.,Scott)

Hardness for other classes of graphs

- c-Free-Flood-It is NP-hard for trees, if $c \geq 4$ (Fleischer and Woeginger, 2012).

Hardness for other classes of graphs

- c-Free-Flood-It is NP-hard for trees, if $c \geq 4$ (Fleischer and Woeginger, 2012).
- Free-Flood-It is NP-hard for split graphs and proper interval graphs (Fukui, Otachi, Uehara, Uno and Uno, 2013).

Spanning trees

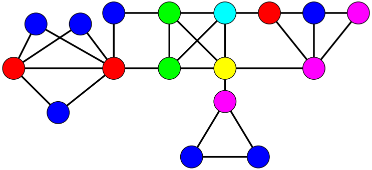
Theorem

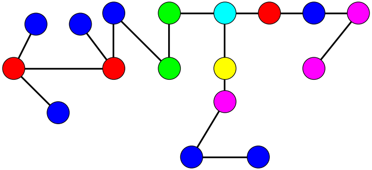
The number of moves required to flood a coloured graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T.

Spanning trees

Theorem

The number of moves required to flood a coloured graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T.

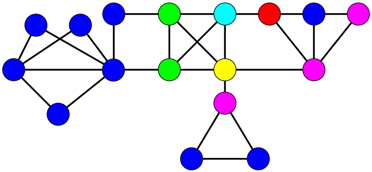


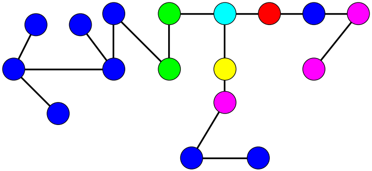


Spanning trees

Theorem

The number of moves required to flood a coloured graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T.

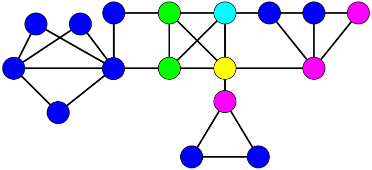


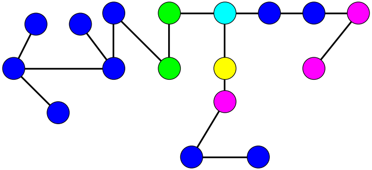


Spanning trees

Theorem

The number of moves required to flood a coloured graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T.

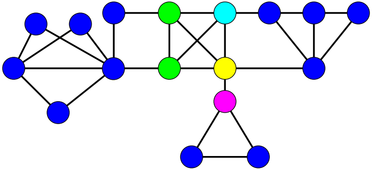


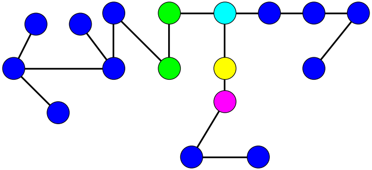


Spanning trees

Theorem

The number of moves required to flood a coloured graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T.

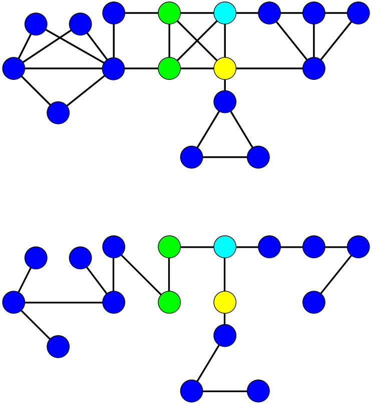




Spanning trees

Theorem

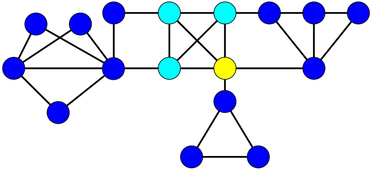
The number of moves required to flood a coloured graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T.

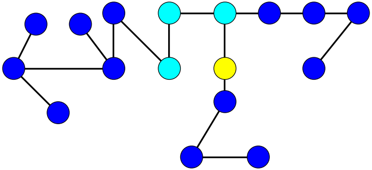


Spanning trees

Theorem

The number of moves required to flood a coloured graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T.

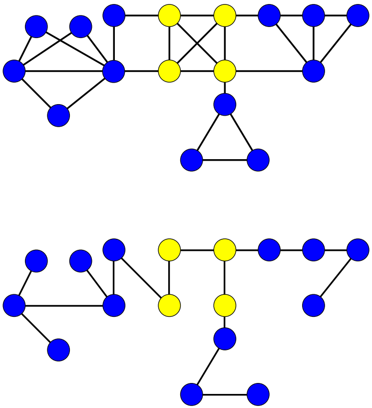




Spanning trees

Theorem

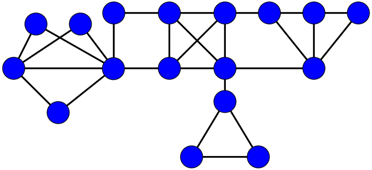
The number of moves required to flood a coloured graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T.

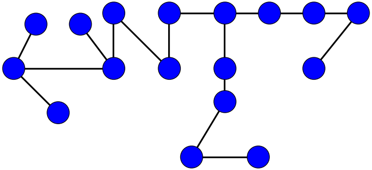


Spanning trees

Theorem

The number of moves required to flood a coloured graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T.





This is useless!

- In general, a graph has an exponential number of spanning trees.

This is useless!

- In general, a graph has an exponential number of spanning trees.
- Besides, FREE FLOOD IT is still NP-hard even on trees.

… or is it?

… or is it?

$$P \neq NP$$

… or is it?

$$\cancel{P \neq NP}$$

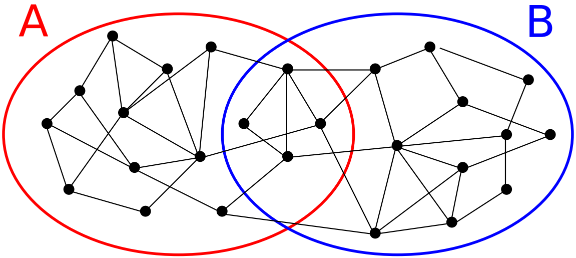
... or is it?

Source: finditinscotland.com

... or is it?

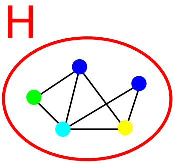
Source: finditinscotland.com

... or is it?



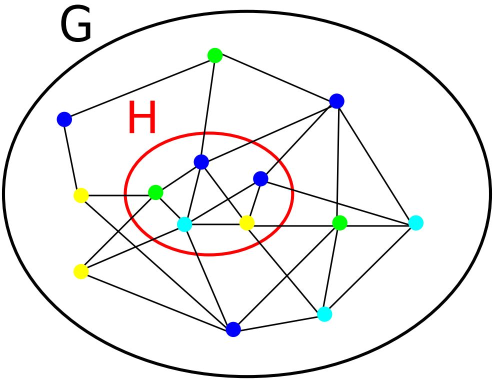
The number of moves required to flood G with colour d is at most the sum of the numbers of moves required to flood A and B respectively with colour d.

... or is it?



The number of moves required to flood a subgraph doesn't increase when we play in a larger graph.

... or is it?

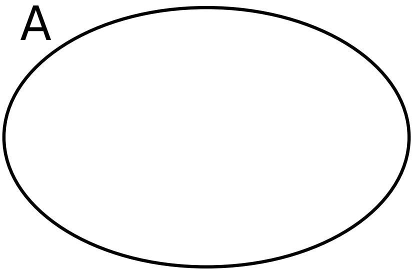


The number of moves required to flood a subgraph doesn't increase when we play in a larger graph.

Application I: Graphs with polynomially many connected subgraphs

Theorem

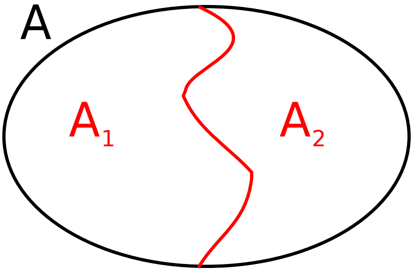
FREE FLOOD IT *can be solved in polynomial time on graphs that have only a polynomial number of connected subgraphs.*



Application I: Graphs with polynomially many connected subgraphs

Theorem

FREE FLOOD IT *can be solved in polynomial time on graphs that have only a polynomial number of connected subgraphs.*



Application I: Graphs with polynomially many connected subgraphs

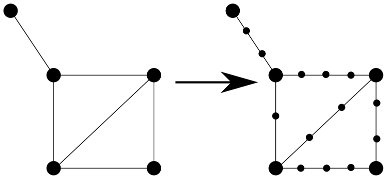
Classes of graphs with only a polynomial number of connected subgraphs include:

- paths
- cycles

Application I: Graphs with polynomially many connected subgraphs

Classes of graphs with only a polynomial number of connected subgraphs include:

- paths
- cycles
- subdivisions of any fixed graph H



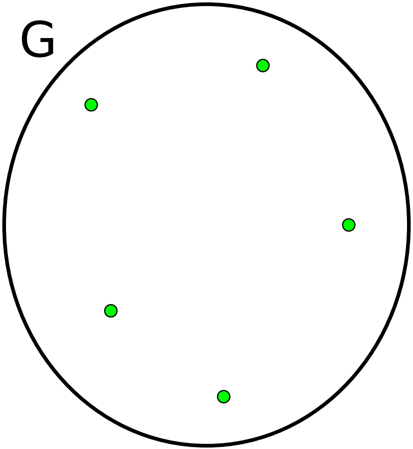
Application II: Connecting k points

Given a coloured graph G and a subset U of at most k vertices, k-LINKING FLOOD IT is the problem of determining the number of moves required to create a single monochromatic component containing U.

Theorem

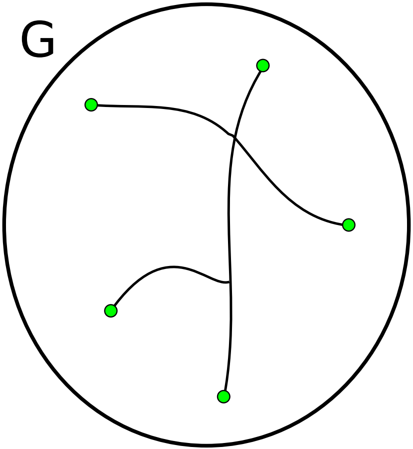
k-LINKING FLOOD IT can be solved in time $O(|V|^{k+3}|E|c^2 2^k)$ on a graph $G = (V, E)$ coloured with c colours.

Application II: Connecting k points



The number of moves required to connect U is equal to the minimum, taken over all subtrees T of G that contain U, of the number of moves required to flood T.

Application II: Connecting k points



The number of moves required to connect U is equal to the minimum, taken over all subtrees T of G that contain U, of the number of moves required to flood T.

Summary

Summary

- Flood-filling problems are FUN!

Summary

- Flood-filling problems are FUN!
- Solving c-Free-Flood-It or c-Fixed-Flood-It, for $c > 2$, is NP-hard in many situations.

Summary

- Flood-filling problems are FUN!
- Solving c-Free-Flood-It or c-Fixed-Flood-It, for $c > 2$, is NP-hard in many situations.
- The number of moves to flood an arbitrary graph can be characterised in terms of the number of moves to flood its spanning trees.
- This gives some useful facts about the behaviour of flooding operations on arbitrary graphs, and can be used to give some polynomial-time algorithms.

Open problems

- Complexity of 3-Fixed-Flood-It and 3-Free-Flood-It on $3 \times n$ boards.

Open problems

- Complexity of 3-Fixed-Flood-It and 3-Free-Flood-It on $3 \times n$ boards.
- Complexity of 3-Fixed-Flood-It and 3-Free-Flood-It on trees.

Open problems

- Complexity of 3-Fixed-Flood-It and 3-Free-Flood-It on $3 \times n$ boards.
- Complexity of 3-Fixed-Flood-It and 3-Free-Flood-It on trees.
- Is k-Linking-Flood-It fixed parameter tractable, with parameter k?

Open problems

- Complexity of 3-Fixed-Flood-It and 3-Free-Flood-It on $3 \times n$ boards.
- Complexity of 3-Fixed-Flood-It and 3-Free-Flood-It on trees.
- Is k-Linking-Flood-It fixed parameter tractable, with parameter k?
- Given a graph G,
  1. what colouring with c colours requires the most moves?
  2. what proper colouring requires the fewest?

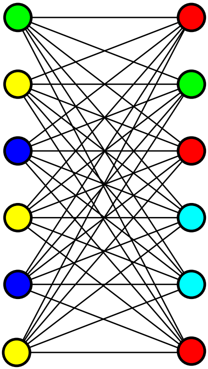
Open problems

- Complexity of 3-Fixed-Flood-It and 3-Free-Flood-It on $3 \times n$ boards.
- Complexity of 3-Fixed-Flood-It and 3-Free-Flood-It on trees.
- Is k-Linking-Flood-It fixed parameter tractable, with parameter k?
- Given a graph G,
  1. what colouring with c colours requires the most moves?
  2. what proper colouring requires the fewest?
- Does the Loch Ness Monster exist?

Thank you

Some easy cases

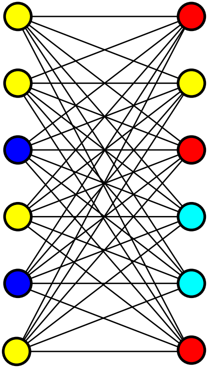
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

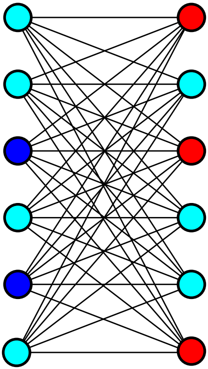
- Complete bipartite graphs



Either $c-1$ or c moves are required.

Some easy cases

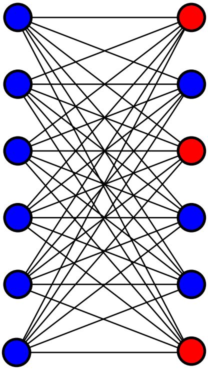
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

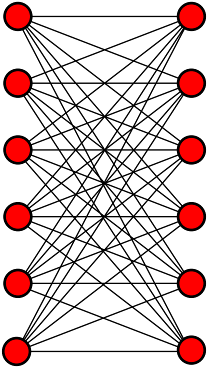
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

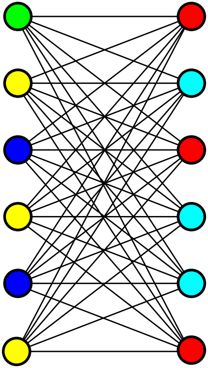
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

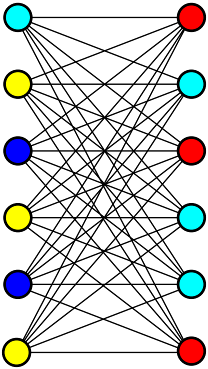
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

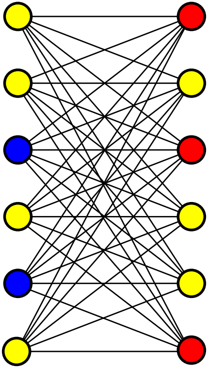
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

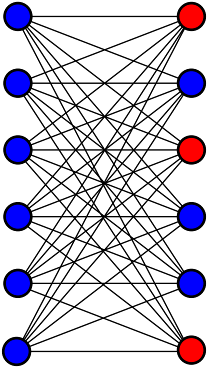
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

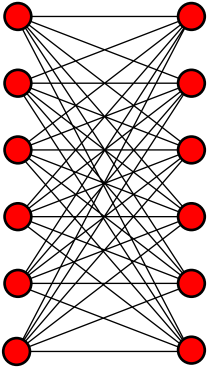
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

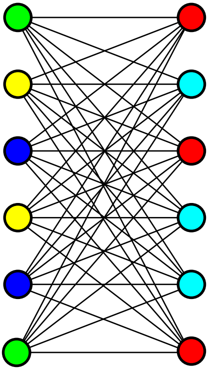
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

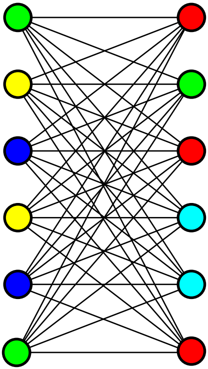
- Complete bipartite graphs



Either $c-1$ or c moves are required.

Some easy cases

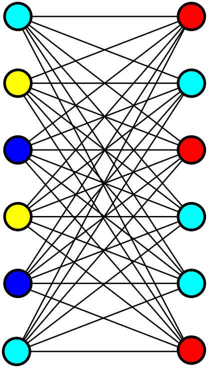
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

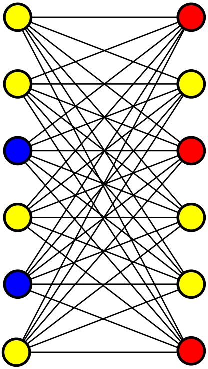
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

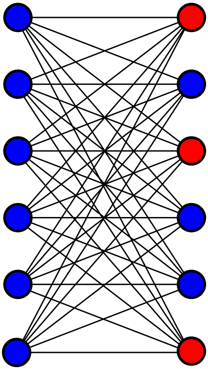
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

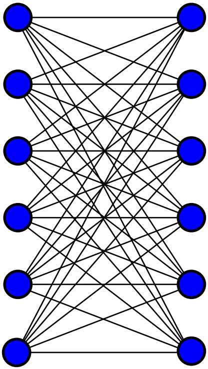
- Complete bipartite graphs



Either $c - 1$ or c moves are required.

Some easy cases

- Complete bipartite graphs



Either $c - 1$ or c moves are required.