Random Spanning Trees

Bhishmaraj S

Chennai Mathematical Institute

bhishma@cmi.ac.in

June 11, 2020

Overview

1. Introduction
   - Problem Definition
   - Applications
   - A bird's eye view of existing algorithms
2. Random Walk Based Algorithms
   - Aldous-Broder Algorithm
   - Wilson's Algorithm
3. Second Section

Problem Definition

Given an undirected connected graph $G = (V, E)$, sample a spanning tree T with probability $\frac{1}{|\mathcal{T}|}$ where $\mathcal{T}$ denotes the set of all spanning trees of G.

Applications

Sampling spanning trees pops up in surprising problems in TCS such as

- Constructing expanders ([GRV09], [FGRV14])
- Approximation algorithms for the travelling salesman problem([GSS11], [AGM+17])
- Graph Sparsification ([FH10], [DK16])
- Analysis of network reliability ([Col87],[NC90], [CDM88])
- Sequence shuffling problem in Bioinformatics ([KMUW96])
- Maze Generation

Applications

Sampling spanning trees pops up in surprising problems in TCS such as

- Constructing expanders ([GRV09], [FGRV14])
- Approximation algorithms for the travelling salesman problem([GSS11], [AGM+17])
- Graph Sparsification ([FH10], [DK16])
- Analysis of network reliability ([Col87],[NC90], [CDM88])
- Sequence shuffling problem in Bioinformatics ([KMUW96])
- Maze Generation

Applications

Sampling spanning trees pops up in surprising problems in TCS such as

- Constructing expanders ([GRV09], [FGRV14])
- Approximation algorithms for the travelling salesman problem([GSS11], [AGM+17])
- Graph Sparsification ([FH10], [DK16])
- Analysis of network reliability ([Col87],[NC90], [CDM88])
- Sequence shuffling problem in Bioinformatics ([KMUW96])
- Maze Generation

Applications

Sampling spanning trees pops up in surprising problems in TCS such as

- Constructing expanders ([GRV09], [FGRV14])
- Approximation algorithms for the travelling salesman problem([GSS11], [AGM+17])
- Graph Sparsification ([FH10], [DK16])
- Analysis of network reliability ([Col87],[NC90], [CDM88])
- Sequence shuffling problem in Bioinformatics ([KMUW96])
- Maze Generation

Applications

Sampling spanning trees pops up in surprising problems in TCS such as

- Constructing expanders ([GRV09], [FGRV14])
- Approximation algorithms for the travelling salesman problem([GSS11], [AGM$^+$17])
- Graph Sparsification ([FH10], [DK16])
- Analysis of network reliability ([Col87],[NC90], [CDM88])
- Sequence shuffling problem in Bioinformatics ([KMUW96])
- Maze Generation

Applications

Sampling spanning trees pops up in surprising problems in TCS such as

- Constructing expanders ([GRV09], [FGRV14])
- Approximation algorithms for the travelling salesman problem([GSS11], [AGM$^+$17])
- Graph Sparsification ([FH10], [DK16])
- Analysis of network reliability ([Col87],[NC90], [CDM88])
- Sequence shuffling problem in Bioinformatics ([KMUW96])
- Maze Generation

Applications

Sampling spanning trees pops up in surprising problems in TCS such as

- Constructing expanders ([GRV09], [FGRV14])
- Approximation algorithms for the travelling salesman problem([GSS11], [AGM$^+$17])
- Graph Sparsification ([FH10], [DK16])
- Analysis of network reliability ([Col87],[NC90], [CDM88])
- Sequence shuffling problem in Bioinformatics ([KMUW96])
- Maze Generation

Proposed Algorithms

In this talk we would review some of the algorithms proposed for this problem and get into details of [HX16]

- **Random Walk Based** - Simulate variants of random walk on the input graph
- **Determinant Based** - Based on Kirchoff Matrix Tree theorem and involve computing determinants of the laplacian matrix
- **Approximation Algorithms**

Proposed Algorithms

In this talk we would review some of the algorithms proposed for this problem and get into details of [HX16]

- **Random Walk Based** - Simulate variants of random walk on the input graph
- **Determinant Based** - Based on Kirchoff Matrix Tree theorem and involve computing determinants of the laplacian matrix
- **Approximation Algorithms**

Proposed Algorithms

In this talk we would review some of the algorithms proposed for this problem and get into details of [HX16]

- **Random Walk Based** - Simulate variants of random walk on the input graph
- **Determinant Based** - Based on Kirchoff Matrix Tree theorem and involve computing determinants of the laplacian matrix
- **Approximation Algorithms**

Proposed Algorithms

In this talk we would review some of the algorithms proposed for this problem and get into details of [HX16]

- **Random Walk Based** - Simulate variants of random walk on the input graph
- **Determinant Based** - Based on Kirchoff Matrix Tree theorem and involve computing determinants of the laplacian matrix
- **Approximation Algorithms**

Aldous-Broder Algorithm

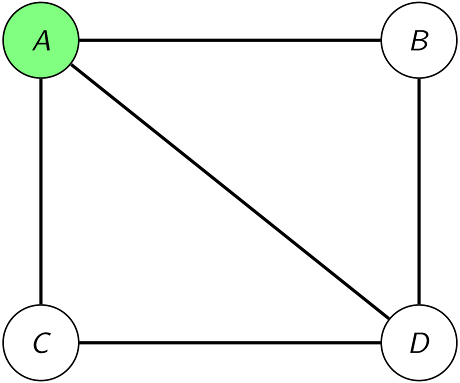
Andrei Broder and David Aldous independently invented the following algorithm

Input: $G = (V, E)$
Output: A random spanning tree

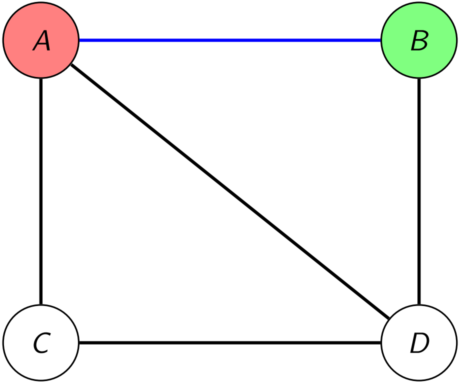
Choose a starting vertex s arbitrarily
$T_V \leftarrow \{s\}$, $T_E \leftarrow \emptyset$
while $|T_V| < |V$ **do**
  $next =_{u.a.r} N(s)$
  if $next \notin T_V$ **then**
    $T_V = T_V \cup \{next\}$
    $T_E = T_E \cup \{(s, next)\}$
  end
  $s = next$
end
return $T = (T_V, T_E)$

Algorithm 1: Aldous-Broder Algorithm

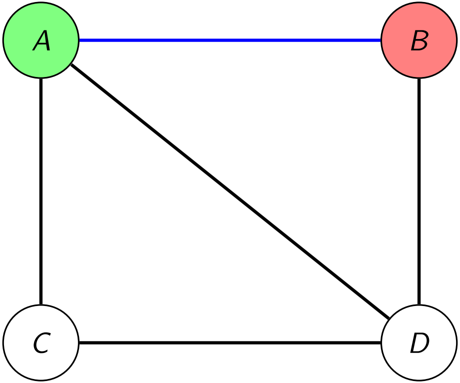
Aldous-Broder Algorithm example



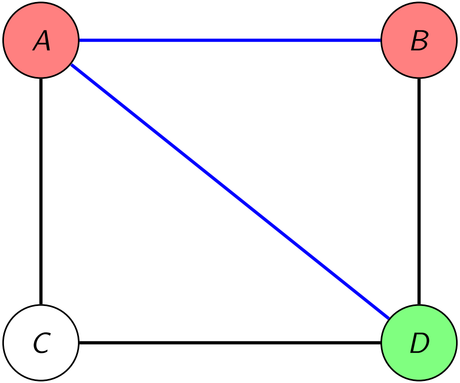
Aldous-Broder Algorithm example



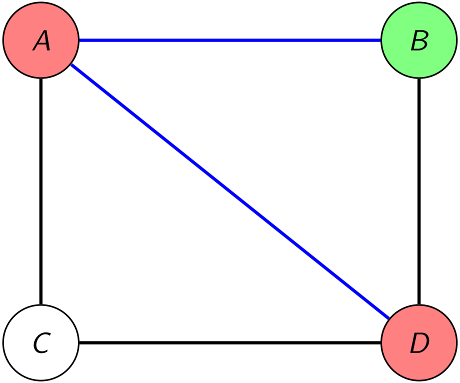
Aldous-Broder Algorithm example



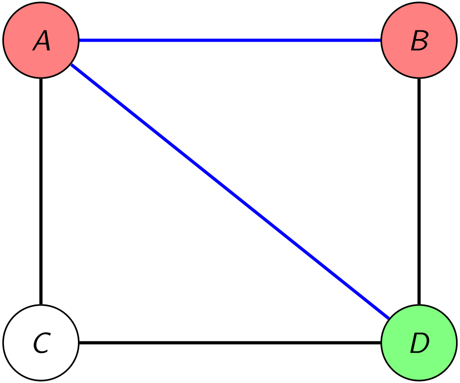
Aldous-Broder Algorithm example



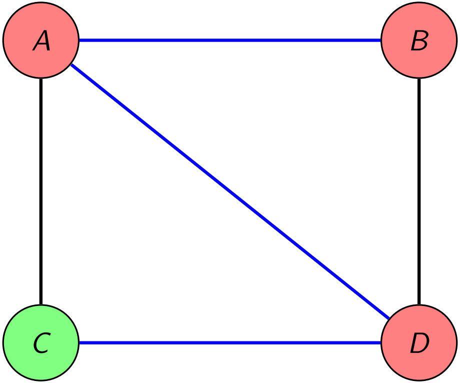
Aldous-Broder Algorithm example



Aldous-Broder Algorithm example



Aldous-Broder Algorithm example



Running Time

Cover Time

$cov_G(u) :=$ The expected number of steps for a random walk starting at u to visit all the vertices in G

Cover Time of G

$cov_G := \max_{u \in V_G} cov_G(u)$

It is known that $cov_G = \mathcal{O}(|V|\,|E|) = \mathcal{O}(|V|^3)$

Running Time

Cover Time
$cov_G(u)$:= The expected number of steps for a random walk starting at u to visit all the vertices in G

Cover Time of G
$cov_G := \max_{u \in V_G} cov_G(u)$

It is known that $cov_G = \mathcal{O}(|V|\,|E|) = \mathcal{O}(|V|^3)$

Running Time

Cover Time
$cov_G(u) :=$ The expected number of steps for a random walk starting at u to visit all the vertices in G

Cover Time of G
$cov_G := \max_{u \in V_G} cov_G(u)$

It is known that $cov_G = \mathcal{O}(|V|\,|E|) = \mathcal{O}(|V|^3)$

Running Time

Cover Time

$cov_G(u) :=$ The expected number of steps for a random walk starting at u to visit all the vertices in G

Cover Time of G

$cov_G := \max_{u \in V_G} cov_G(u)$

It is known that $cov_G = \mathcal{O}(|V|\ |E|) = \mathcal{O}(|V|^3)$

Wilson's Algorithm

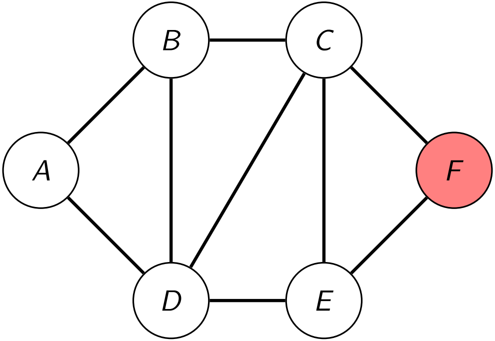
In 1996 Wilson proposed a variant of random walk called loop erased random walk .

Input: $G = (V, E)$ and root $r \in V$
Output: Parent pointer array called *next*

inTree$[i] \leftarrow$ *False*, $\forall\ i \neq r$
inTree$[r] \leftarrow$ *True*
next$[r] \leftarrow$ NULL
for $i \leftarrow 1$ *to* n **do**
    $u = i$
    while $\neg inTree[u]$ **do**
        next$[u] =_{u.a.r} N(u)$
        $u =$ next$[u]$
    end
    $u = i$
    while $\neg inTree[u]$ **do**
        inTree$[u] =$ *True*
        $u =$ next$[u]$
    end
end
return *next*

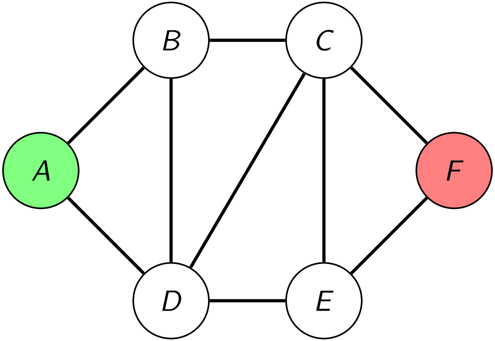
Algorithm 2: Wilson's Algorithm

Wilson's algorithm example

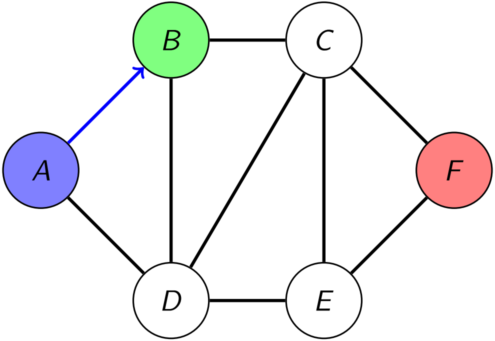


Wilson's algorithm example

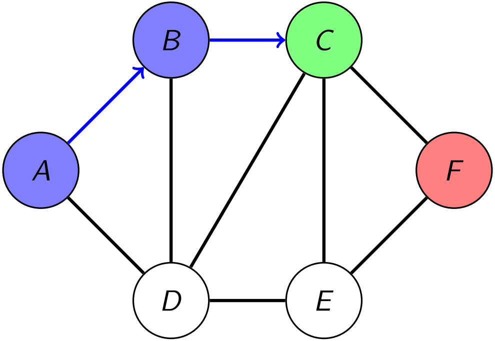
Start at A



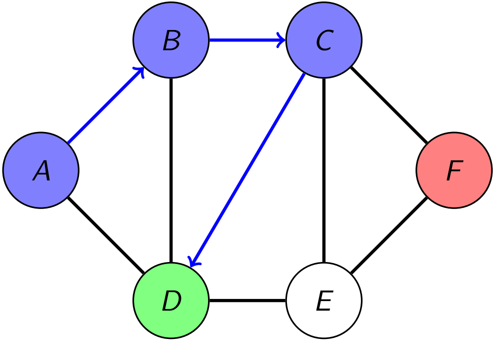
Wilson's algorithm example



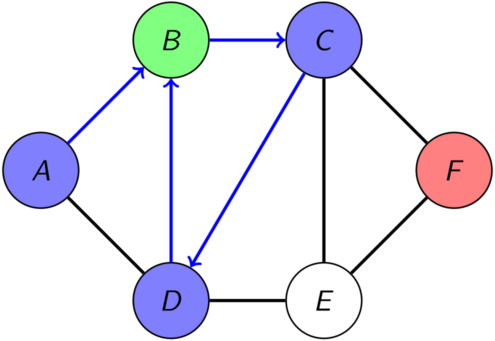
Wilson's algorithm example



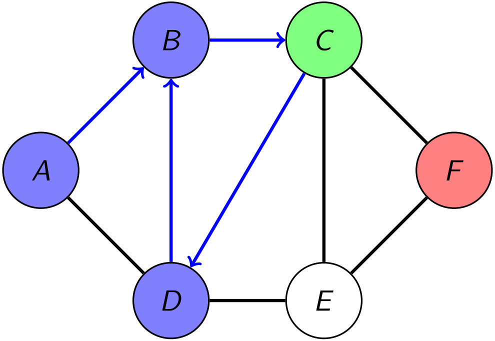
Wilson's algorithm example



Wilson's algorithm example

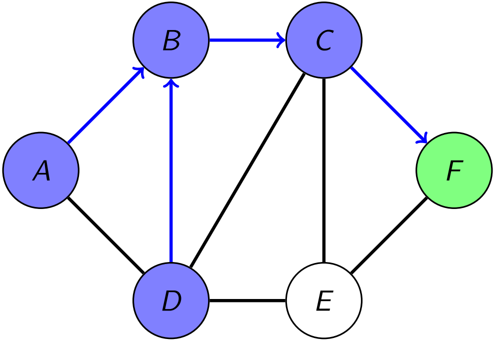


Wilson's algorithm example



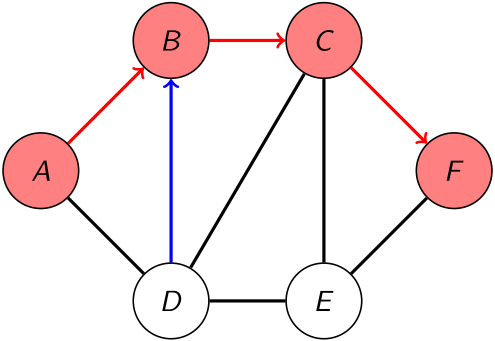
Wilson's algorithm example

Notice the **next(C)** has changed from D to F.



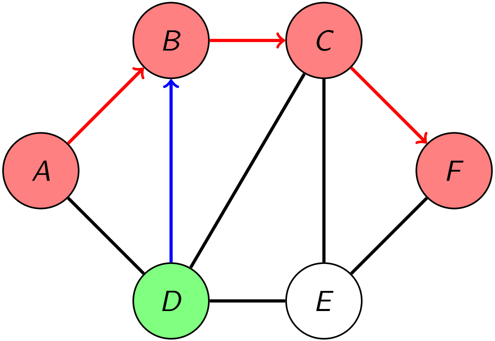
Wilson's algorithm example

Since a vertex already in the tree has been reached (namely F), starting from A we trace the successors and set their **inTree** value to *True*

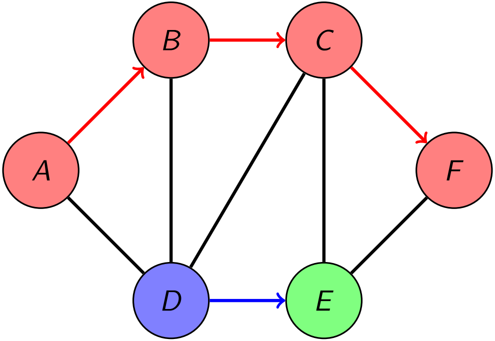


Wilson's algorithm example

Since B, C are already in the tree they will be skipped and now will start at D

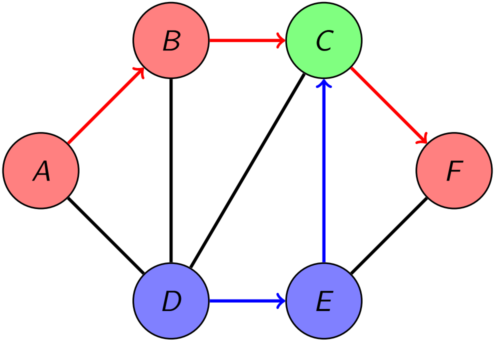


Wilson's algorithm example



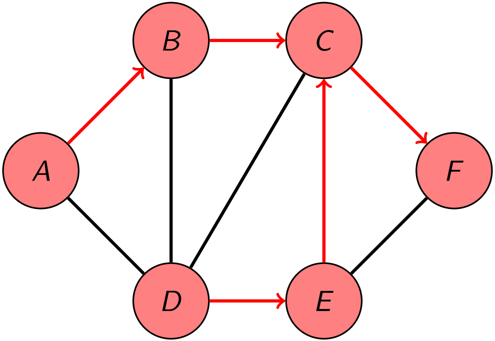
Wilson's algorithm example

Since C is already in the tree, the random walk stops and the algorithm retraces from D and includes the vertices into the tree



Wilson's algorithm example

Since C is already in the tree, the random walk stops and the algorithm retraces from D and includes the vertices into the tree

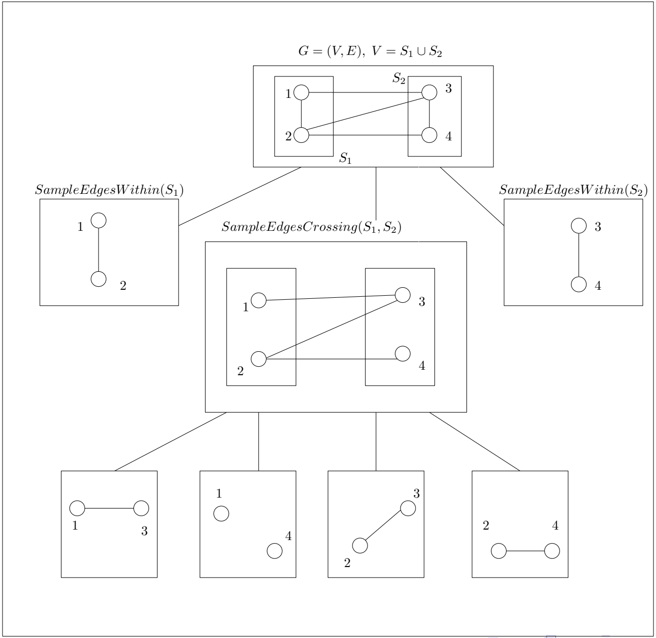


Naive algorithm using effective resistance

Input: $G = (V, E)$ and L_G^+
Output: Set of edges corresponding to a random spanning tree

```
for e = (u, v) ∈ E do
    R_e^eff = (χ_u − χ_v)^T L_G^+ (χ_u − χ_v);
    if (X ∼ Bernoulli(R_e^eff)) = 1 then
        Add edge e to the spanning tree;
        G = G/e;
    else
        G = G \ e ;
    end
    Update L_G^+ ;
end
```

Algorithm 3: Sampling uniform spanning tree using chain rule


$G = (V, E),\ V = S_1 \cup S_2$
S_2
1
3
2
4
S_1
$SampleEdgesWithin(S_1)$
1
2
$SampleEdgesCrossing(S_1, S_2)$
1
3
2
4
$SampleEdgesWithin(S_2)$
3
4
1
3
1
4
3
2
2
4

Multiple Columns

Heading

1. Statement
2. Explanation
3. Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Table

Treatments	**Response 1**	**Response 2**
Treatment 1	0.0003262	0.562
Treatment 2	0.0015681	0.910
Treatment 3	0.0009271	0.296

Table: Table caption

Theorem

Theorem (Mass–energy equivalence)

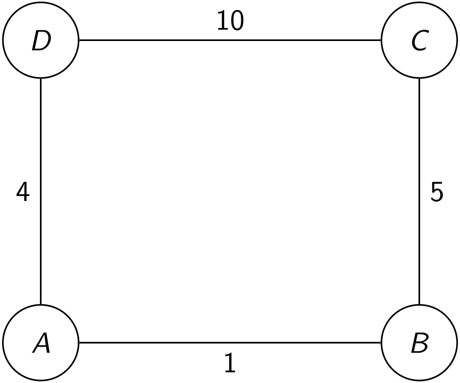
$E = mc^2$

Verbatim

Example (Theorem Slide Code)

```
\begin{frame}
\frametitle{Theorem}
\begin{theorem}[Mass--energy equivalence]
$E = mc^2$
\end{theorem}
\end{frame}
```

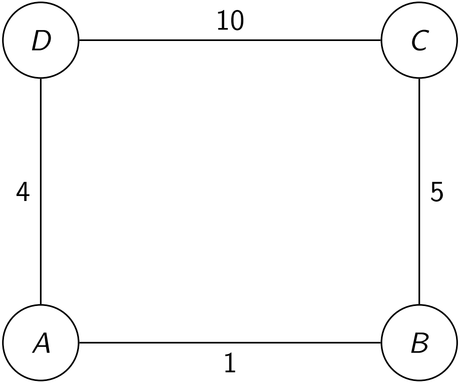
Figure



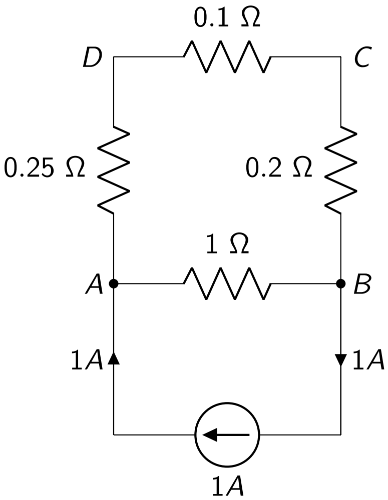
(a) The original graph G

(b) The electric network version of G

Figure



(a) The original graph G



(b) The electric network version of G

Thank You

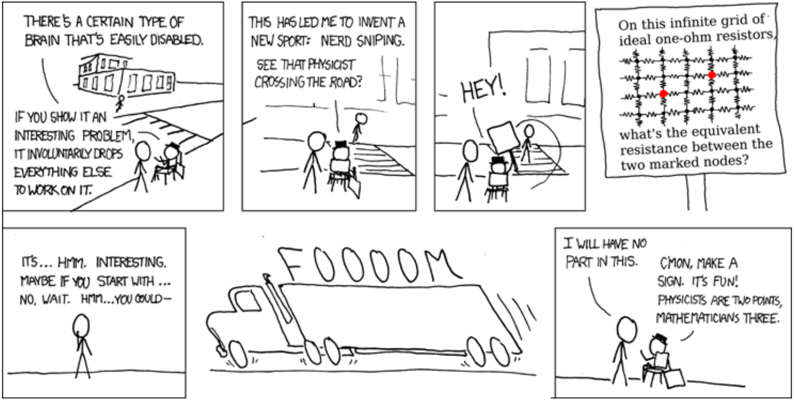


Figure: Obligatory xkcd

References

Arash Asadpour, Michel X. Goemans, Aleksander Mdry, Shayan Oveis Gharan, and Amin Saberi, *An o(log n/log log n)-approximation algorithm for the asymmetric traveling salesman problem*, Operations Research **65** (2017), no. 4, 1043–1061.

Charles J Colbourn, Bradley M Debroni, and Wendy J Myrvold, *Estimating the coefficients of the reliability polynomial*, Congressus Numerantium **62** (1988), 217–223.

Charles J. Colbourn, *The combinatorics of network reliability*, Oxford University Press, Inc., USA, 1987.

Shlomi Dolev and Daniel Khankin, *Random spanning trees for expanders, sparsifiers, and virtual network security*, arXiv preprint arXiv:1612.02569 (2016).

Alan Frieze, Navin Goyal, Luis Rademacher, and Santosh Vempala, *Expanders via random spanning trees*, SIAM Journal on Computing