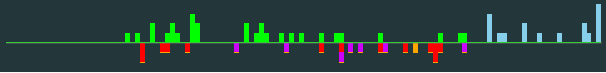
G: Graph Theft Auto

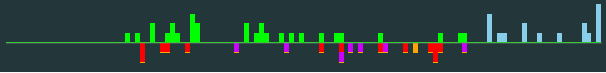
Problem author: Lammert Westerdijk



Problem: On an undirected unweighted graph, some gangsters walk from their hideout at unknown vertex to vertex 1 using the shortest path. You can place cameras at vertices that detect if the gangsters walk past this vertex and the number of edges they have traversed so far. Determine the minimum number of required cameras to spot the gangsters and be able to determine their hideout.

G: Graph Theft Auto

Problem author: Lammert Westerdijk

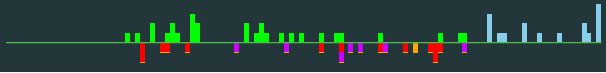


Problem: On an undirected unweighted graph, some gangsters walk from their hideout at unknown vertex to vertex 1 using the shortest path. You can place cameras at vertices that detect if the gangsters walk past this vertex and the number of edges they have traversed so far. Determine the minimum number of required cameras to spot the gangsters and be able to determine their hideout.

Step 1: Determine the shortest paths. We can calculate the shortest path from any starting vertex to all other vertices in $\mathcal{O}(n+m)$ running time using BFS.
Run BFS for all starting vertices is too slow ($\mathcal{O}(n^2+mn)$).
We can run BFS from vertex 1: $\mathcal{O}(n+m)$ is fast enough.

G: Graph Theft Auto

Problem author: Lammert Westerdijk



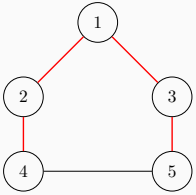
Problem: On an undirected unweighted graph, some gangsters walk from their hideout at unknown vertex to vertex 1 using the shortest path. You can place cameras at vertices that detect if the gangsters walk past this vertex and the number of edges they have traversed so far. Determine the minimum number of required cameras to spot the gangsters and be able to determine their hideout.

Step 1: Determine the shortest paths. We can calculate the shortest path from any starting vertex to all other vertices in $\mathcal{O}(n+m)$ running time using BFS.

Run BFS for all starting vertices is too slow ($\mathcal{O}(n^2+mn)$).

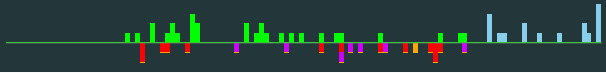
We can run BFS from vertex 1: $\mathcal{O}(n+m)$ is fast enough.

The gangsters always walk upward to the root of the BFS tree (forget all other edges):



G: Graph Theft Auto

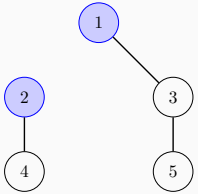
Problem author: Lammert Westerdijk



Step 2: Place cameras optimally.

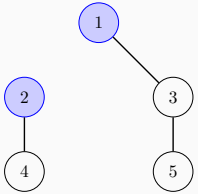
Step 2: Place cameras optimally.

Observation: If different hideouts are not distinguished by a camera, then they will not be distinguished higher in tree as well. We can cut edges from cameras to parent and treat subtrees separately:



Step 2: Place cameras optimally.

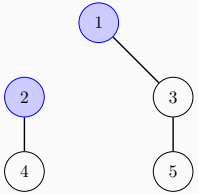
Observation: If different hideouts are not distinguished by a camera, then they will not be distinguished higher in tree as well. We can cut edges from cameras to parent and treat subtrees separately:



Condition: We must distinguish all hideouts: subtrees are all paths (2 children not distinguishable).

Step 2: Place cameras optimally.

Observation: If different hideouts are not distinguished by a camera, then they will not be distinguished higher in tree as well. We can cut edges from cameras to parent and treat subtrees separately:

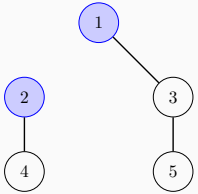


Condition: We must distinguish all hideouts: subtrees are all paths (2 children not distinguishable).

Solution: All leaves need their own path. We can create these paths by cutting off as high as possible: the answer is the number of leaves in BFS-tree!

Step 2: Place cameras optimally.

Observation: If different hideouts are not distinguished by a camera, then they will not be distinguished higher in tree as well. We can cut edges from cameras to parent and treat subtrees separately:



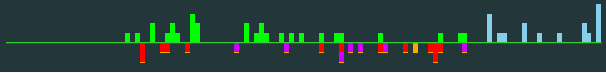
Condition: We must distinguish all hideouts: subtrees are all paths (2 children not distinguishable).

Solution: All leaves need their own path. We can create these paths by cutting off as high as possible: the answer is the number of leaves in BFS-tree!

Running time: $\mathcal{O}(n + m)$

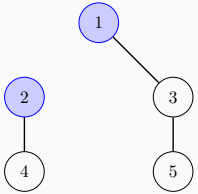
G: Graph Theft Auto

Problem author: Lammert Westerdijk



Step 2: Place cameras optimally.

Observation: If different hideouts are not distinguished by a camera, then they will not be distinguished higher in tree as well. We can cut edges from cameras to parent and treat subtrees separately:



Condition: We must distinguish all hideouts: subtrees are all paths (2 children not distinguishable).

Solution: All leaves need their own path. We can create these paths by cutting off as high as possible: the answer is the number of leaves in BFS-tree!

Running time: $\mathcal{O}(n + m)$

Statistics: 66 submissions, 29 accepted, 16 unknown