T182 - Cave Exploration

Percy Wong {percywtc}
The Problem

You are at some node of an unknown tree

You have to find out the structure of this tree by:

- Walking through the corridors
- Placing and removing flags
The Problem

- **void reportCorridor(int x)**
  - He will enter corridor A if $x = 1, 6, 11, ...$
  - He will enter corridor B if $x = 2, 7, 12, ...$
  - He will enter corridor C if $x = 3, 8, 13, ...$
  - He will enter corridor D if $x = 4, 9, 14, ...$
  - He will enter corridor E if $x = 5, 10, 15, ...$

- **void placeFlag()**
- **void removeFlag()**
- **int countFlags()**
- **bool detectDeadEnd()**
## Statistics

<table>
<thead>
<tr>
<th>Max</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Subtasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>39.641</td>
<td>28.911</td>
<td>13:35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>14:28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17:19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11:12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8:12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>22:4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15:4</td>
</tr>
</tbody>
</table>
Difficulties?

- You don’t even know where you (or Robo) are
- You don’t even know how many corridors in a room
  - so you can’t go back to the room you just come from easily

So the task seems impossible?!
Difficulties?

So the task seems impossible?!

We should learn from the subtasks!
Subtask 1

It is given that the tree is “star-shape”

So our only possible initial positions are:

- at the centroid
- at one of the leaves
Subtask 1

The first crucial observation

- if we keep performing `chooseCorridor(1)`
- we can walk through the whole star
- why?
Subtask 1

The first crucial observation

- if we keep performing `chooseCorridor(1)`
- we can walk through the whole star
- why?
Subtask 1

The first crucial observation

- if we keep performing chooseCorridor(1)
- we can walk through the whole star

So we can actually do something like

- keep walking by performing chooseCorridor(1)
- mark the node as visited if not visited before
- maintain visited by placeFlag() and countFlags()
- count the number of nodes is sufficient
Subtask 1

do 60000 times: // you can actually reduce iterations
chooseCorridor(1)
if (countFlags() == 0) // meaning not visited
   placeFlag()
treeSize++

```
treeSize
1 2
1 3
...
1 treeSize
```
Subtask 2

It is given that the tree is “star-shape” or “chain-shape”

Is the observation in “star-shape” true in “chain-shape”?

- if we keep performing chooseCorridor(1)
- we can walk through the whole chain
- why?
Subtask 2

It is given that the tree is “star-shape” or “chain-shape”

Is the observation in “star-shape” true in “chain-shape”?

- if we keep performing `chooseCorridor(1)`
- we can walk through the whole chain
- why?
Subtask 2

Is the observation in “star-shape” true in “chain-shape”?

- if we keep performing chooseCorridor(1)
- we can walk through the whole chain

So we can actually do this again:

- keep walking by performing chooseCorridor(1)
- mark the node as visited if not visited before
- maintain visited by placeFlag() and countFlags()
- count the number of nodes is sufficient
Subtask 2

But then, how to determine if the cave is “star” or “chain”?
Subtask 2

But then, how to determine if the cave is “star” or “chain”? We can make use of detectDeadEnd()

- “star” has many dead-ends
- “chain” has exactly two
Subtask 2

do 60000 times: // you can actually reduce iterations
chooseCorridor(1)
if (countFlags() == 0) // meaning not visited
   placeFlag()
treeSize++
if (detectDeadEnd())
   deadEndCount++
Subtask 2

deadEndCount > 2

N
1 2
1 3
...
1 N

deadEndCount == 2

N
1 2
2 3
...
N-1 N
Subtask 2

Be careful with these two cases

- they have the properties of both “star-shape” and “chain-shape”
Subtask 3 - 7

From now on, the cave can be any tree

- Subtask 3 - 5: \( F \) is quite large
- Subtask 6 - 7: \( F \) is only 2, perhaps expecting a different solution

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \max N )</th>
<th>( L )</th>
<th>( F )</th>
<th>( D )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>17</td>
<td>( 100 )</td>
<td>( 60000 )</td>
<td>true</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>( 100 )</td>
<td>( 60000 )</td>
<td>true</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>( 100 )</td>
<td>( 60000 )</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>( 100 )</td>
<td>( 60000 )</td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>( 100 )</td>
<td>( 30000 )</td>
<td>false</td>
</tr>
</tbody>
</table>
Subtask 3 - 7

The crucial crucial crucial observation again again again again

- if we keep performing `chooseCorridor(1)`
- we can walk through the whole tree
- the behaviour is exactly the same as:
  - running DFS
  - Euler Tour
Subtask 3 // $L = 60000$, $F = 10000$, $D = \text{true}$

With $F = 10000$, we can actually number the nodes with flags

- each time reaching a node, check if it’s new by countFlags() == 0
- if it’s new, instead of putting ONE flag, put a new unused number
- for reaching the $k$-th new node, place $k$ flags

- so nodes have label now
- by maintaining the label before and after chooseCorridor() 
- we can easily output the tree
Subtask 3 // $L = 60000, F = 10000, D = \text{true}$

consider the tree is rooted, with our initial position as the root

after calling chooseCorridor(1)

- if $\text{countFlags}() = 0$, we reach a new node
  - increase $N$ by 1
  - place $N$ flags on this node to label it
  - memorize this edge, between parent and current

requires at most $1+2+3+\ldots+100 = 5050$ flags
Subtask 4 // \( L = 60000, F = 1000, D = \text{true} \)

5050 flags == too many

we can optimize it by only memorizing the depth \( \% \ 3 \)

still able to know where is our current position by

- comparing the label of previous and current node
- if \((3->2 \ || \ 2->1 \ || \ 1->3)\), we go up
- if \((3->1 \ || \ 2->3 \ || \ 1->2)\), we go down

requires at most \(1+2+3+\ldots+3 = 297\) flags
Subtask 5 // $L = 60000, F = 100, D = \text{true}$

297 flags == still too many

after calling \text{chooseCorridor}(1)

- if $\text{countFlags}() \neq 0$, we back to parent
  - we are able to know our current position
- if $\text{countFlags}() = 0$, we reach a new node
  - increase $N$ by 1
  - place $N$ flags \textbf{ONE flag} on this node to mark visited
- we are still able to know our current position

requires at most $1+1+\ldots+1 = 100$ flags
Subtask 6 - 7 // F = 2

Subtask 6 - 7 are with extremely small F, expecting a very different solution

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>maxN = 100, L = 60000, F = 10000, D = true</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>maxN = 100, L = 60000, F = 1000, D = true</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>maxN = 100, L = 60000, F = 100, D = true</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>maxN = 100, L = 60000, F = 2, D = true</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>maxN = 100, L = 30000, F = 2, D = false</td>
</tr>
</tbody>
</table>
Subtask 6 // \( L = 60000, \ F = 2, \ D = \text{true} \)

with only 2 flags, what can we do? the idea is

- for each node, find the number of children
- do this recursively
Subtask 6 // $L = 60000$, $F = 2$, $D = \text{true}$

with only 2 flags, what can we do? the idea is

- for each node, find the number of children
- do this recursively

if we place a flag at the root

- keep running `chooseCorridor(1)`
- when we meet `countFlags() == 1`
  - we back to the root
  - meaning we walked through the subtree of our child
- but we still don’t know how many children the root has
Subtask 6 // $L = 60000$, $F = 2$, $D = \text{true}$

with only 2 flags, what can we do? the idea is

- for each node, find the number of children
- do this recursively

if we place a flag at the root, and also the first child

- when we meet $\text{countFlags}() = 1$
  - we back to the root first child
- continue to run $\text{chooseCorridor}(1)$
  - we can count the number of children with the method stated in the previous page
  - when we encounter $\text{countFlags}() = 1$ twice consecutively, we back to the first child
Subtask 6 // \( L = 60000, F = 2, D = \text{true} \)

with only 2 flags, what can we do? the idea is

- for each node, find the number of children
- do this recursively

for nodes other than the root

- we incorrectly count the parent as a child
- so just delete the last child found

as everytime after solving a node, we are at its first child

- we can go back by running chooseCorridor(1)
- until it reaches the first child
Subtask 6 // L = 60000, F = 2, D = true

with only 2 flags, what can we do? the idea is

- for each node, find the number of children
- do this recursively

have to carefully handle the dead-ends (leaves)

- especially when the first child is a leaf
- easiest way: detectDeadEnd()
Subtask 7 // L = 30000, F = 2, D = false

with only 2 flags, what can we do? the idea is

- for each node, find the number of children
- do this recursively

have to carefully handle the dead-ends (leaves)

- especially when the first child is a leaf
- easiest way: detectDeadEnd()
- maintaining subtree size
- by counting chooseCorridor(1) called before getting back
Subtask 7 // \( L = 30000, F = 2, D = \text{false} \)

my implementation on this solution has called (in worst cases)

- \( \text{chooseCorridor}(1) \) for 29700 times
- enough to get 100 points :) 

if for every non-root nodes

- place the flag at itself and its first child
- place the flag at parent and itself
- worst case: calling \( \text{chooseCorridor}(1) \) for 10099 times
- (thanks Steven Lau - cylau)