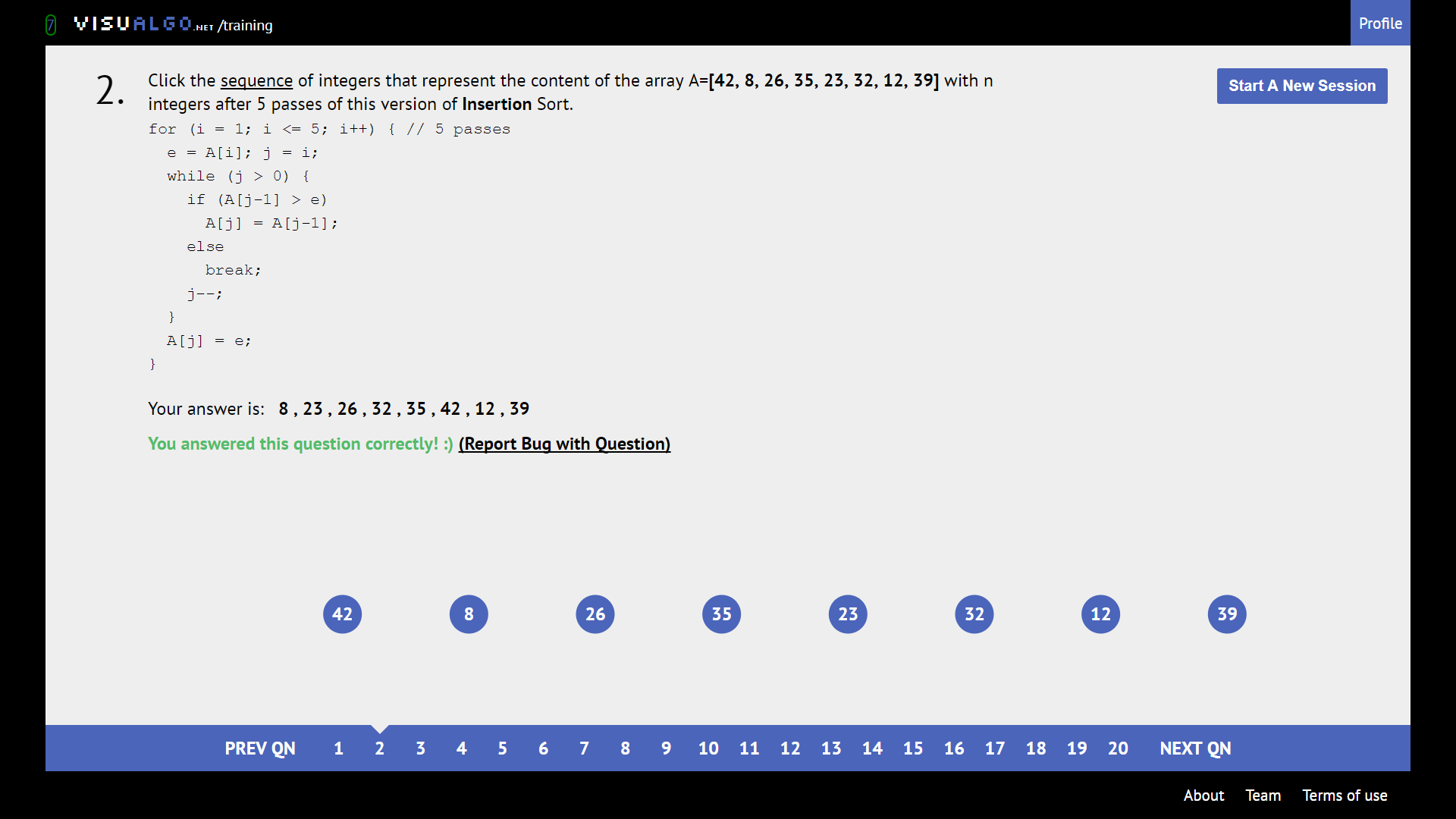
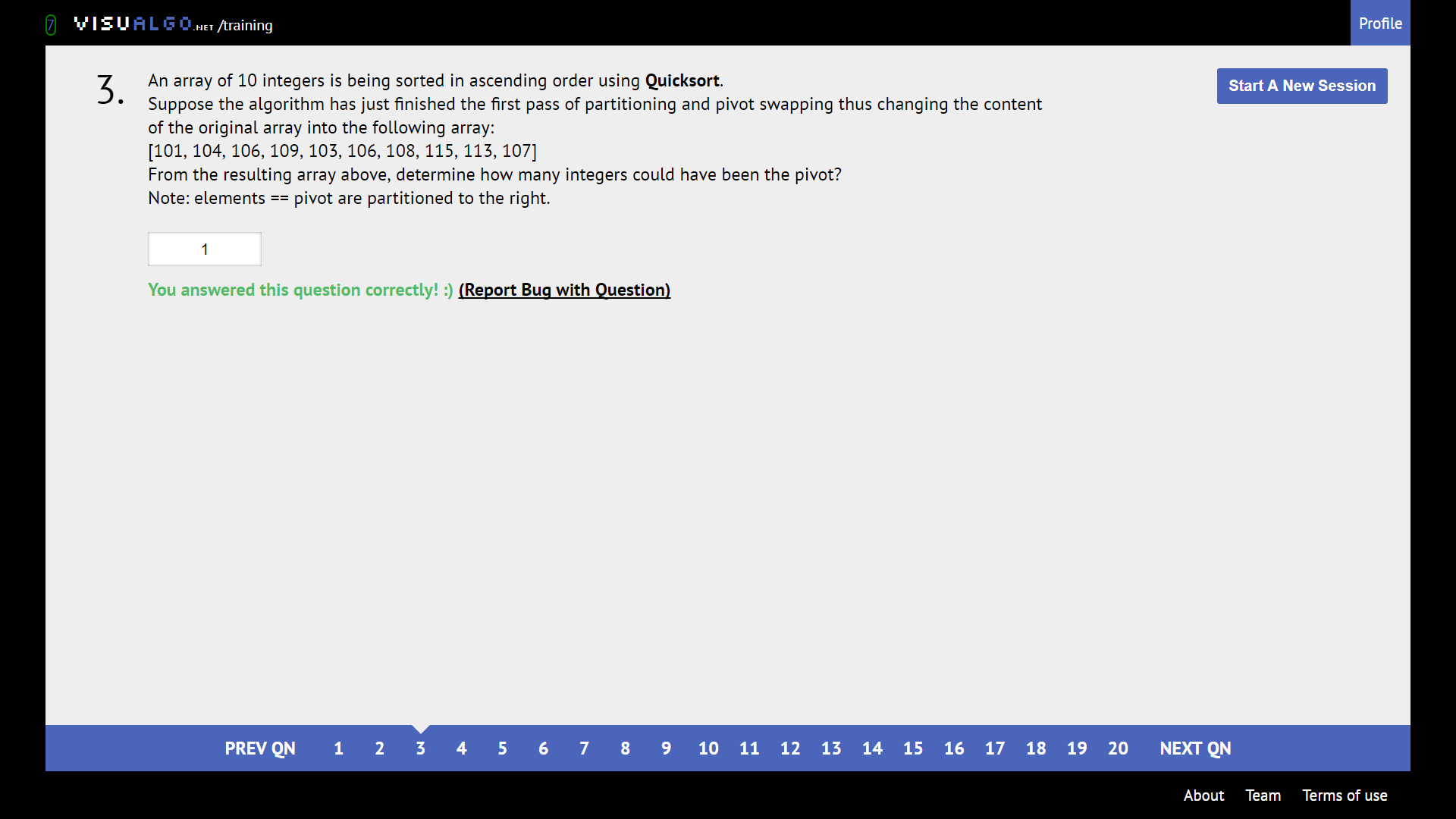
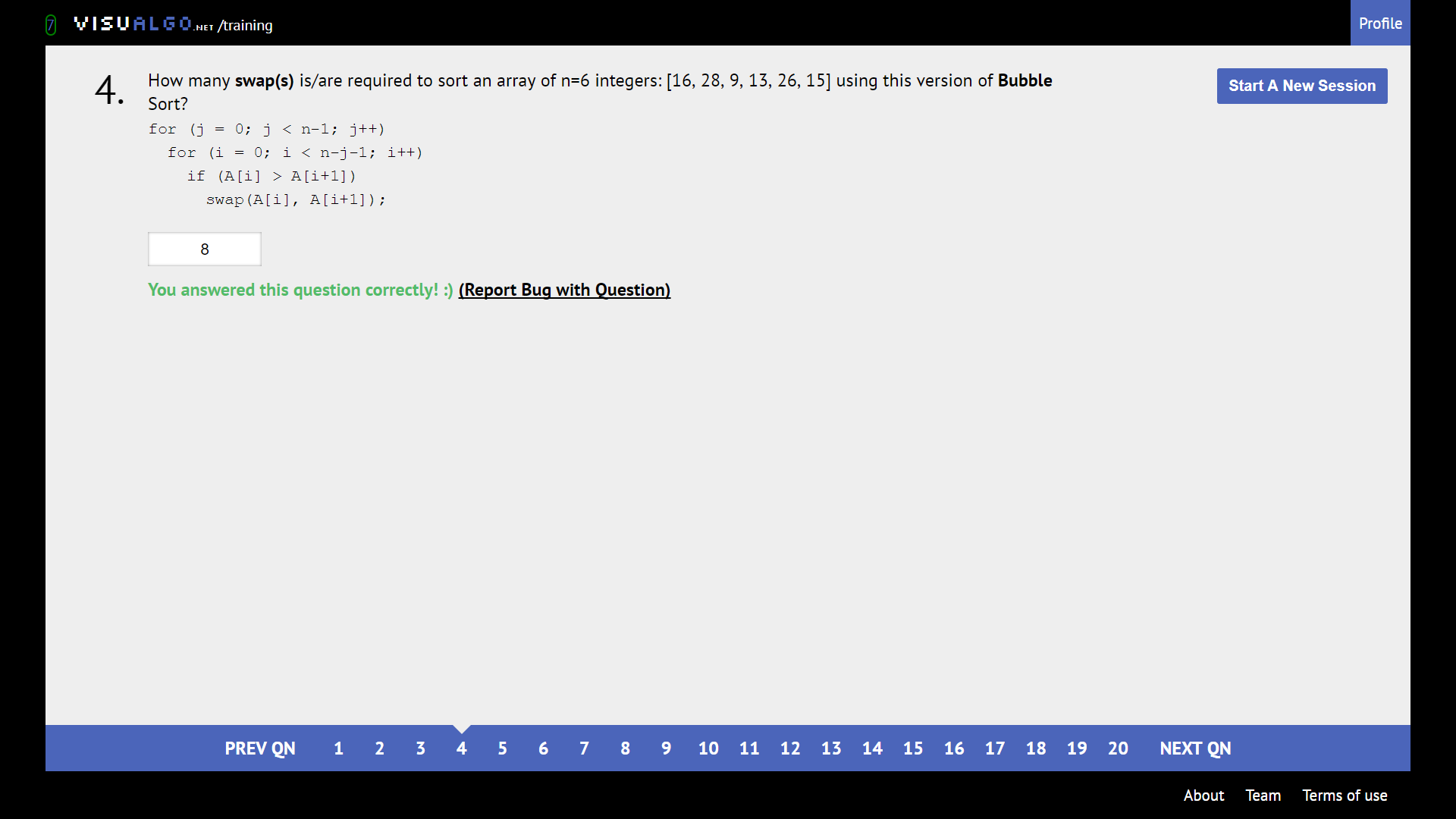
**Sorting**



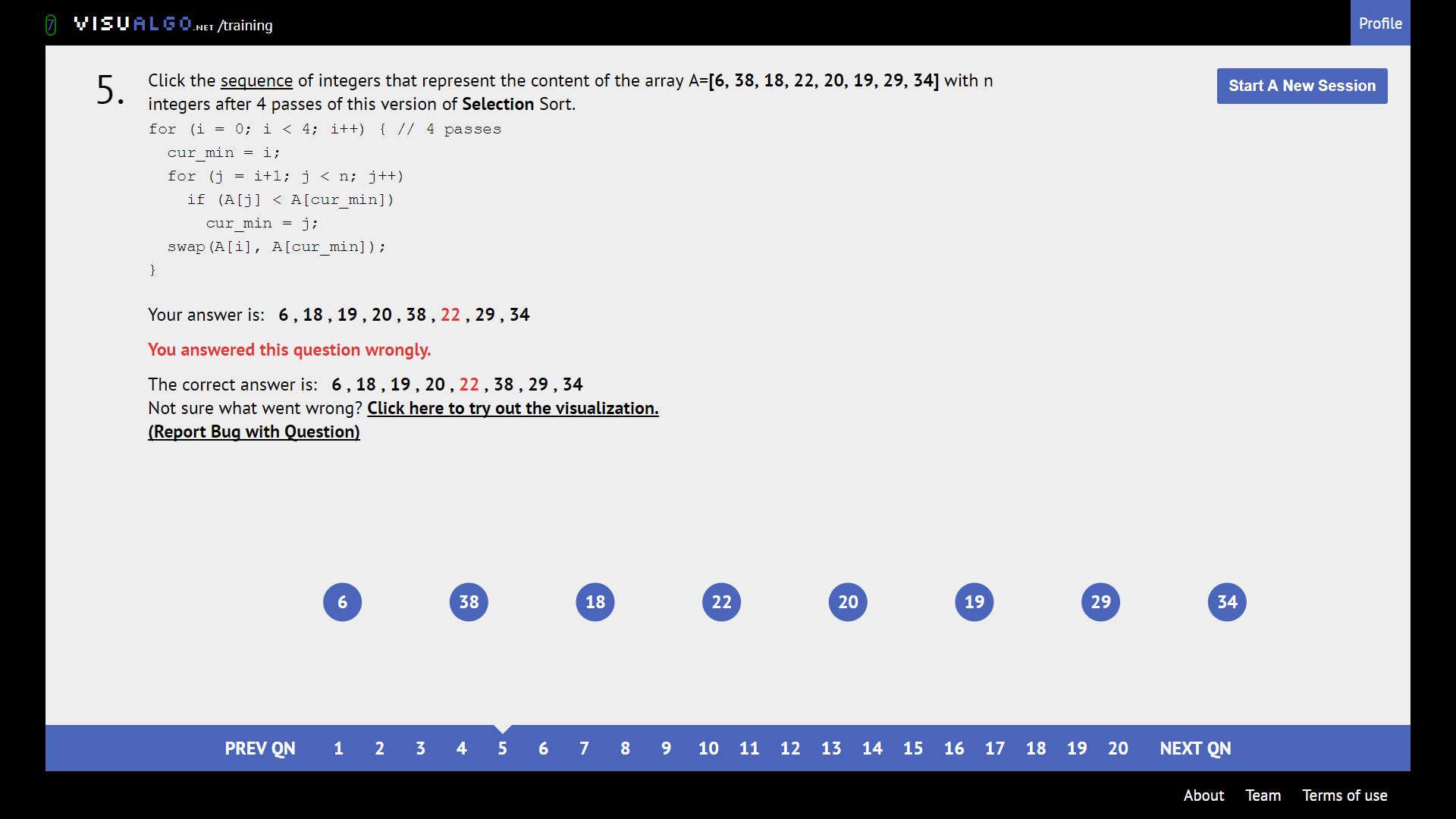
First n+1 numbers will be sorted. Insertion sort is STABLE, so the rest of the elements is unaffected.



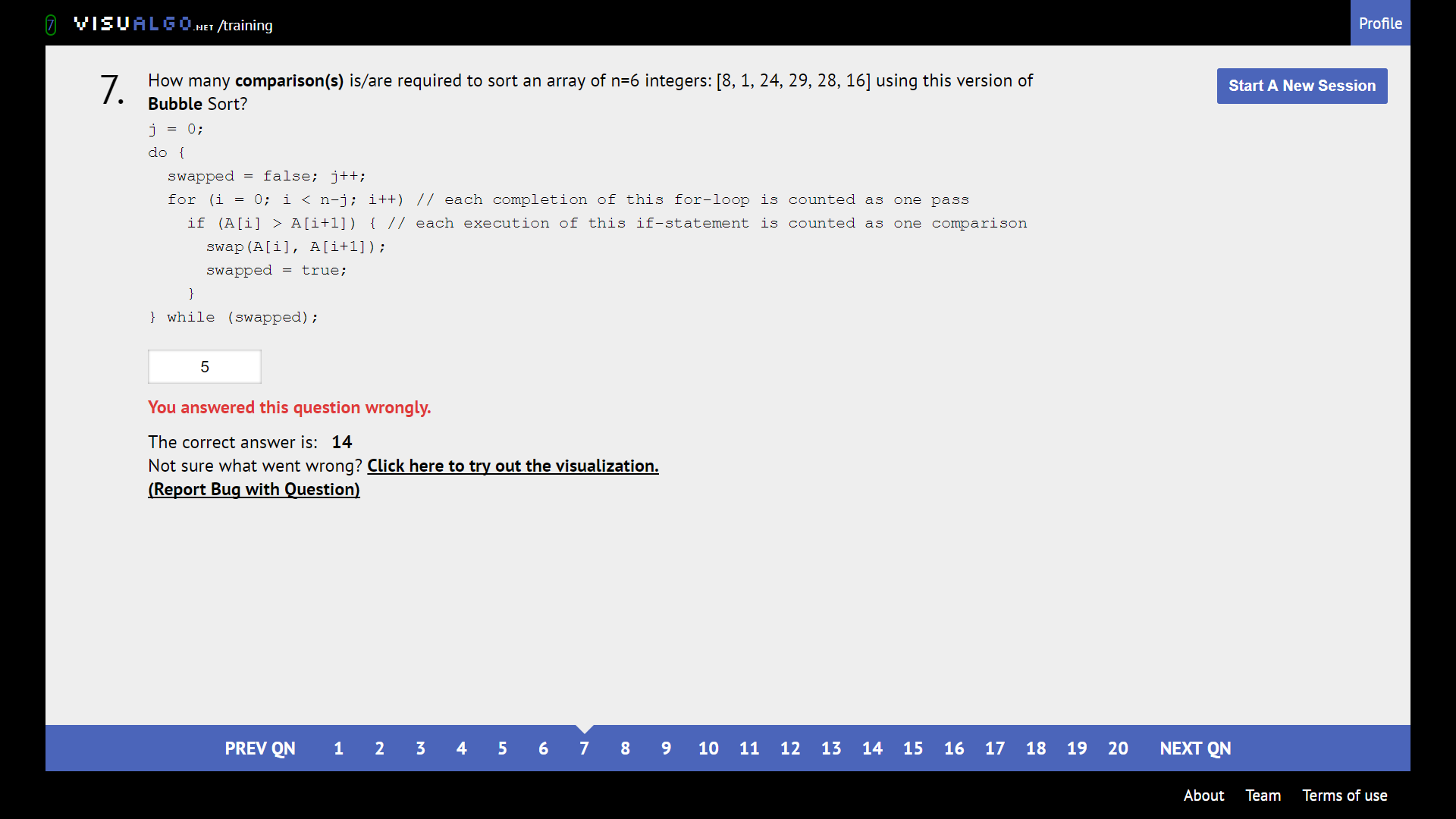
Find integer that will be greater than all to its left and smaller than all to its right

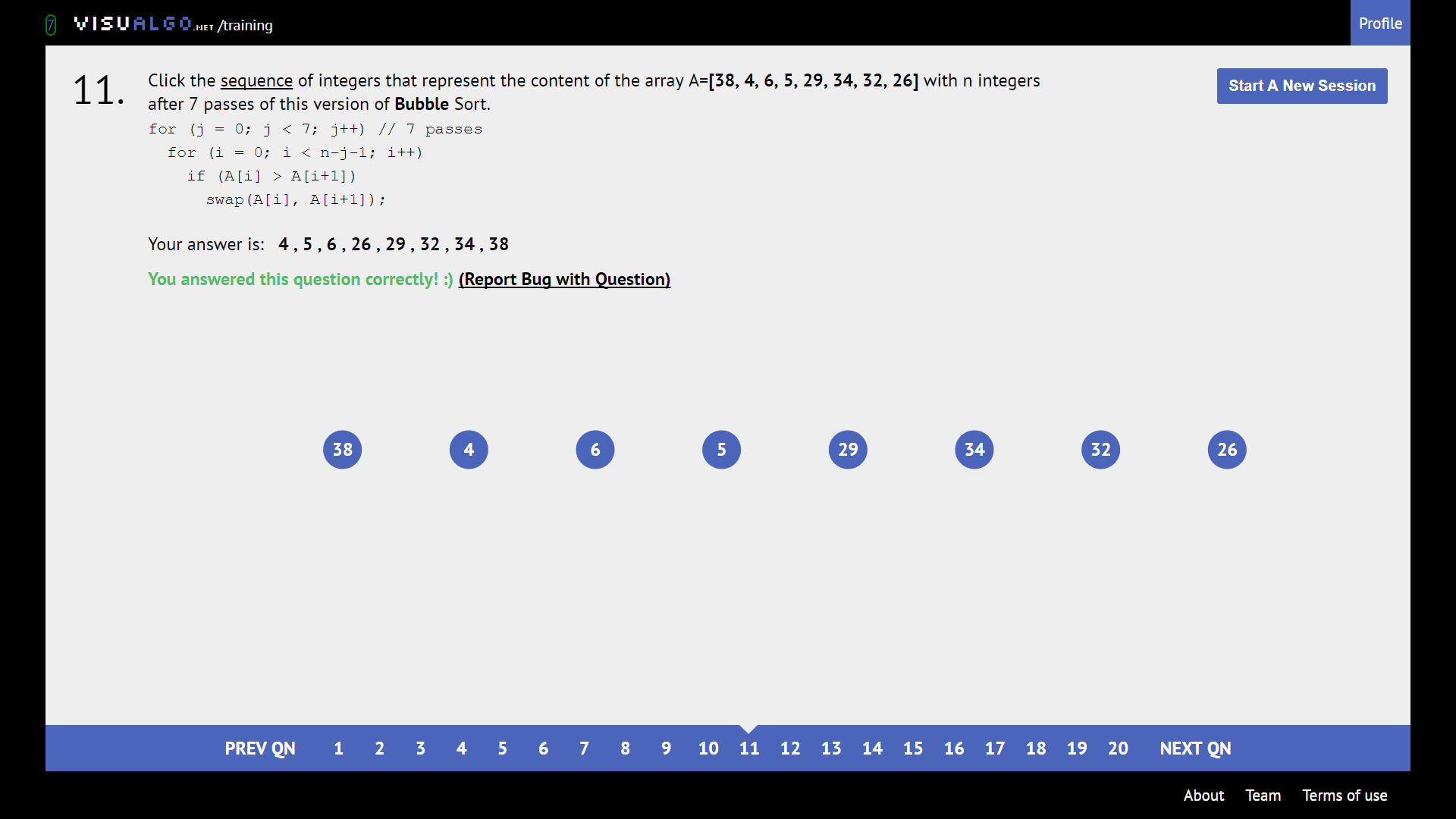


Brute force count swaps

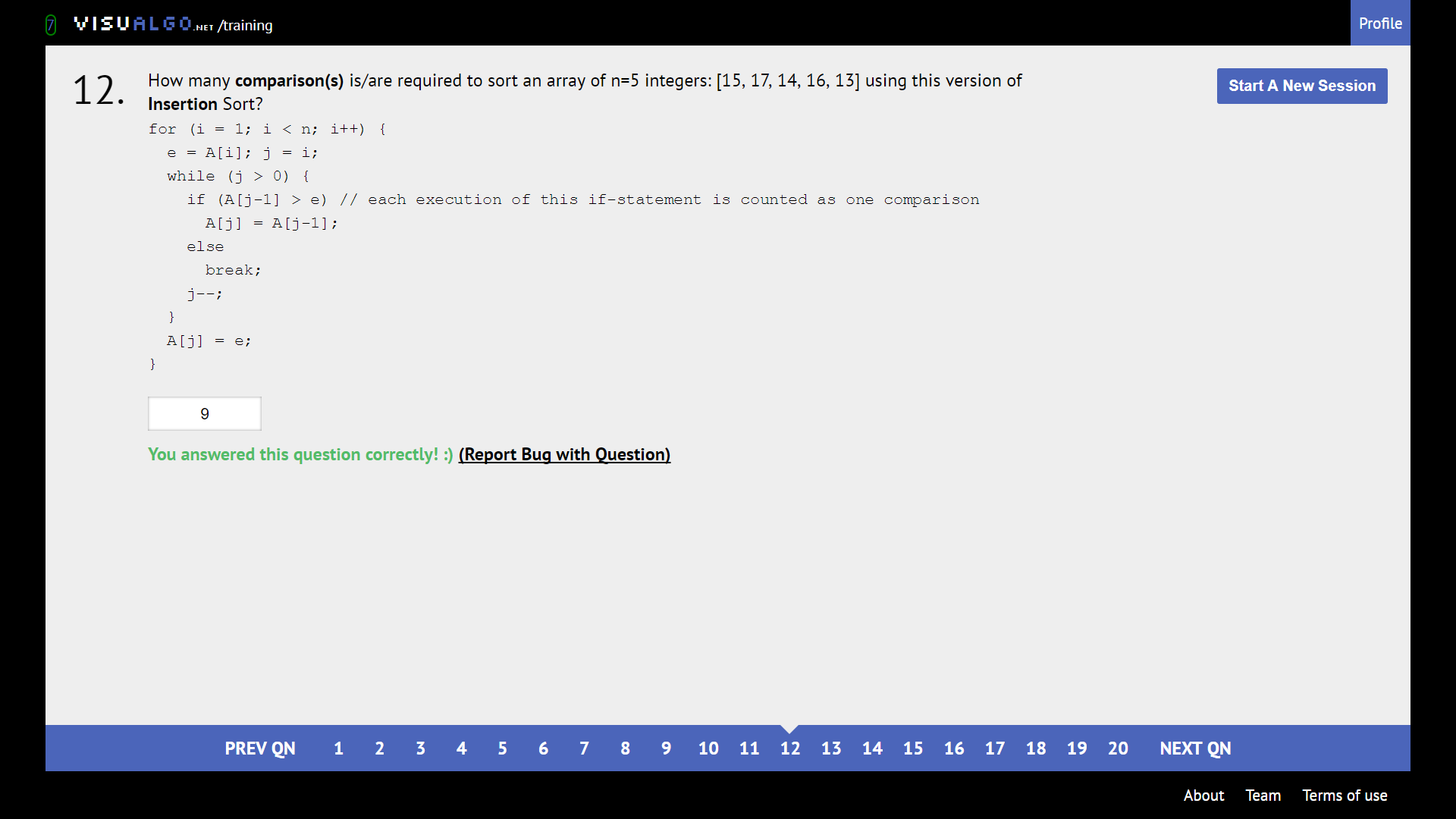


Selection sort is UNSTABLE. Keep track of where elements are swapped to.

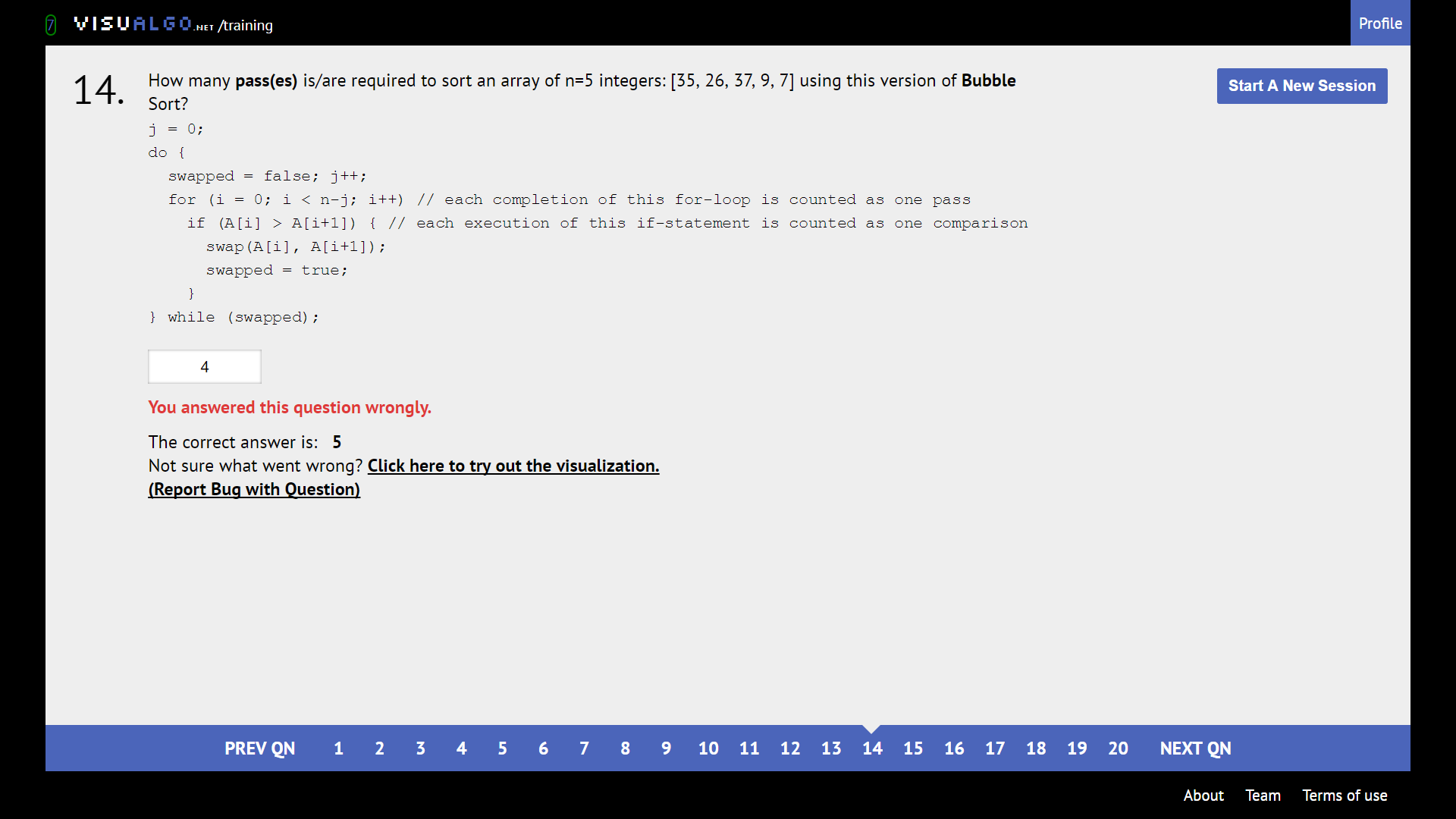




Biggest 7 elements are bubbled right. So, this sequence of 8 integers will be fully sorted lol

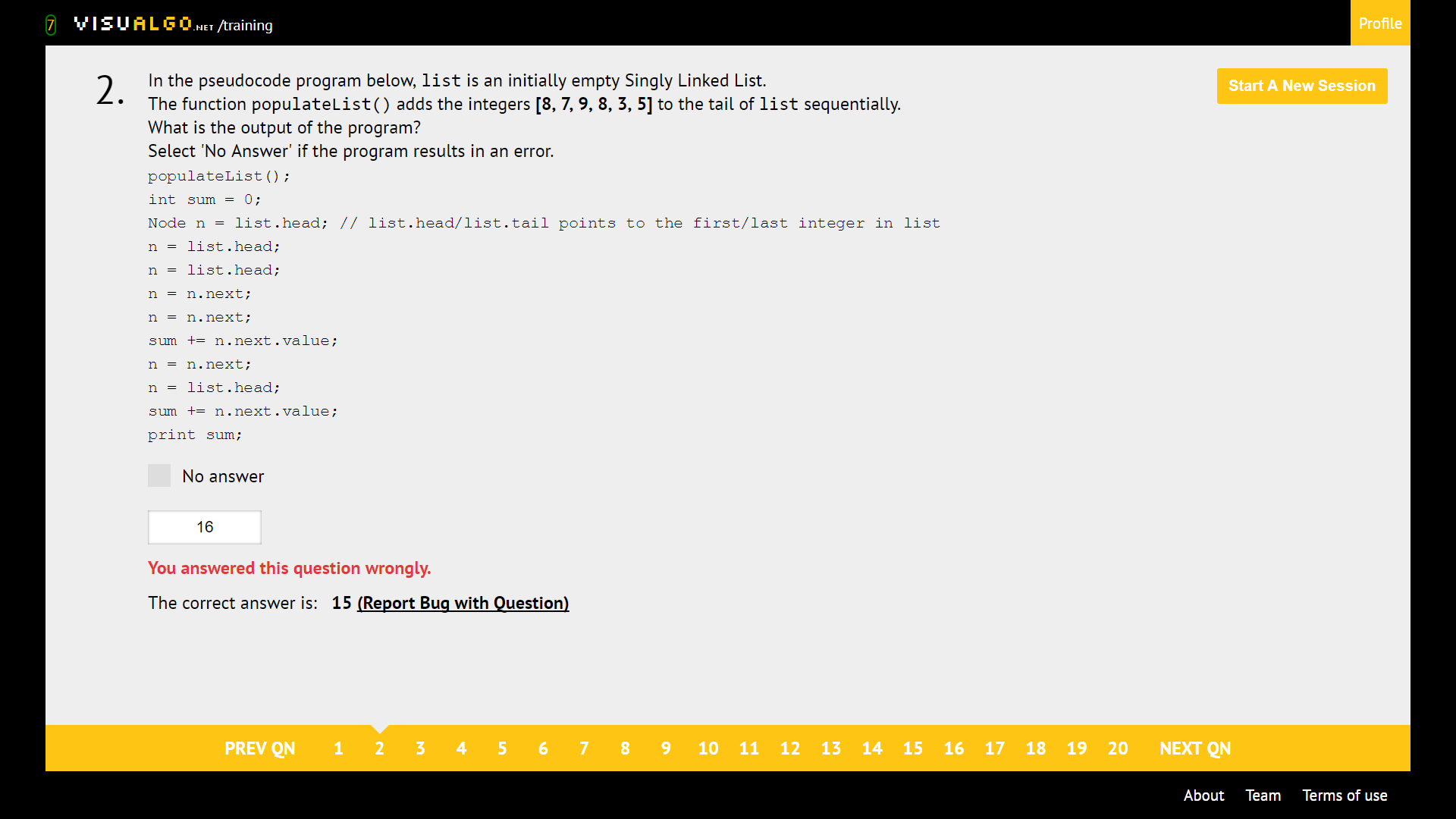


Become the algorithm

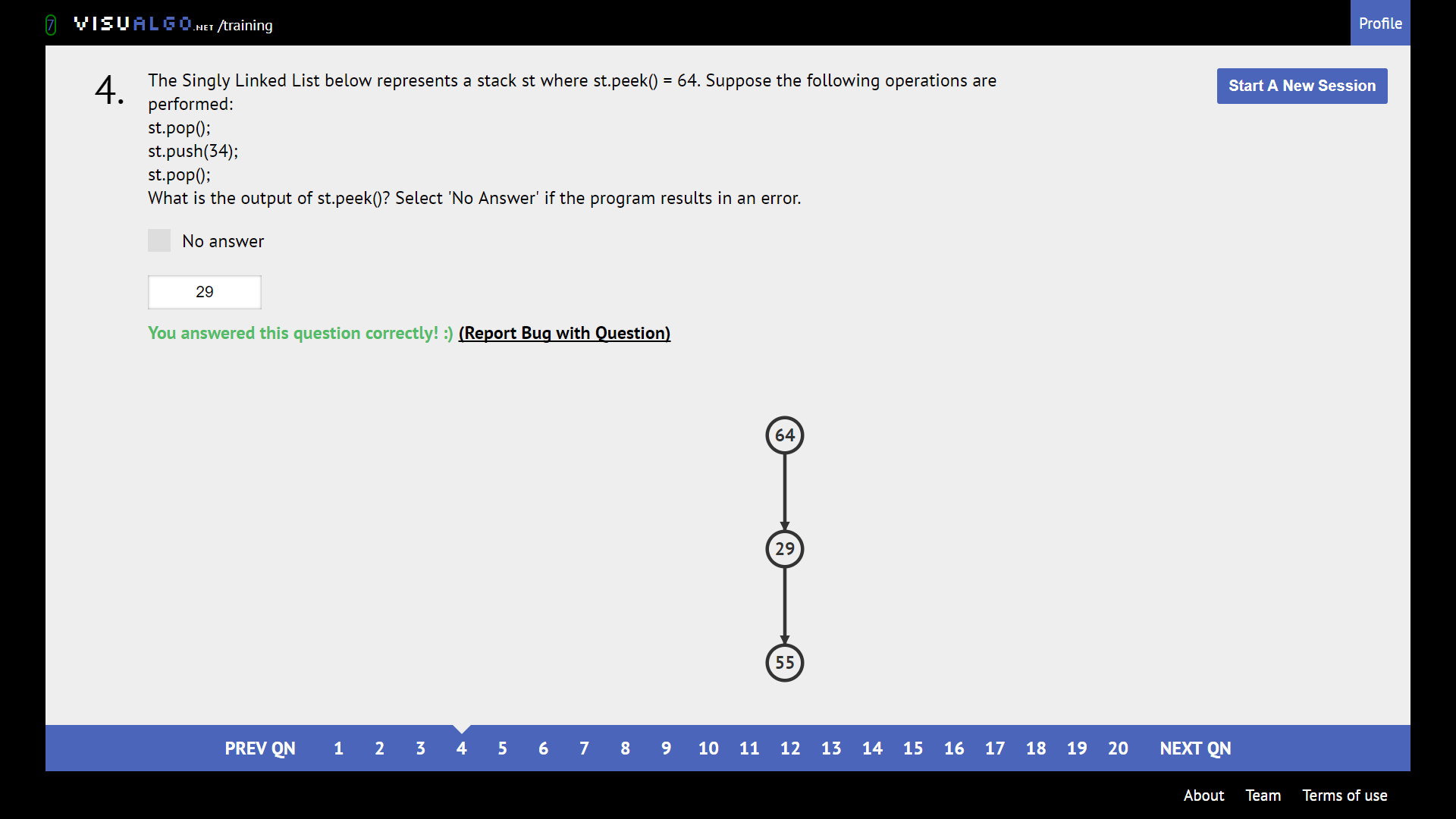


Become the algorithm. This is the “smart” version which breaks if no new swaps are made in the pass-through.

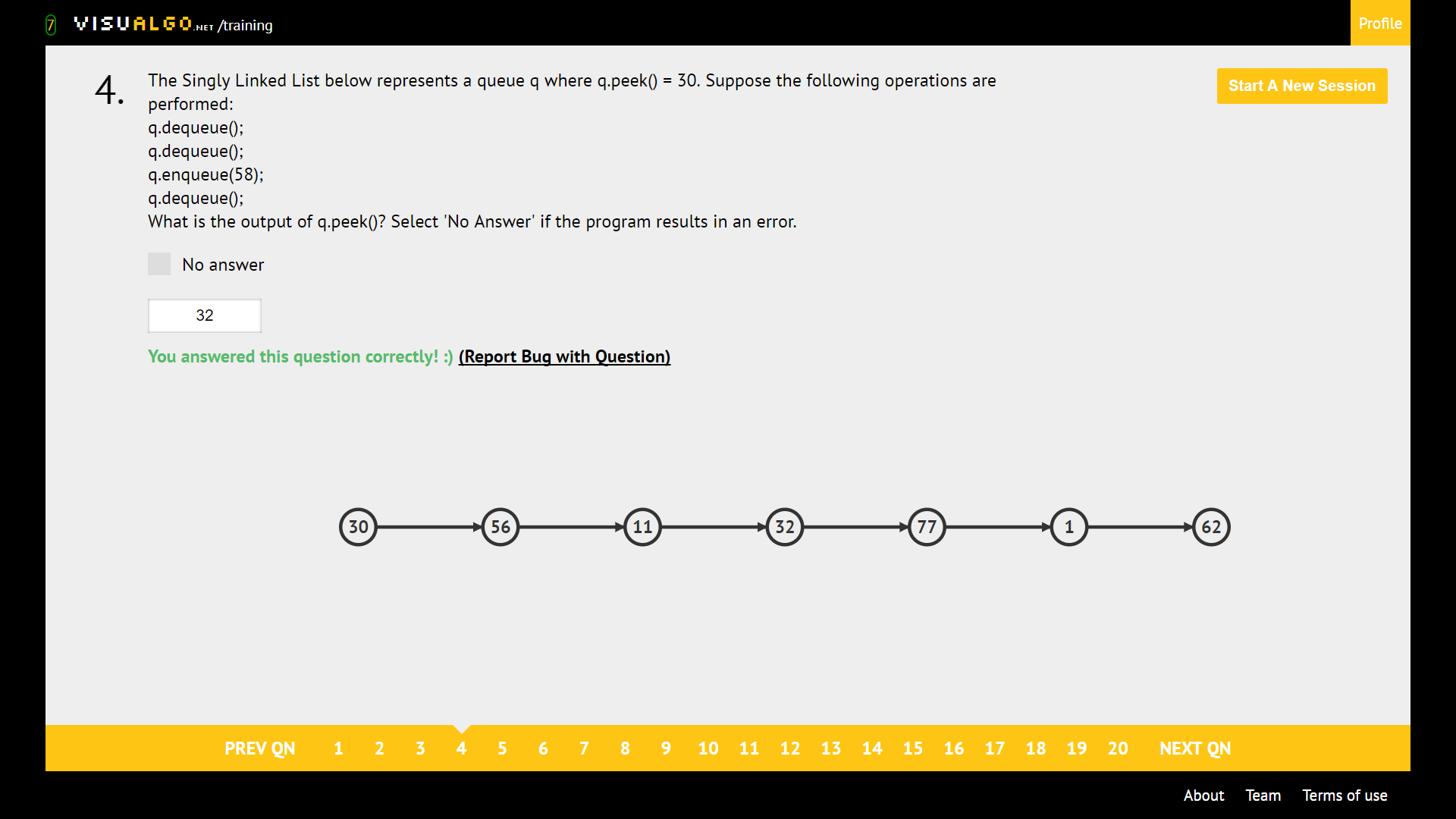
**Linked list**

****

Just be careful. Tail pointer is usually not updated.



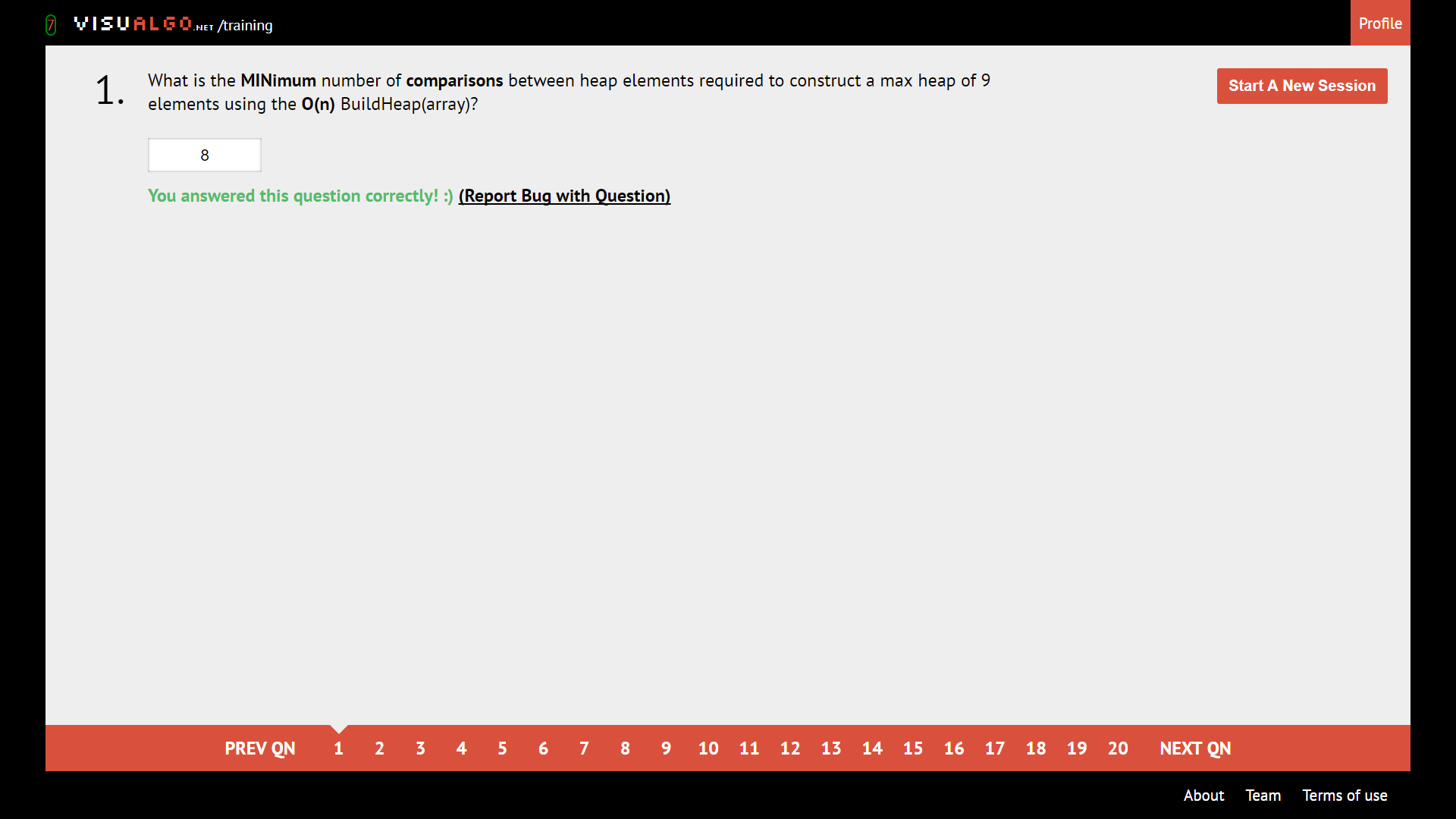
Add and remove stuff from the TOP, just like tray stacking



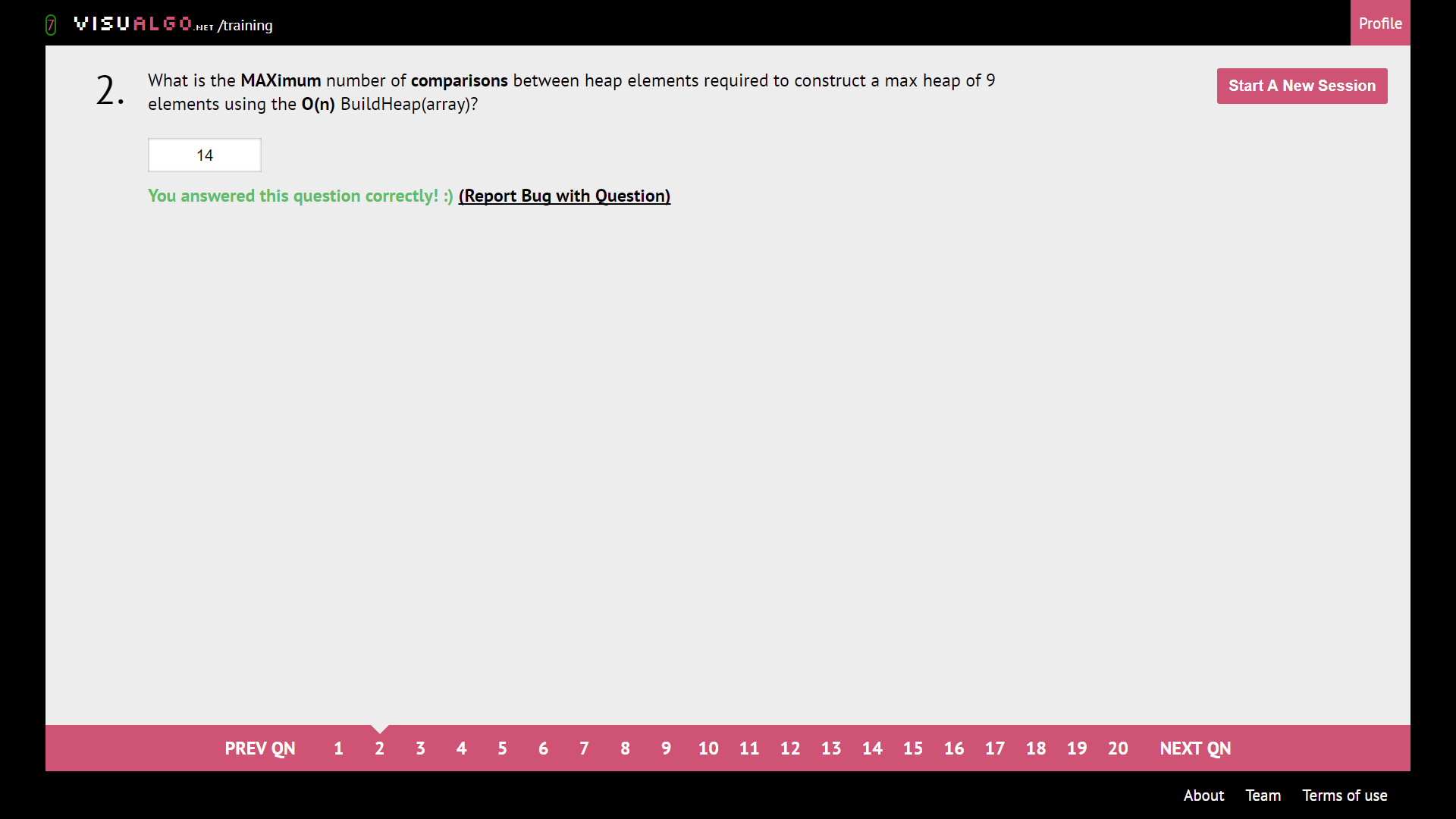
Enqueue adds at the RIGHT, removes from the LEFT

**Binary Heap**

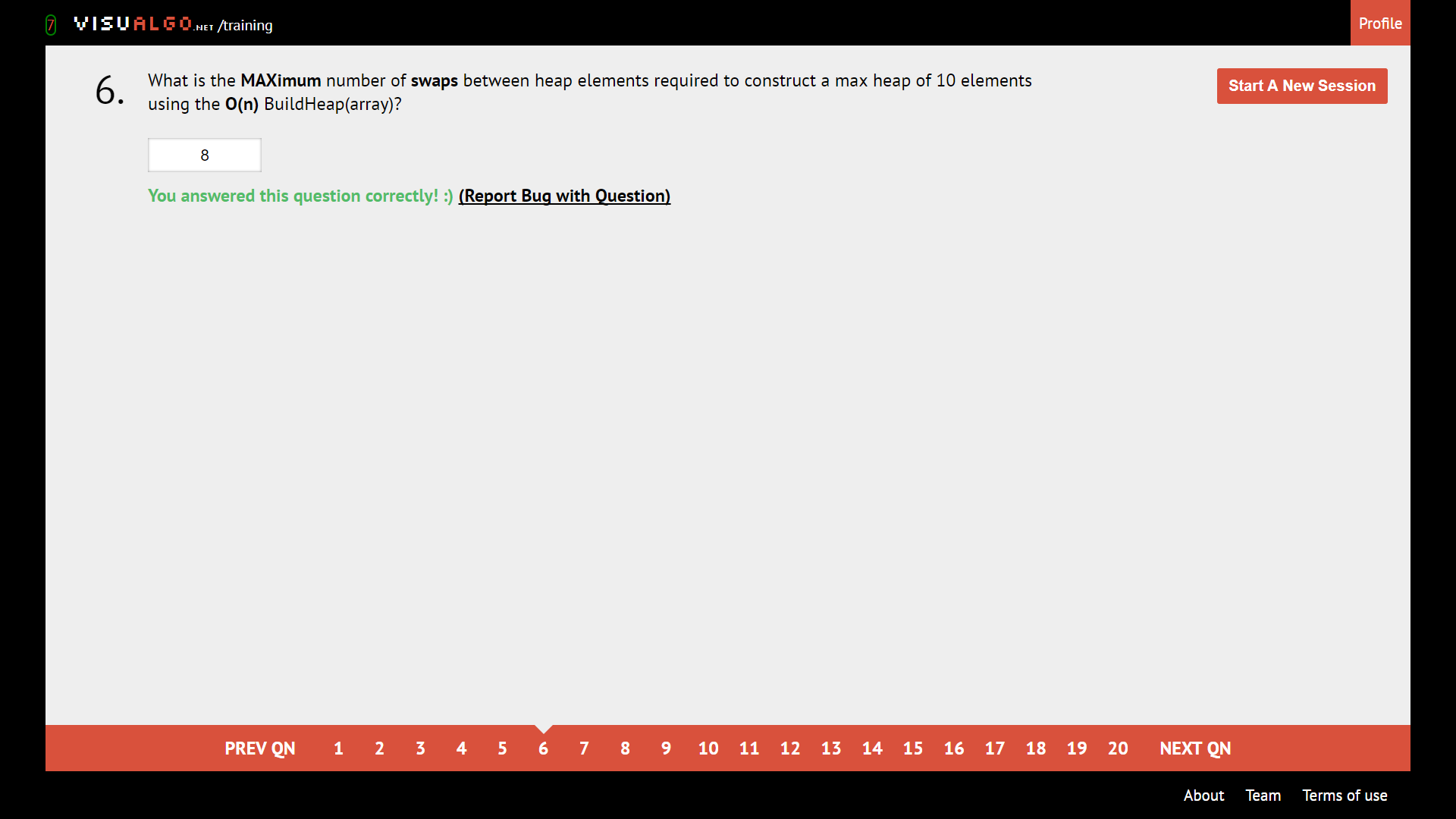
Valid Max Heap: Make sure parent bigger than child



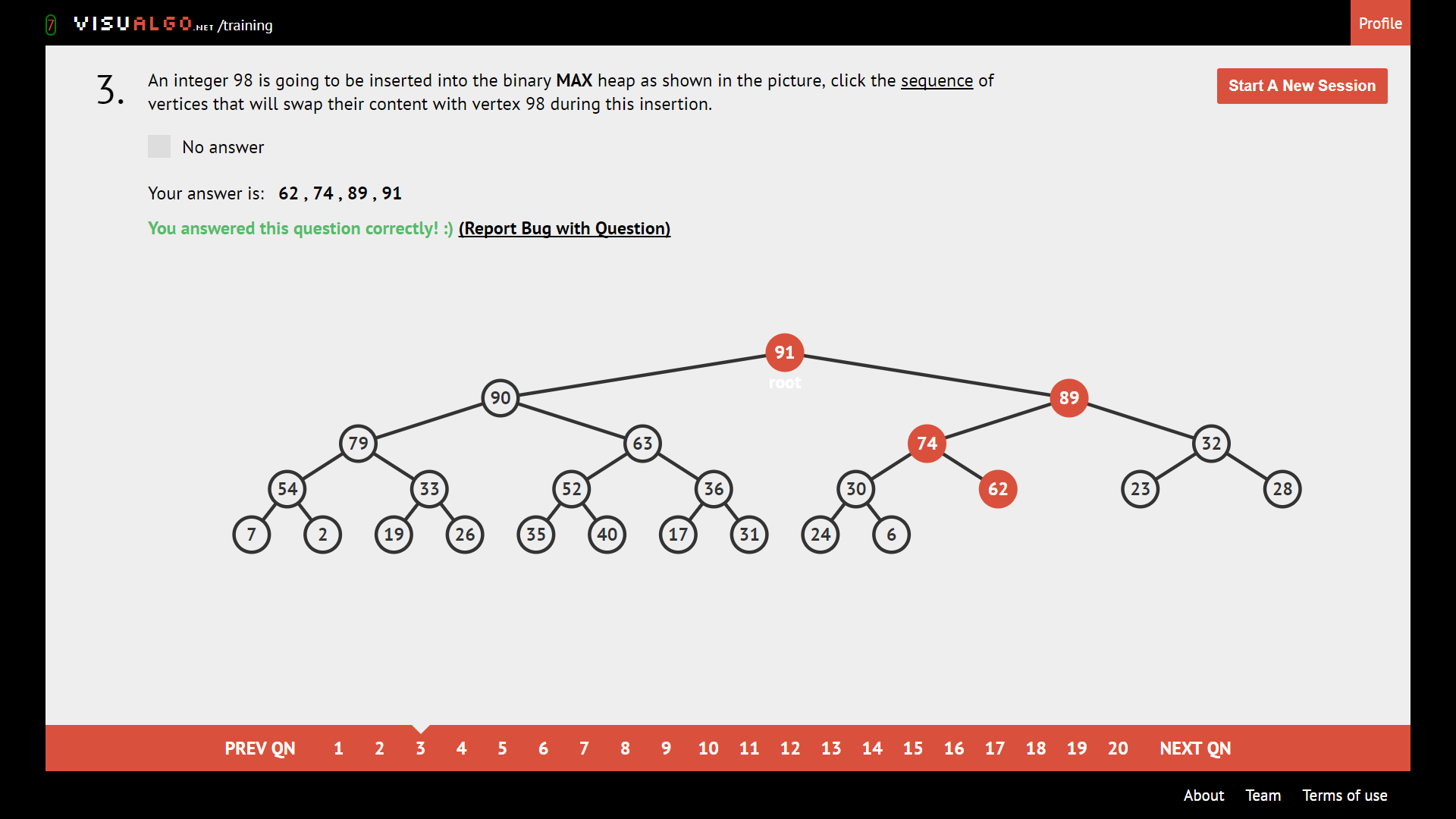
Refer to chart



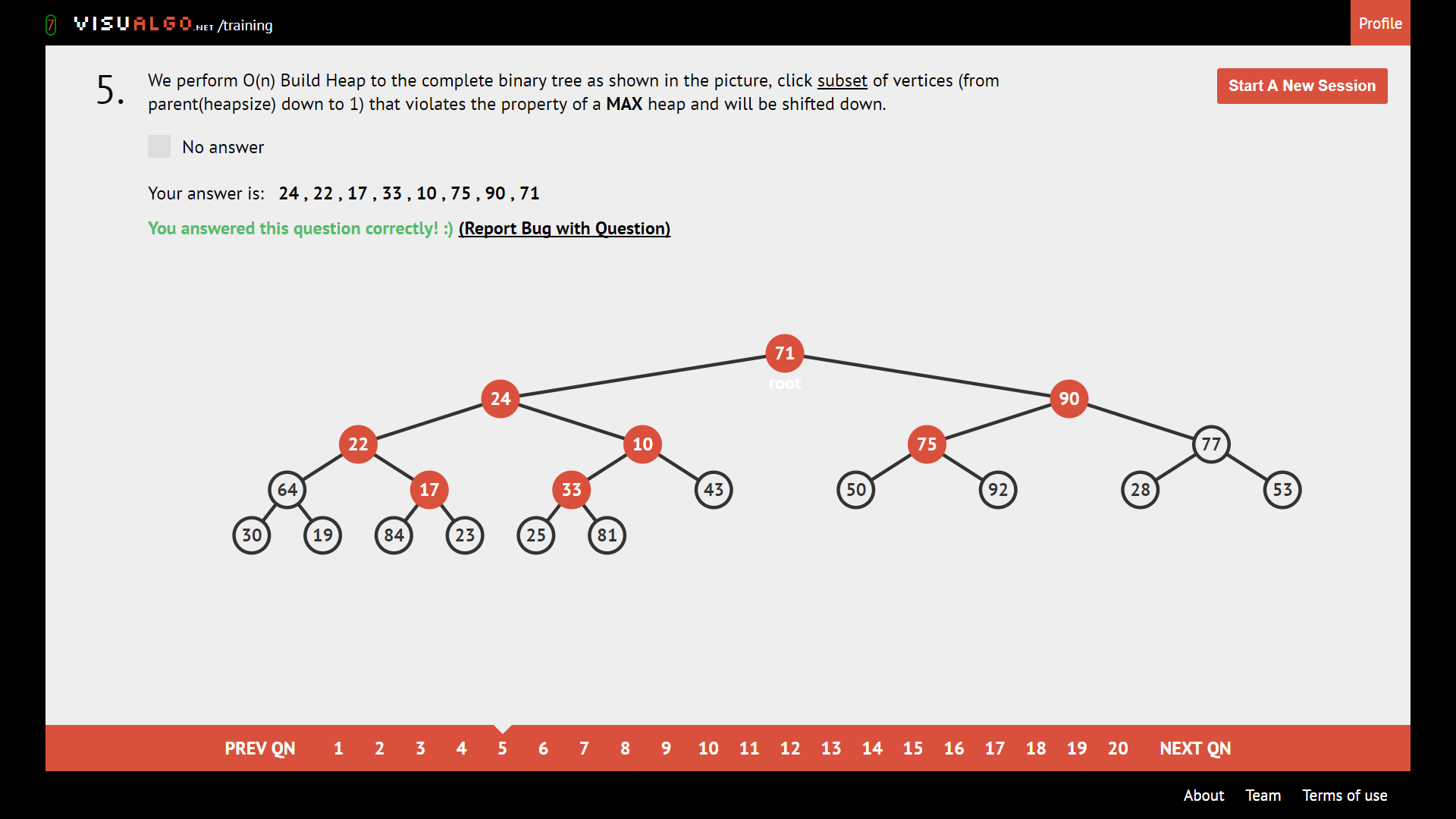
Refer to chart



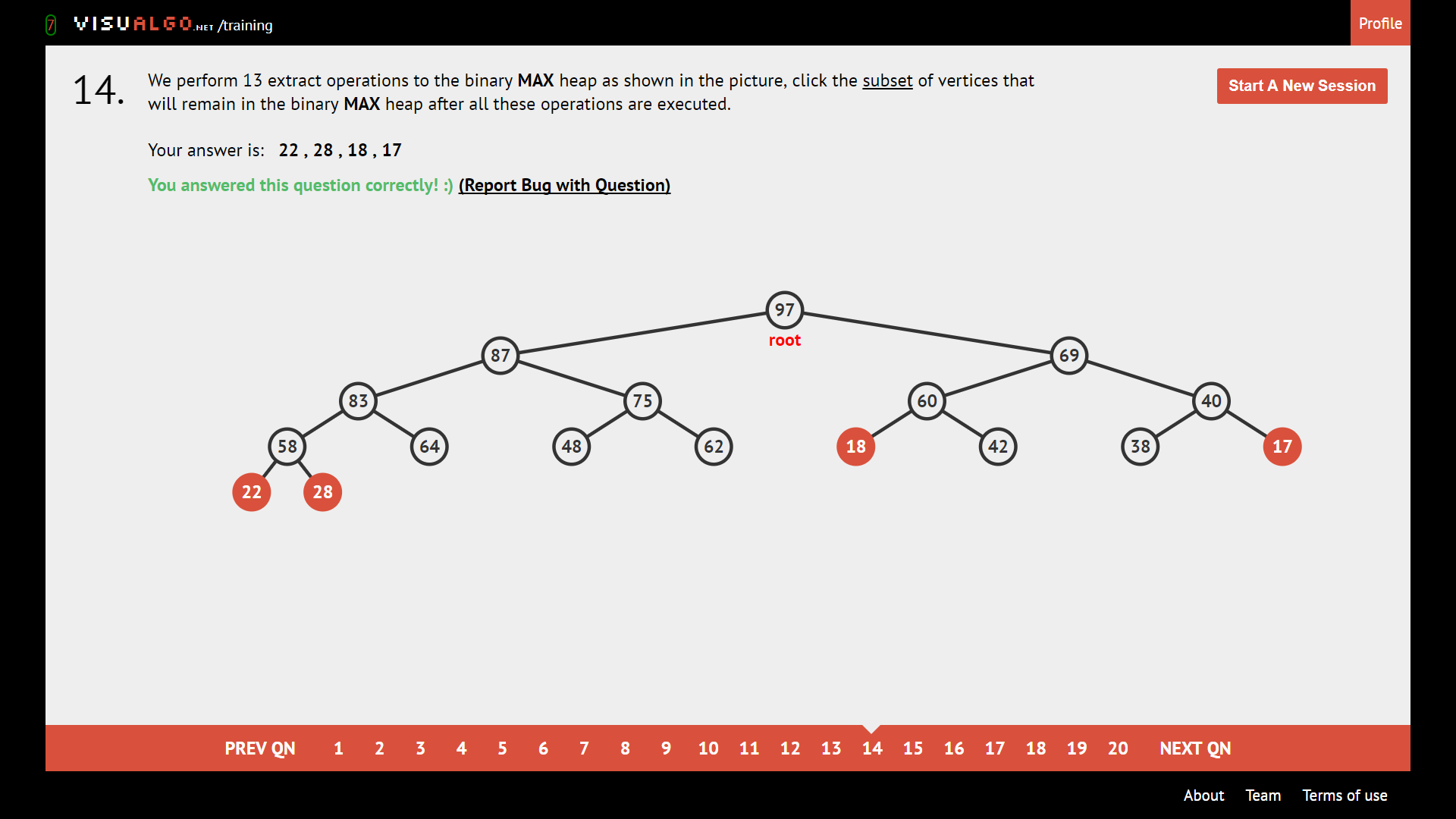
Refer to chart.



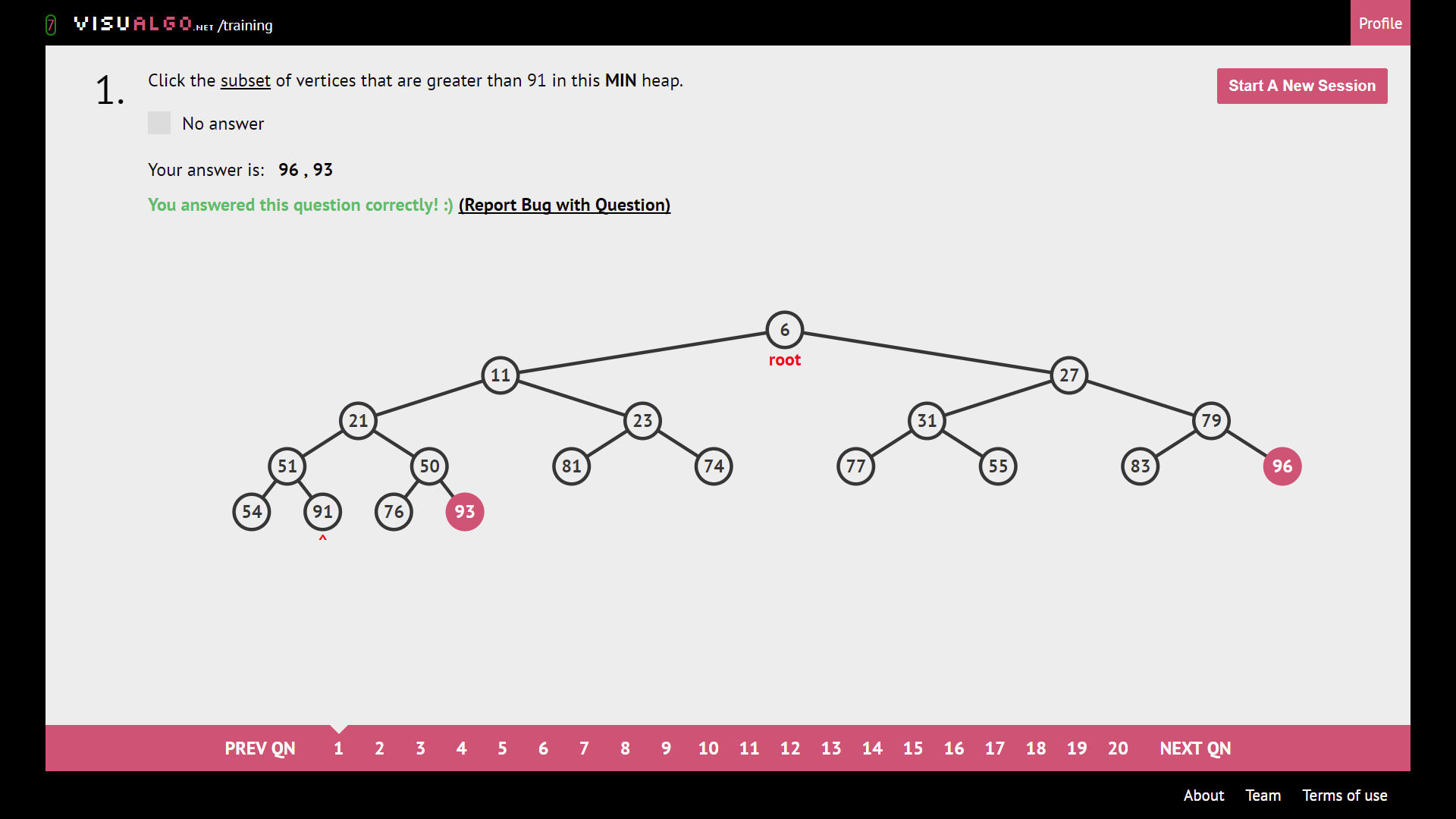
Add to bottom-most and left-most available spot. Bubble it up.



Vertices that are smaller than any of their descendants will be shifted down.



Smallest 17 – 13 = 4 elements are remaining



Find all vertex bigger than number. This q is dumb.

**BST**

BST: Left child < current < Right child.

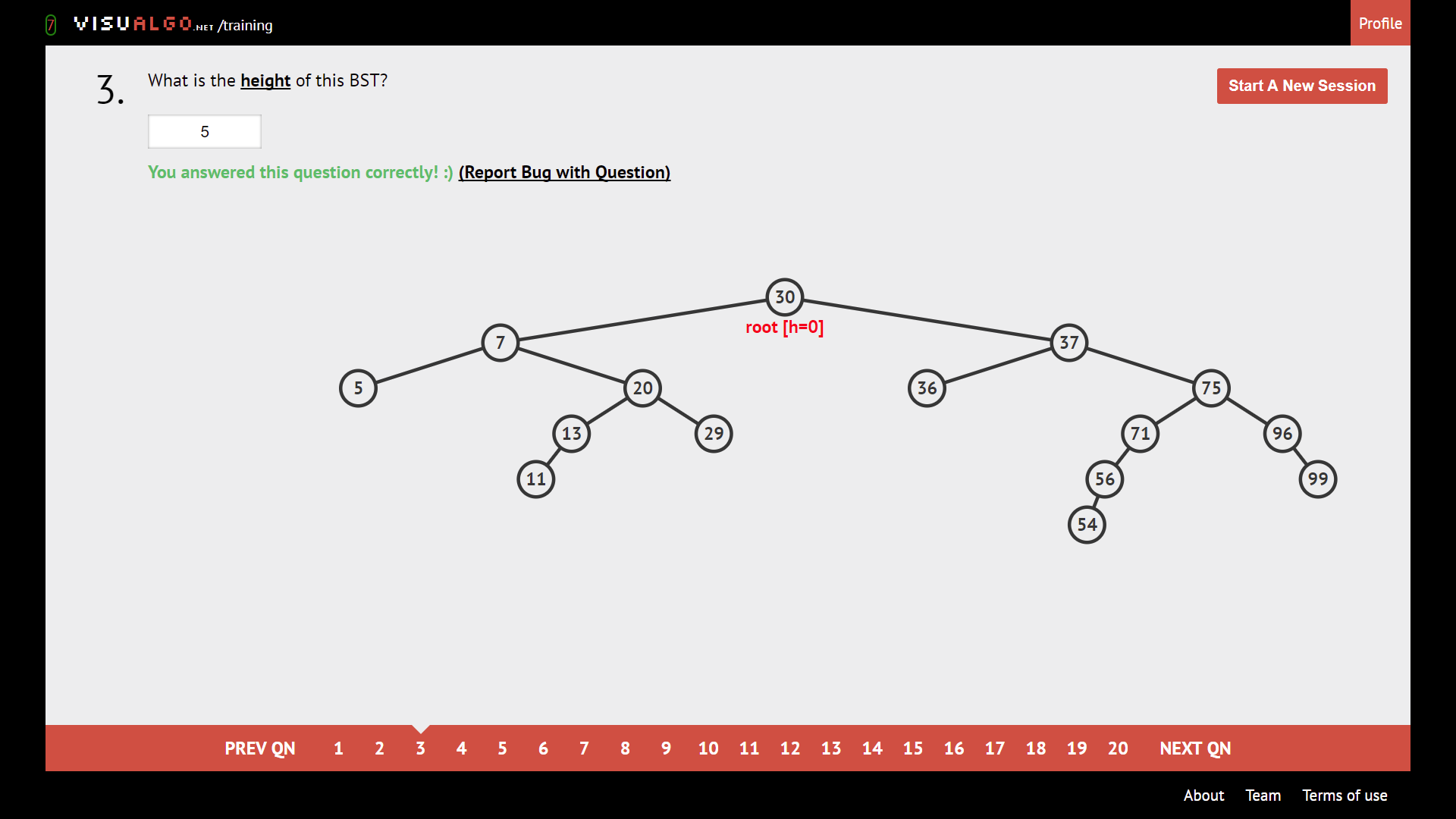
Successor: Most direct way to the next element bigger than the node of interest

Predecessor: Most direct way to the next element smaller than the node of interest

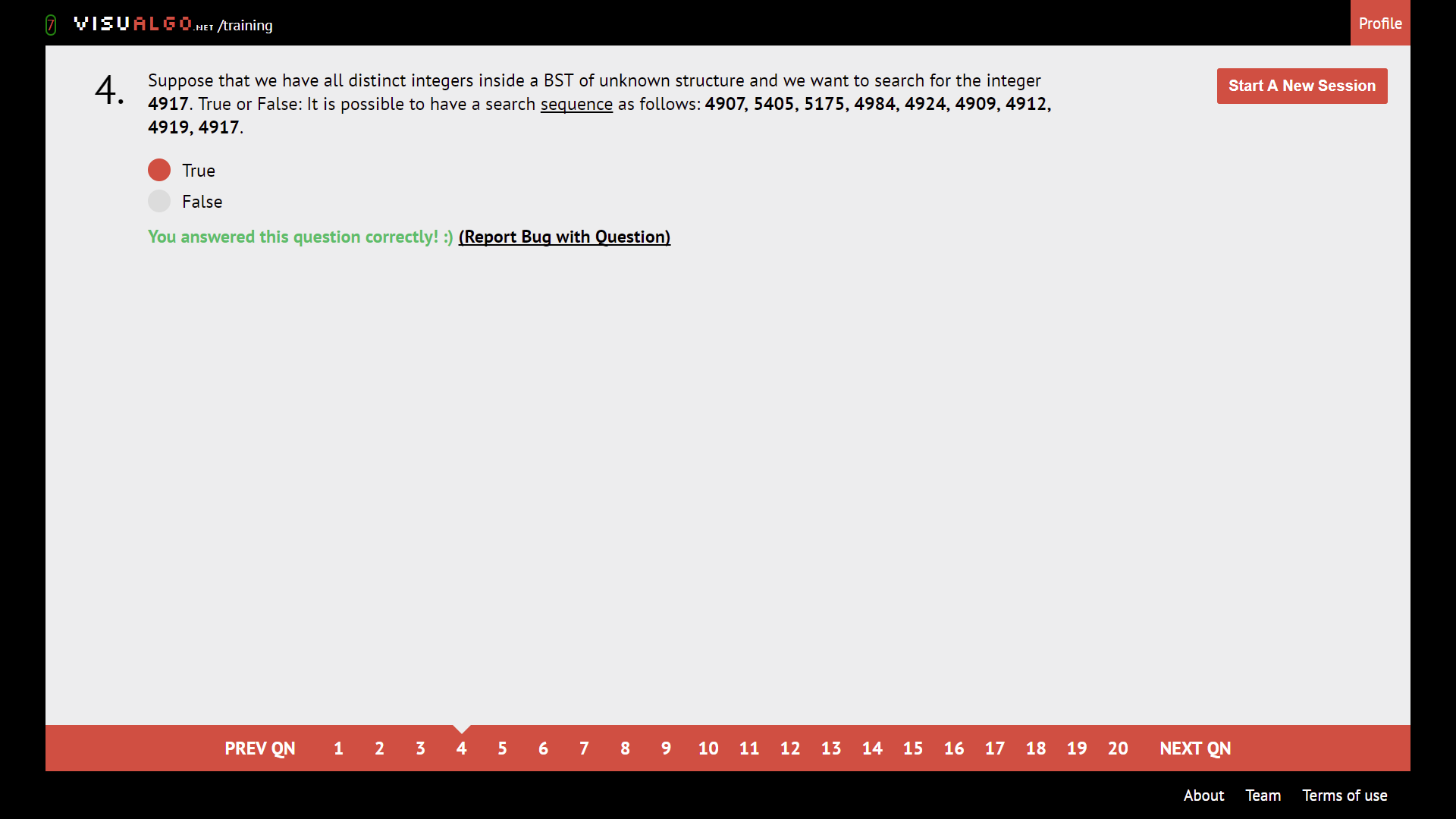
**Pre**order: **Print first**, then check children

**In**order: Check left, **then print**, then check right

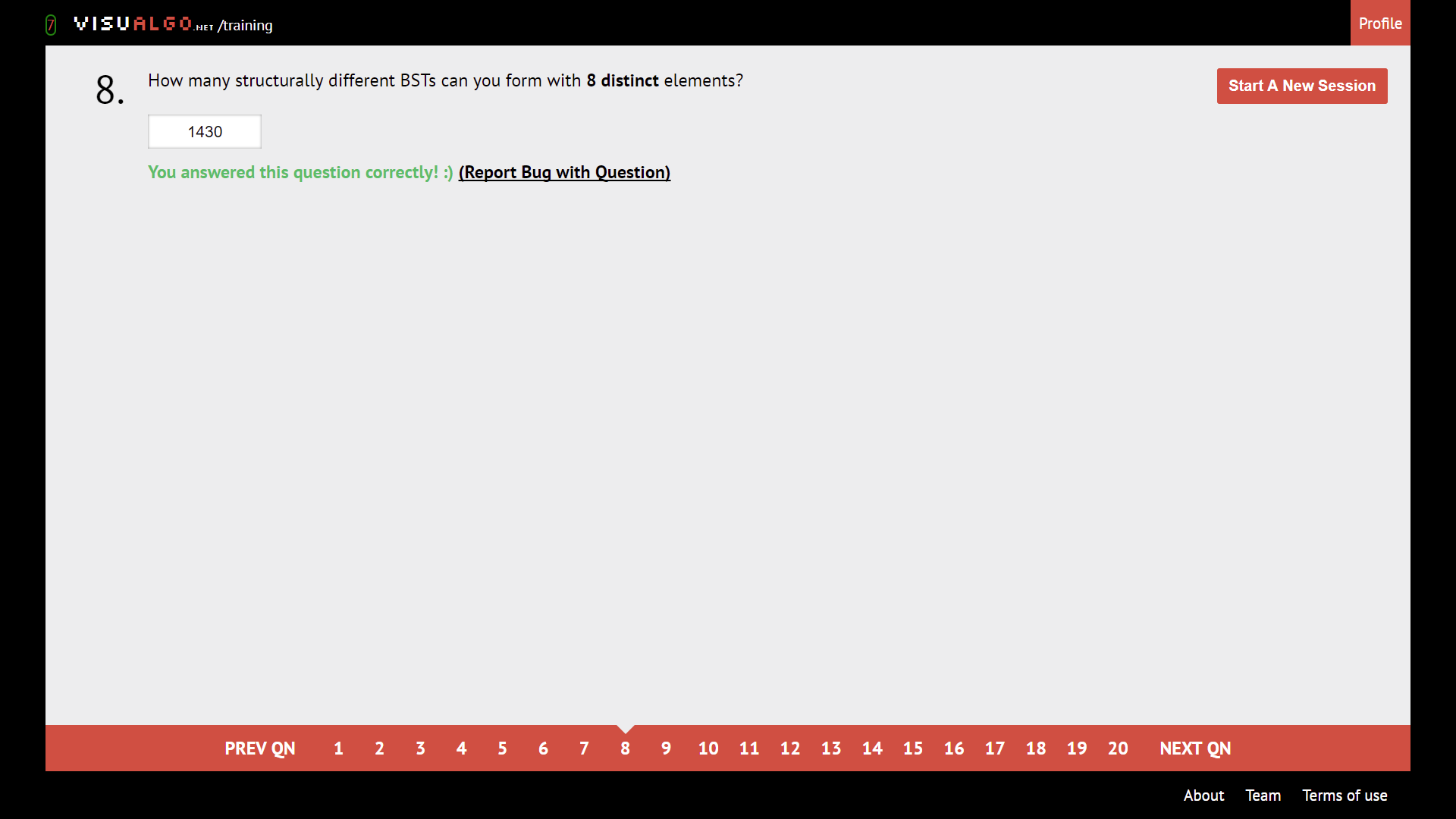
**Post**order: Check children first, **then print.**

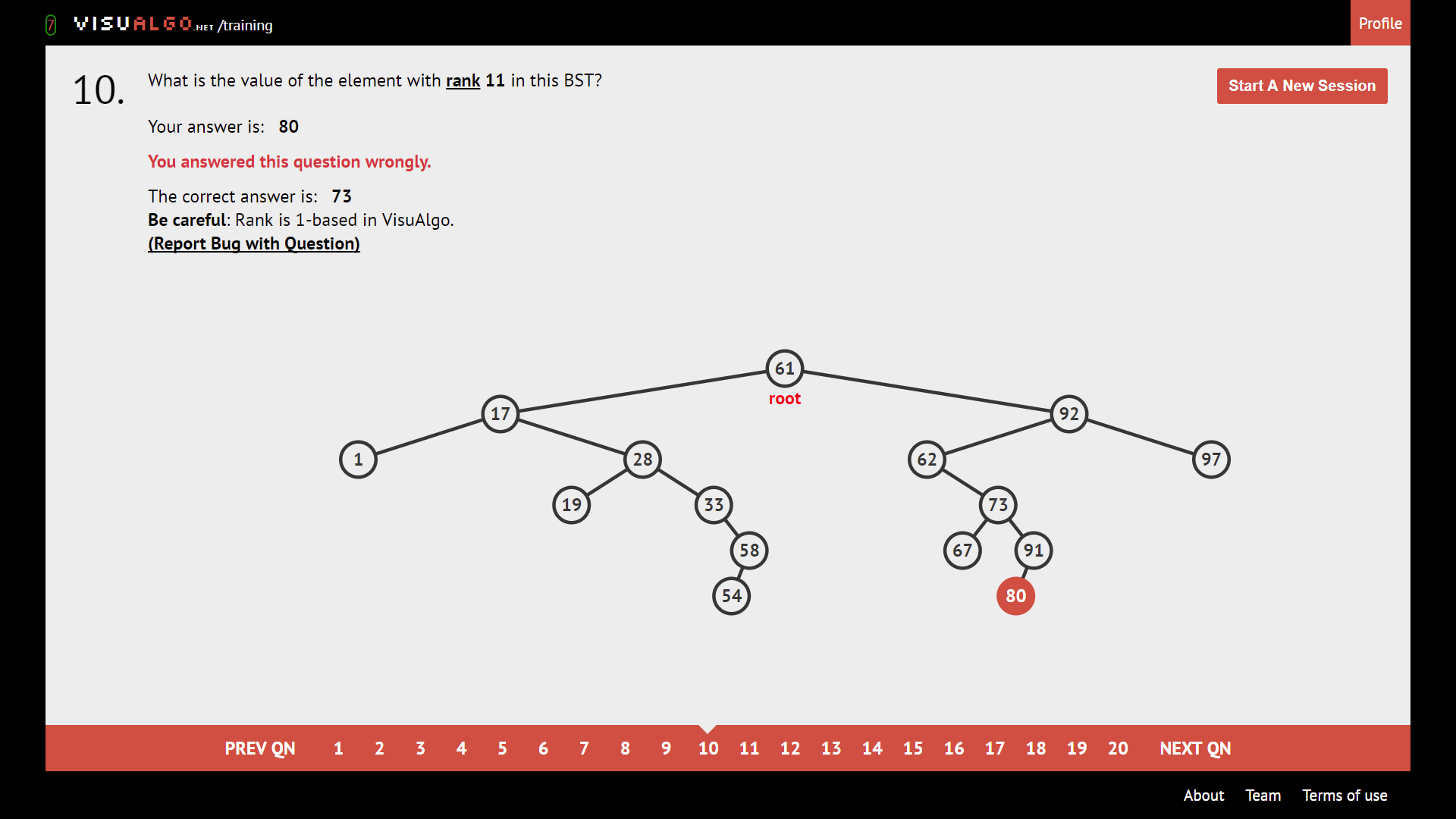


Largest number of edges from root to leaf.



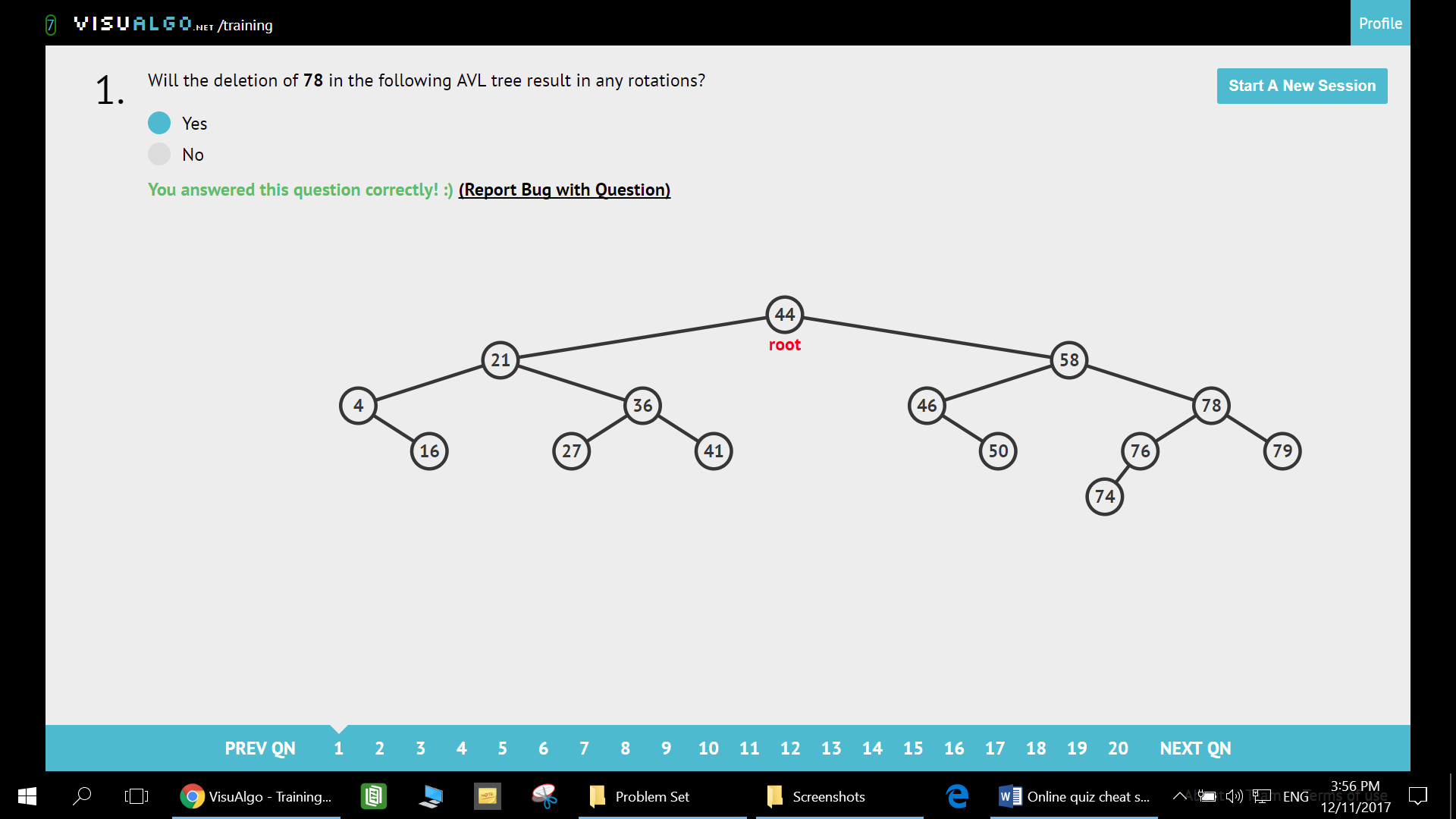
If target is bigger, next number should be bigger; if target smaller, next number should be smaller.

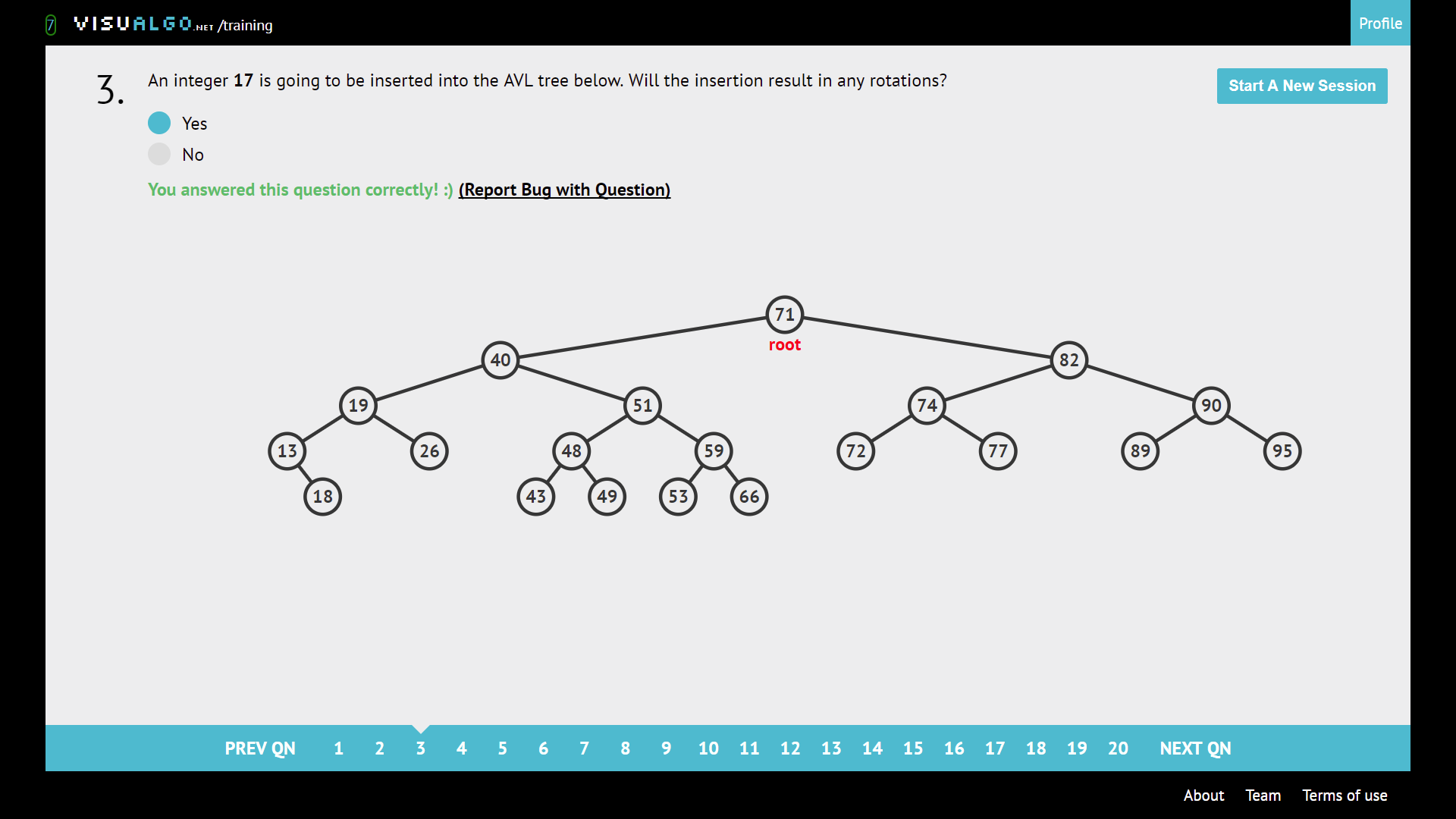




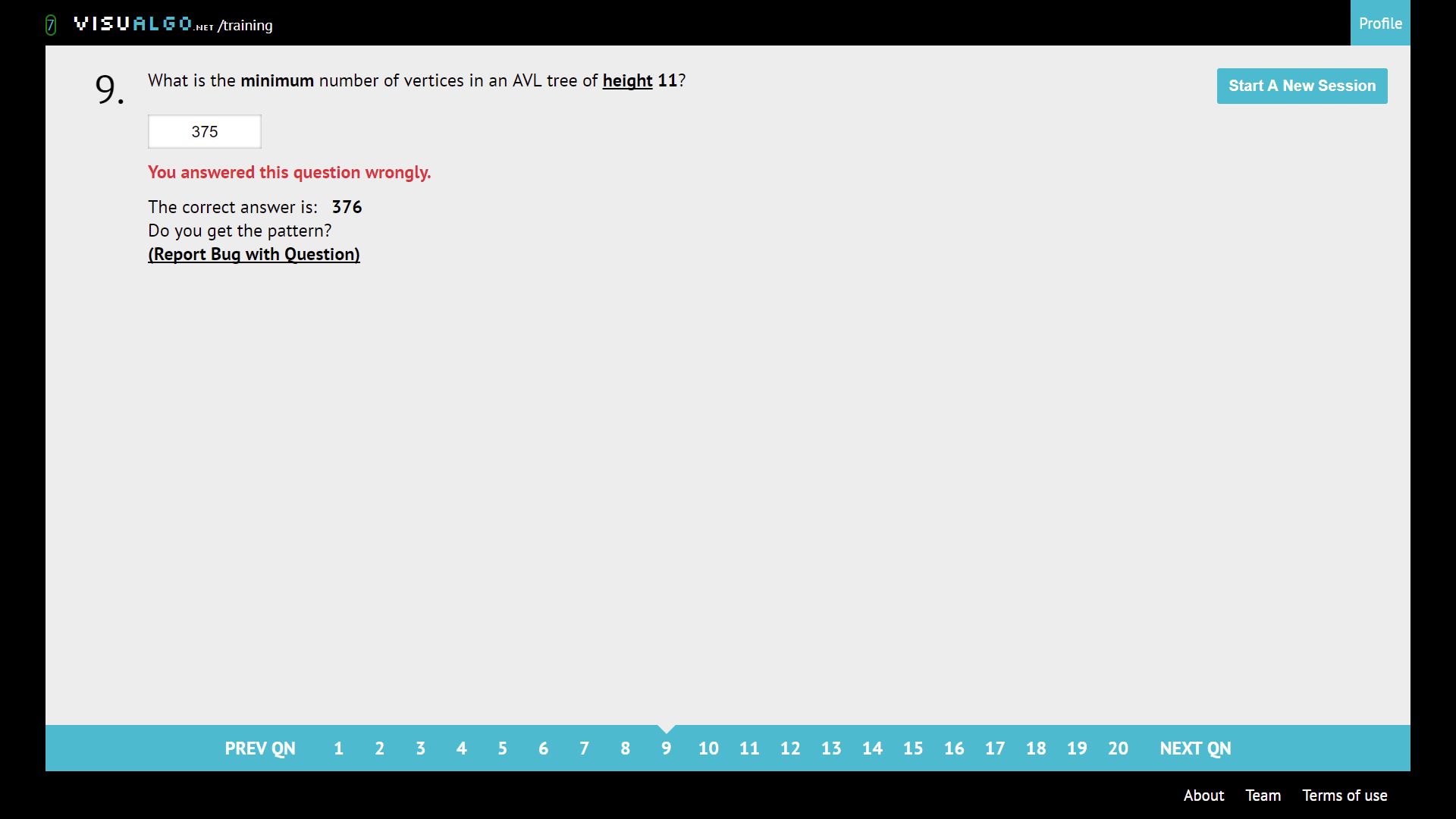
1-indexed. Leftmost is 1, count until rank.

**AVL**



Search upwards starting from the replaced node and see if anyone complains

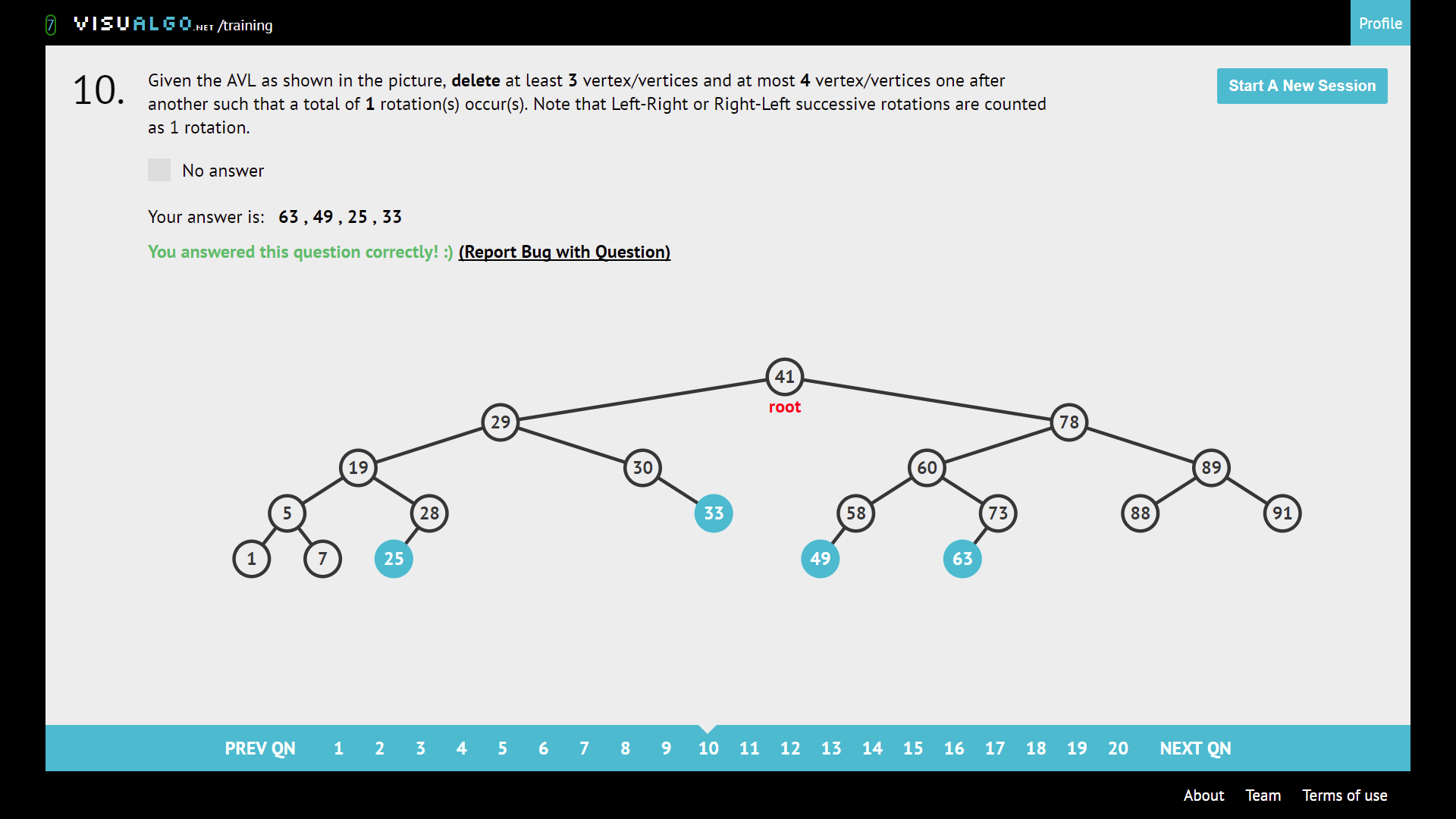
Insert into correct position and check for rotations.



Height(n) = height(n-1) + height(n-2) + 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Height | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Vertices | 1 | 2 | 4 | 7 | 12 | 20 | 33 | 54 | 88 | 143 | 232 | 376 | 609 | 986 | 1596 | 2583 |

^ the chart is available behind also



Delete and check for rotation.

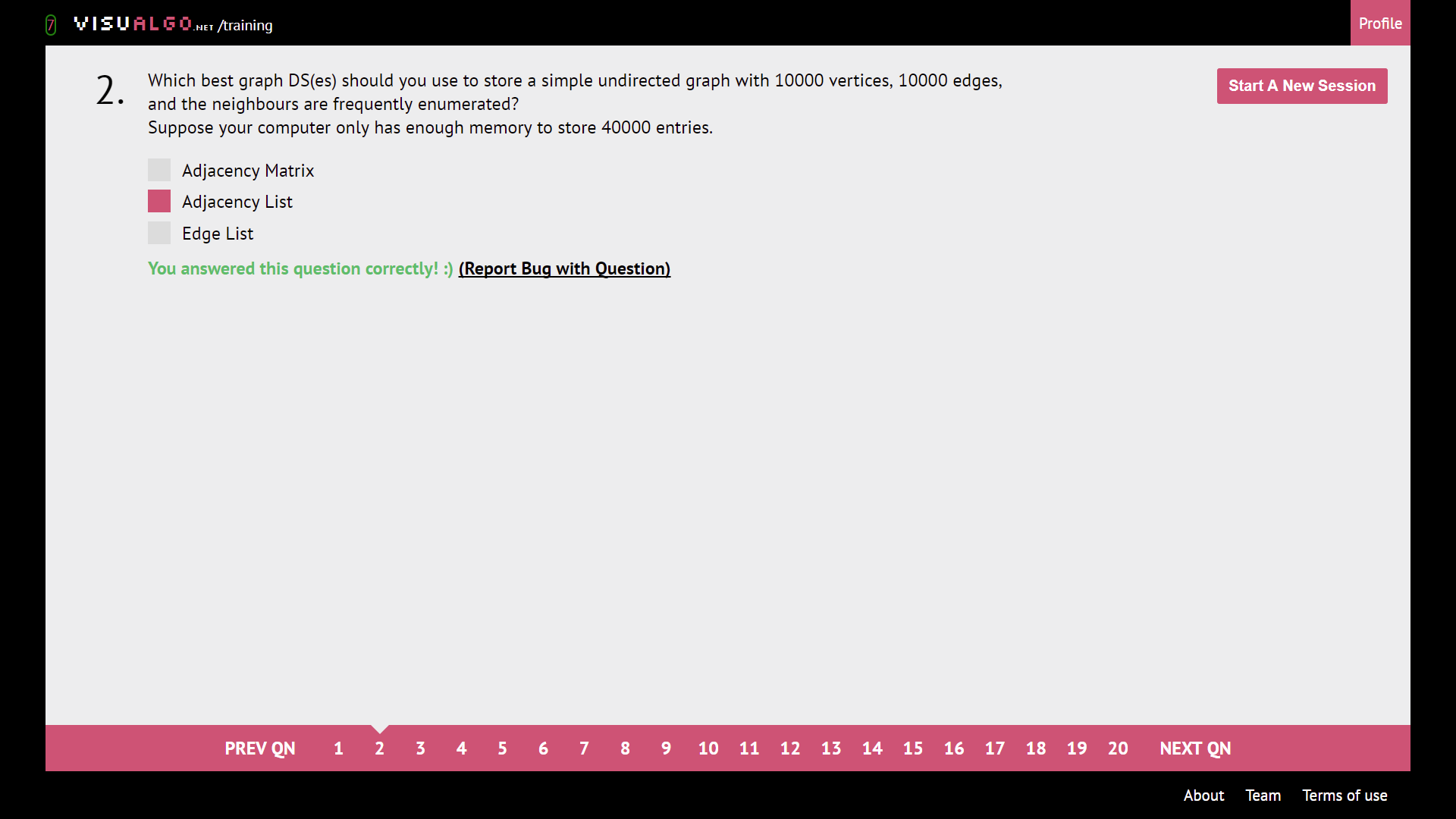
**Graph DS**

Directed acyclic, just make sure small number -> bigger number.

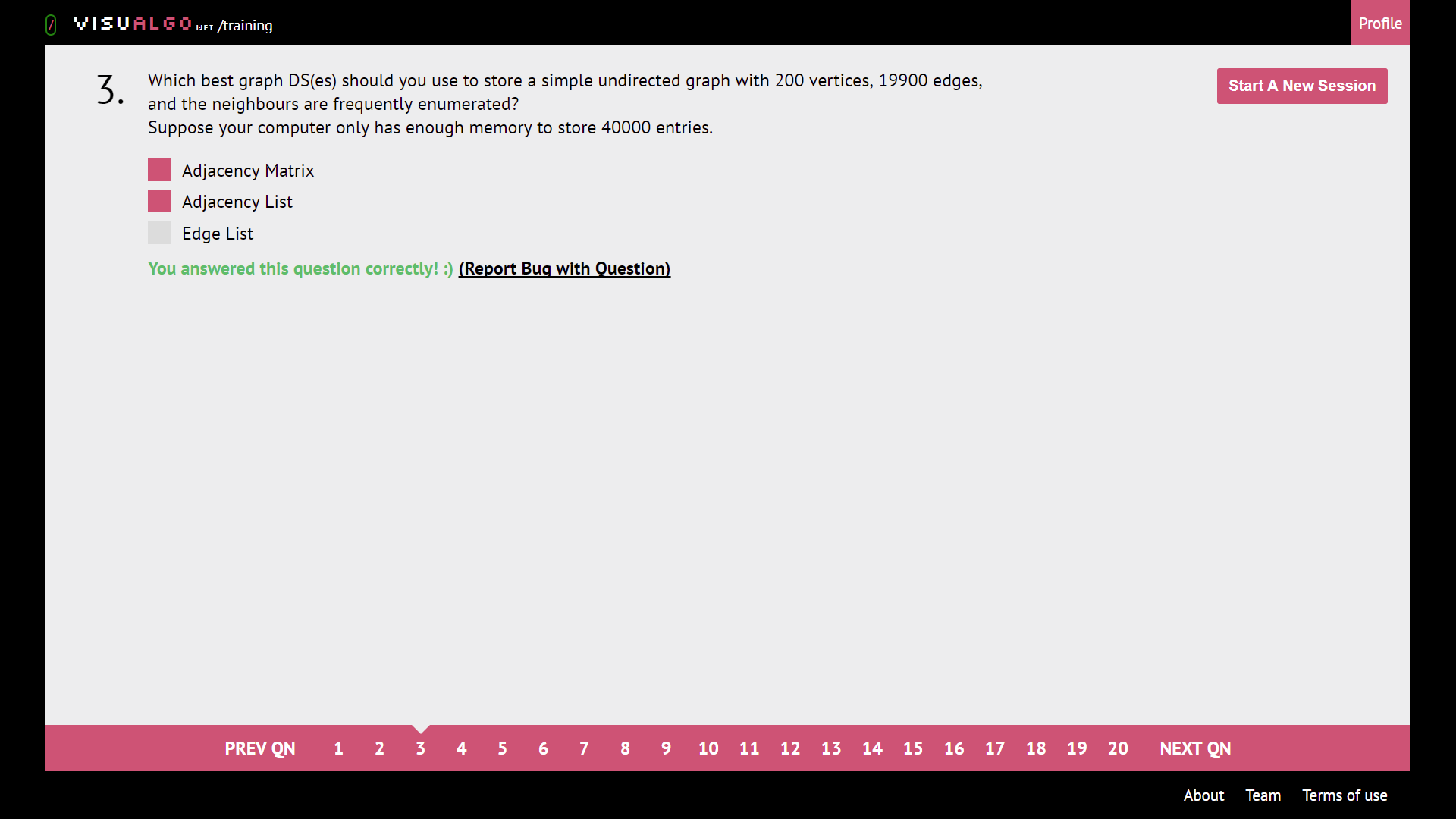
Tree: V vertices, V-1 edges, acyclic.

Complete graph: An edge between any pair of vertices.

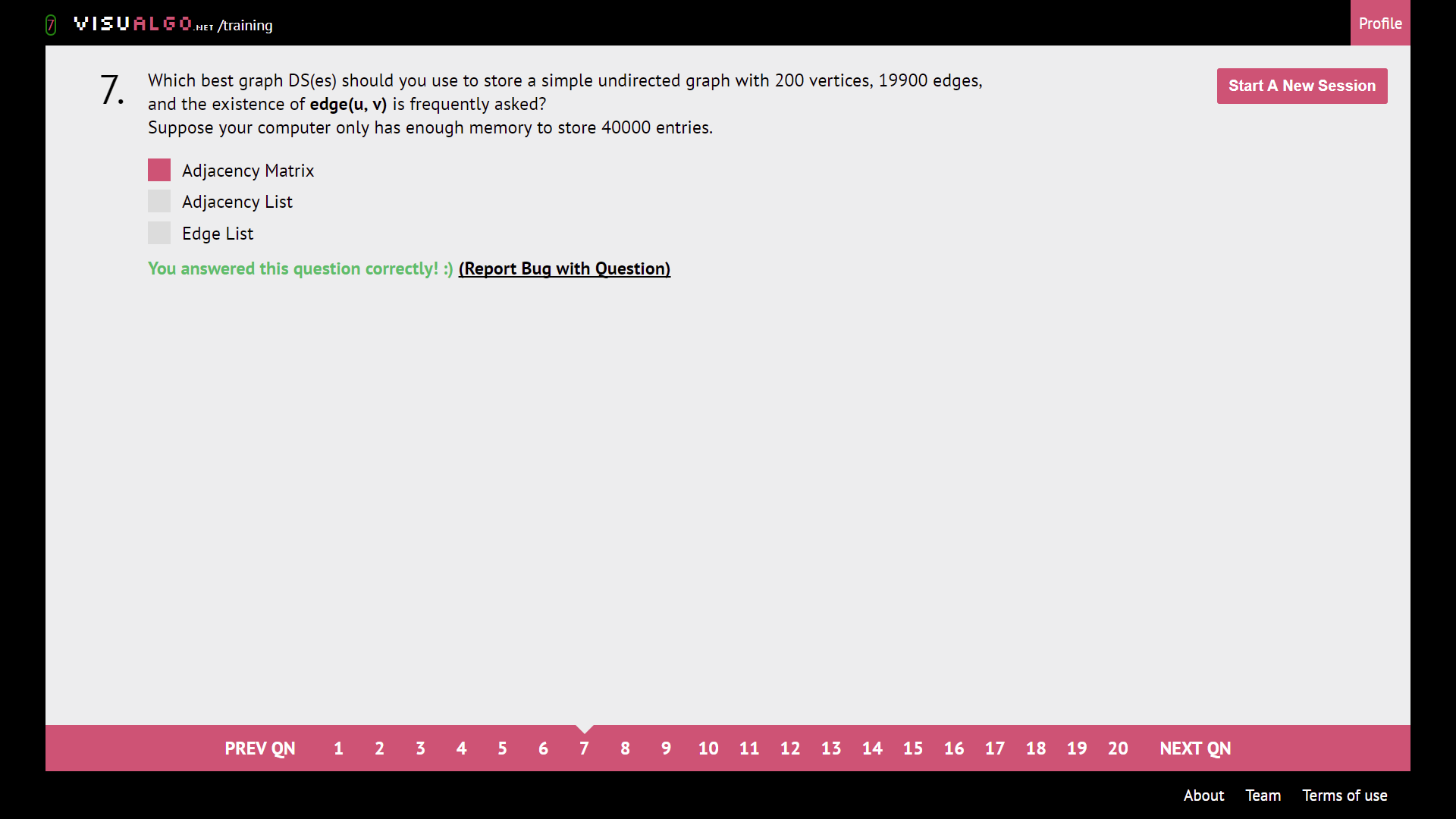
Connected graph: From any vertex, can get to all other vertex



Need to enumerate neighbours, **BUT NOT ENOUGH MEMORY (we need V^2 entries for AM)**



Adjacency Matrix enumerate not as fast but still accepted, **so depends on memory.**

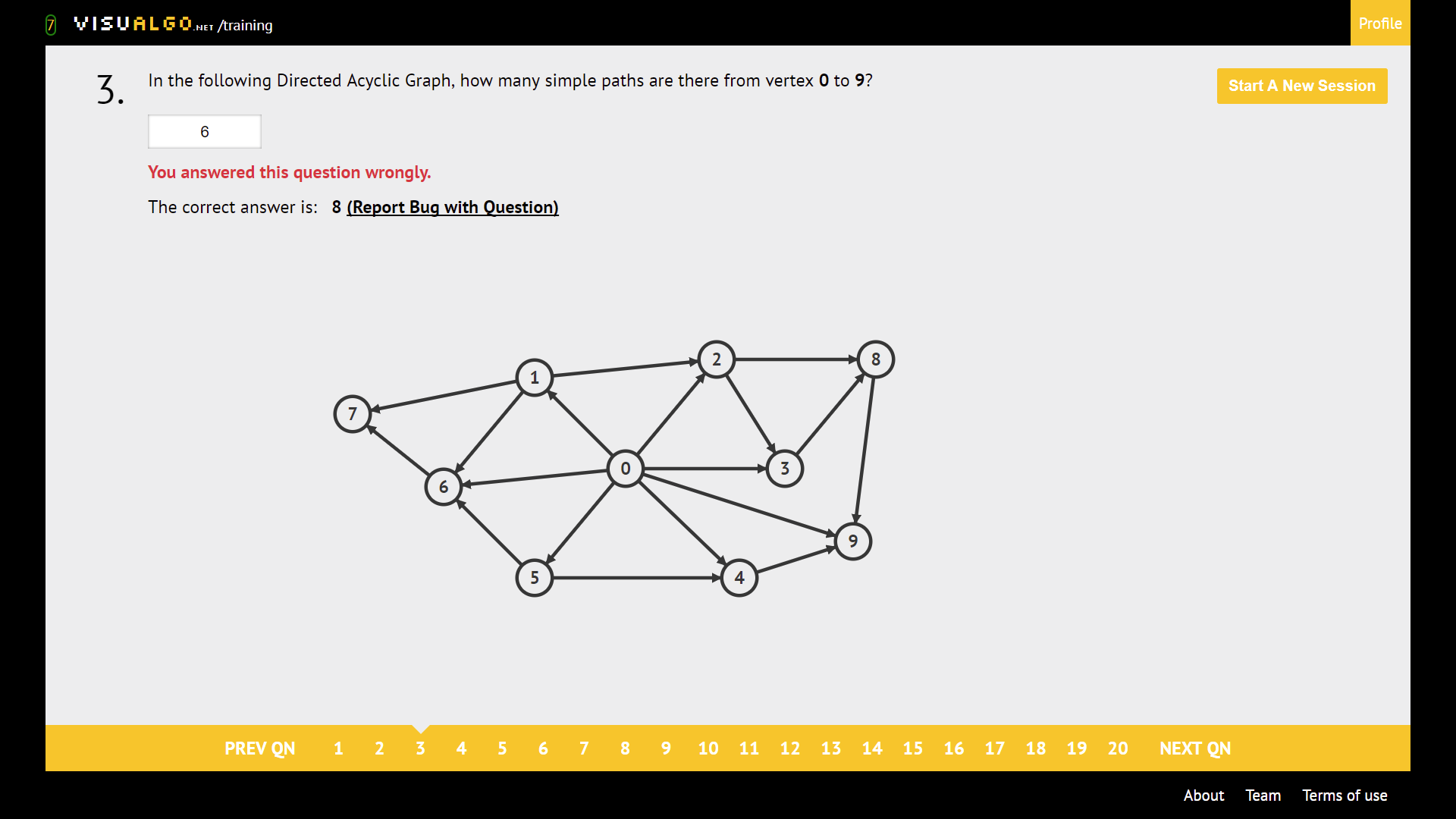


Ask about existence of edge -> AM (Good for Floyd-Warshall’s 4 line wonder :D)

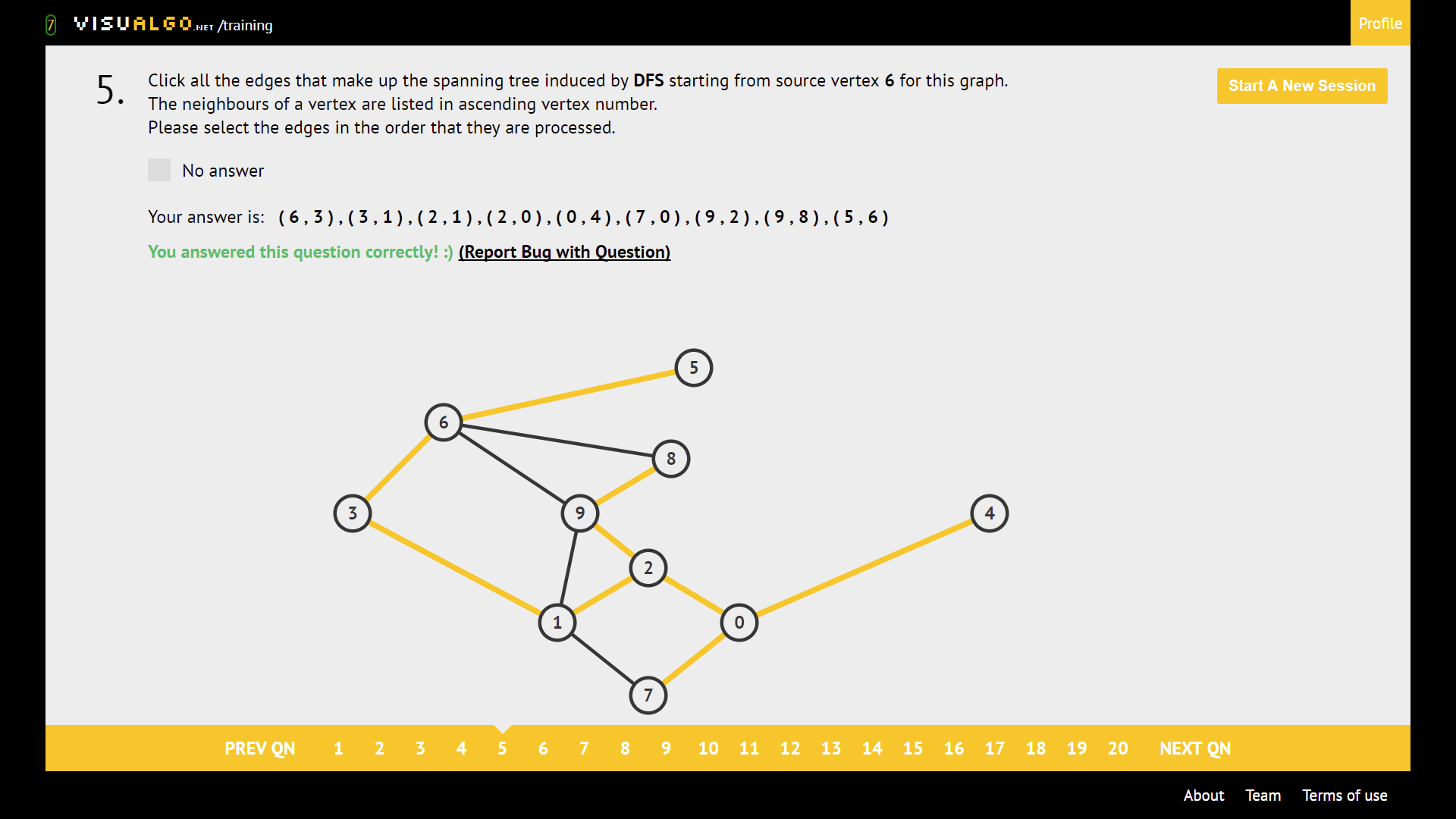


Edge list can be sorted by edge.

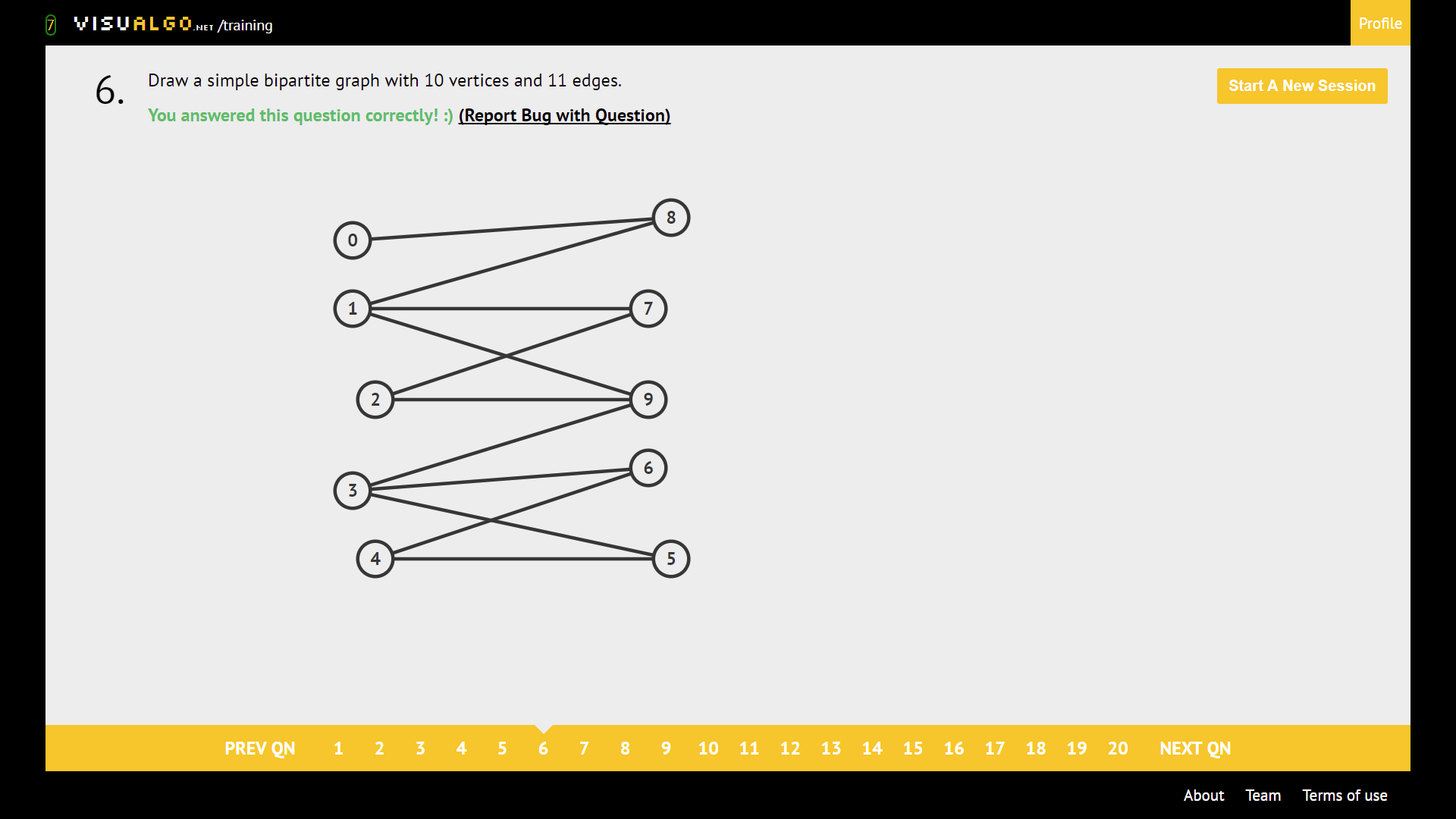
**Graph traversal**



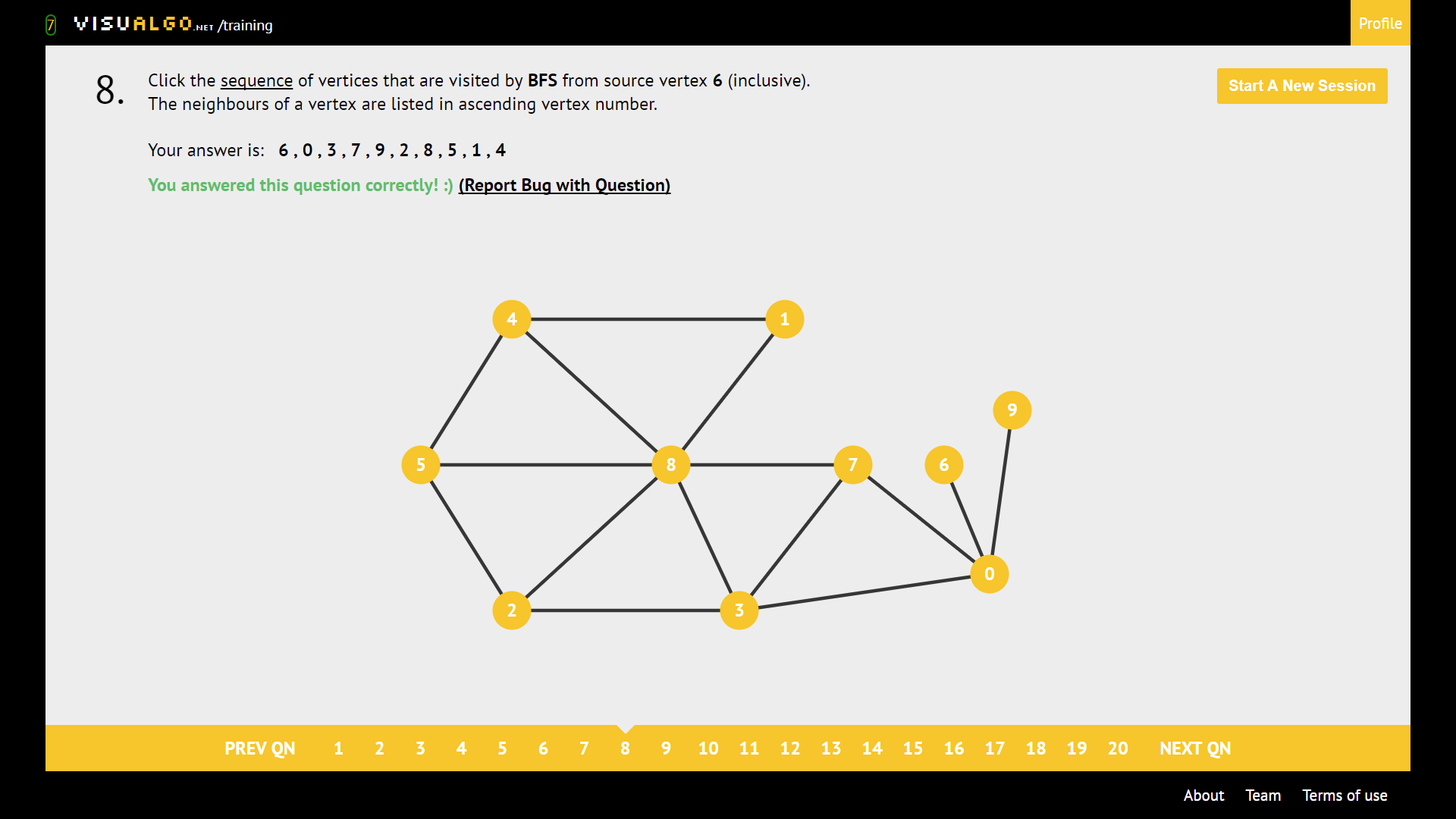
Write down all the paths. (tedious, but idiotproof)



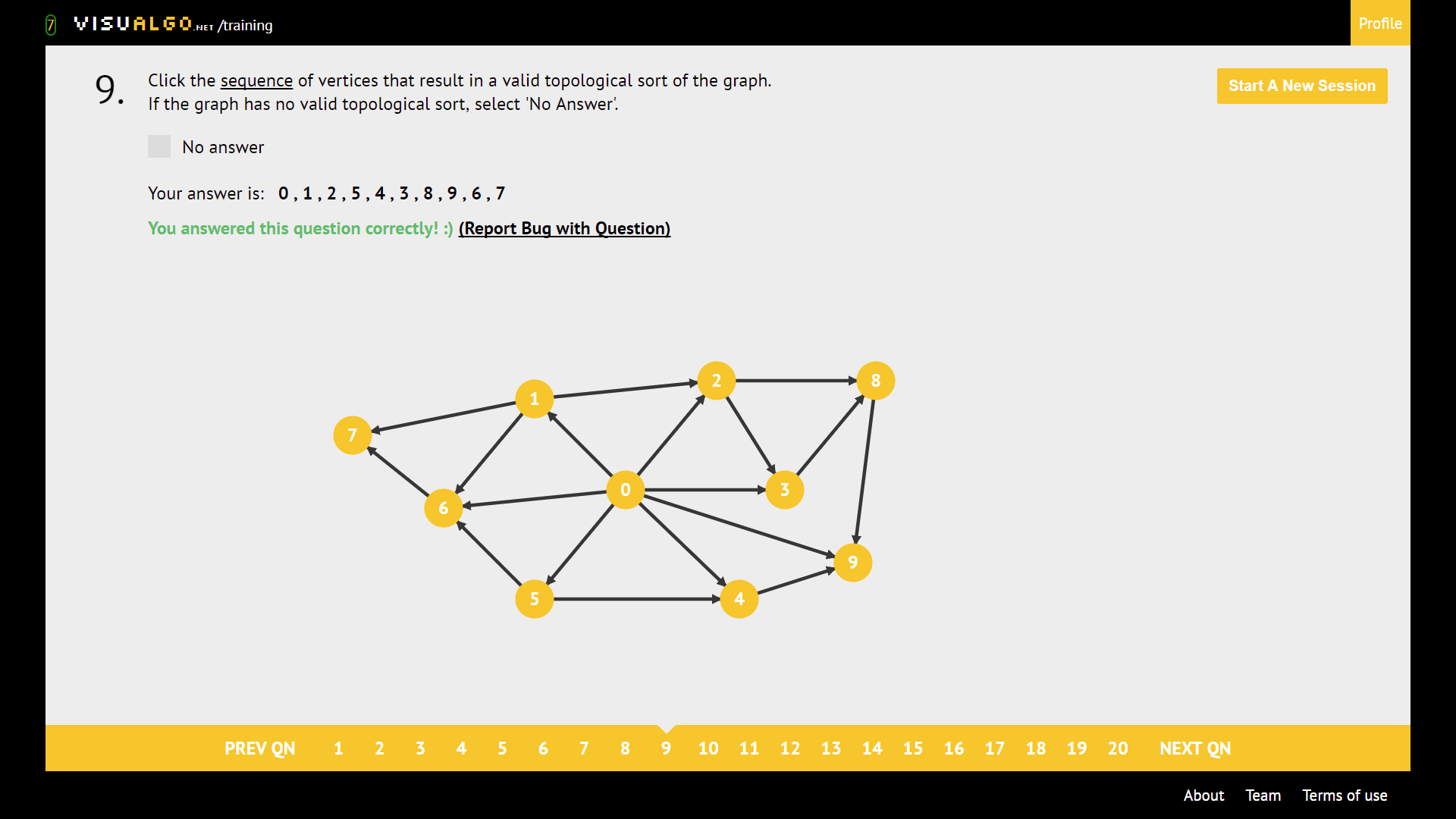
DFS: Go deep, recurse back and go deep for neighbour. Take node of order of neighbours.



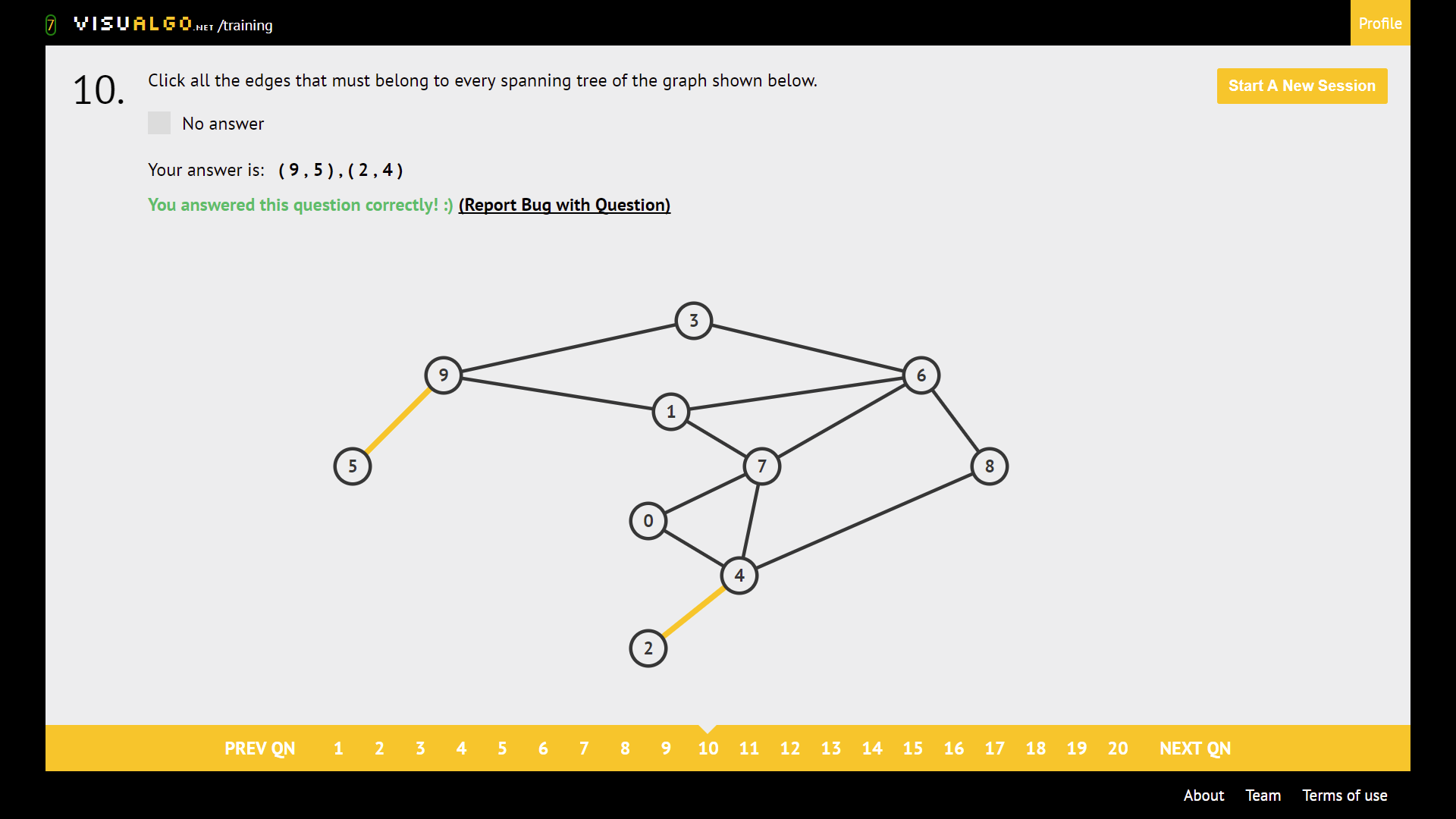
Disjoint set members are not connected to one another.



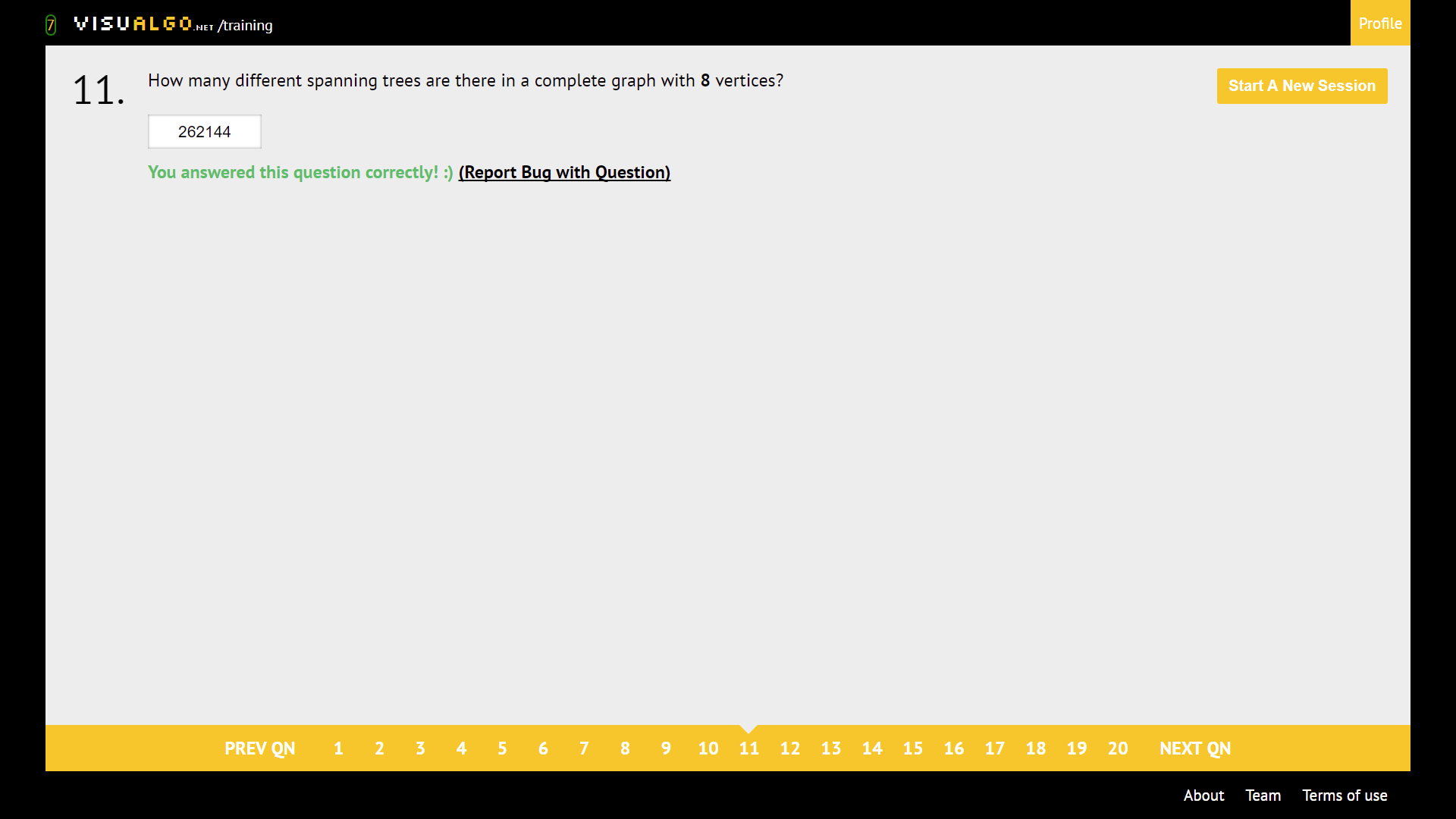
BFS: Visit all neighbours, then all neighbours of neighbours, and so on. Keep track of the order of exploration of neighbours!

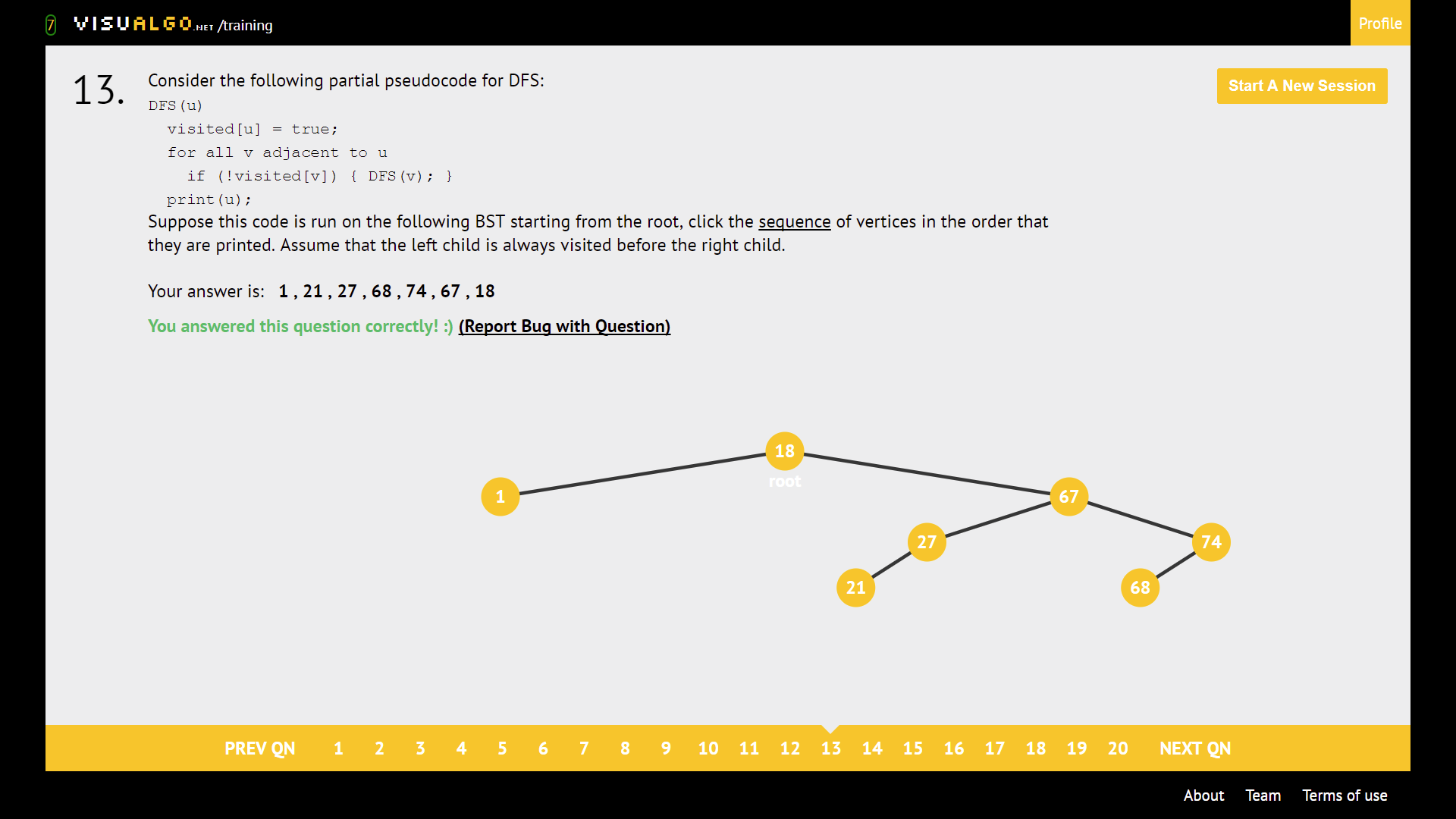


Topological sort

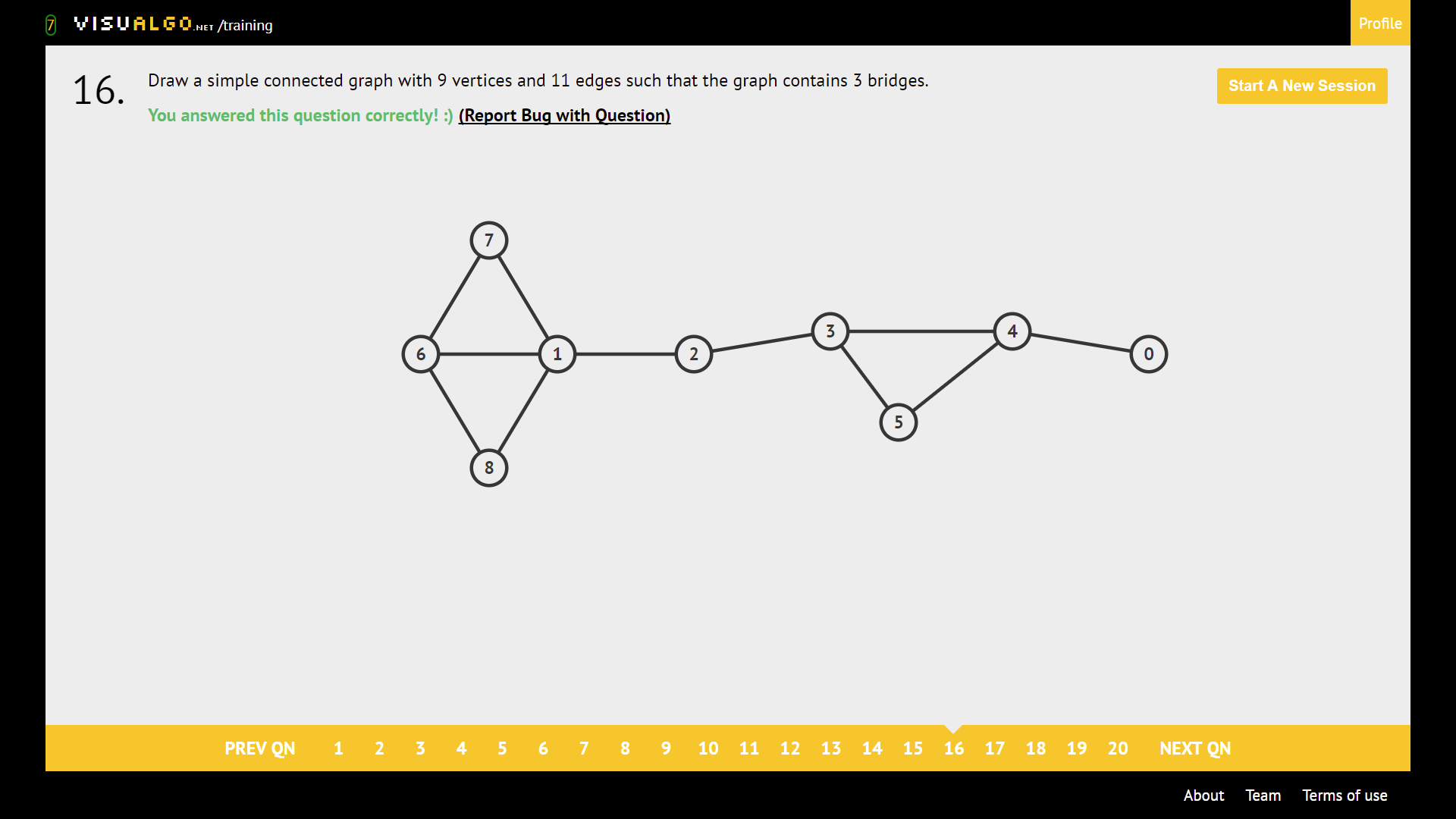


Vertex with only 1 edge. “Burn all the bridges”





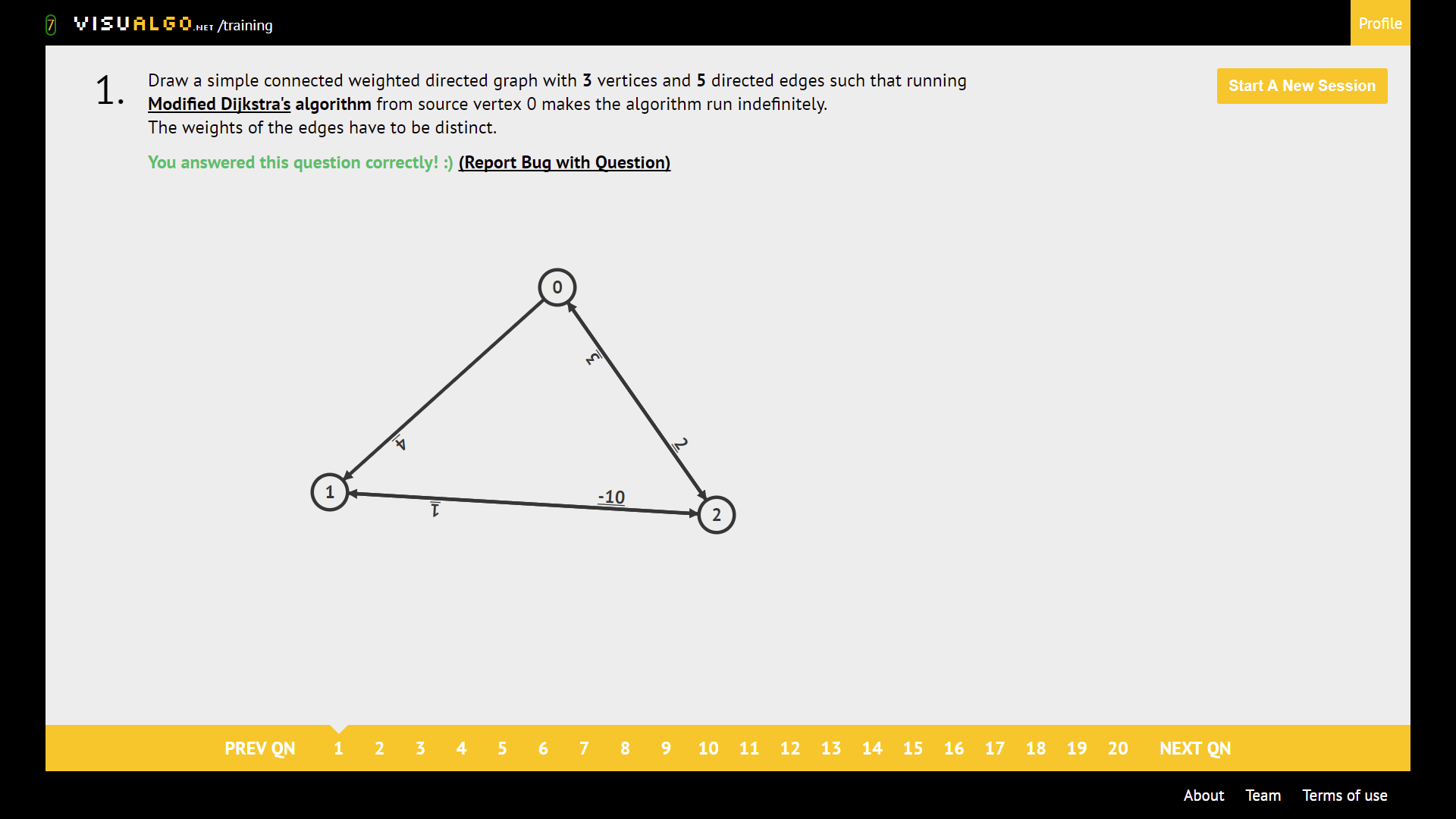
Print comes after exploring all neighbours -> Post-order traversal



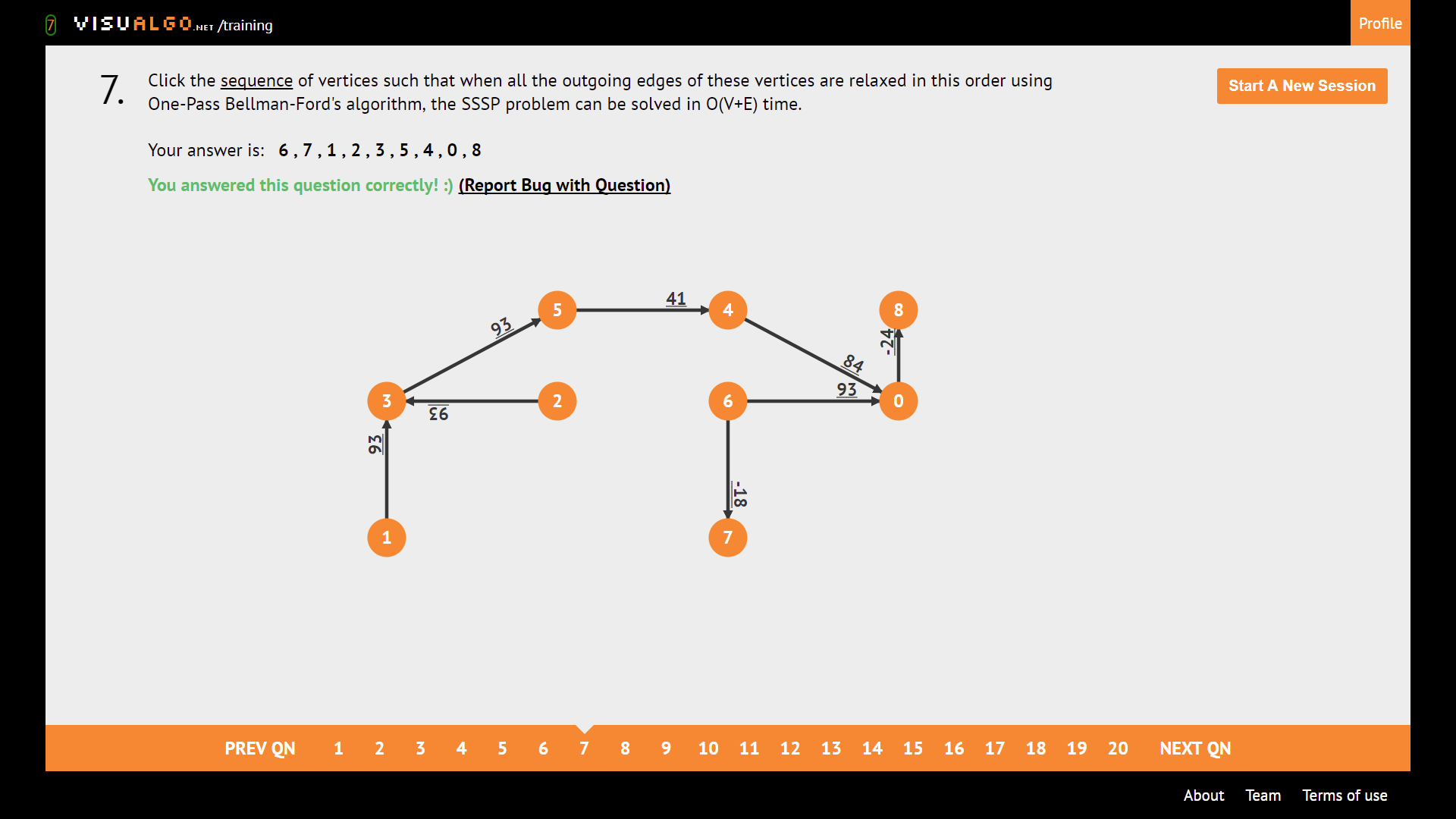
Bridge: Edge of undirected graph which removal disconnects the graph.

Cut vertex/Articulation point: Vertex of undirected graph which removal disconnects the graph.

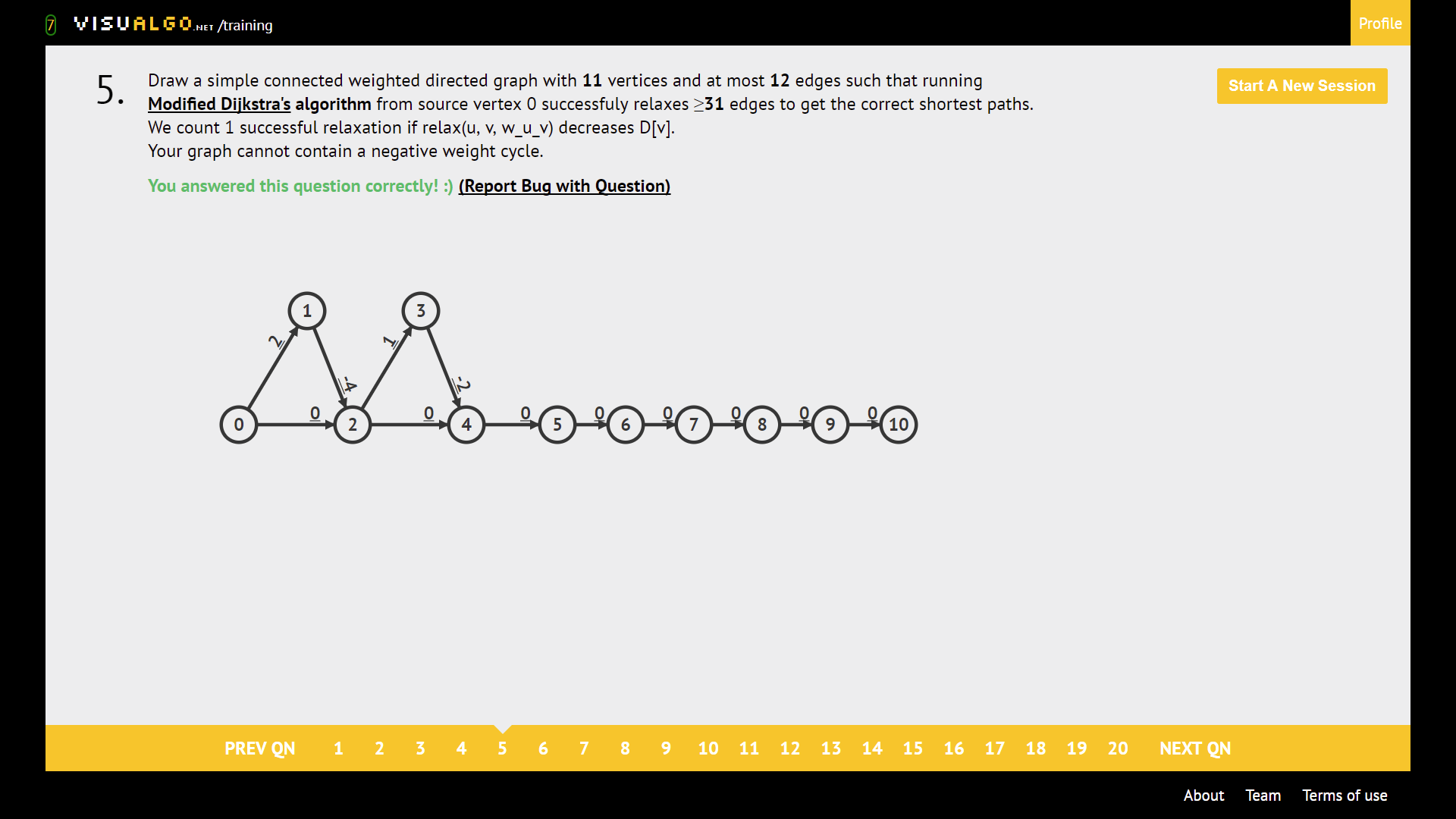
**SSSP**



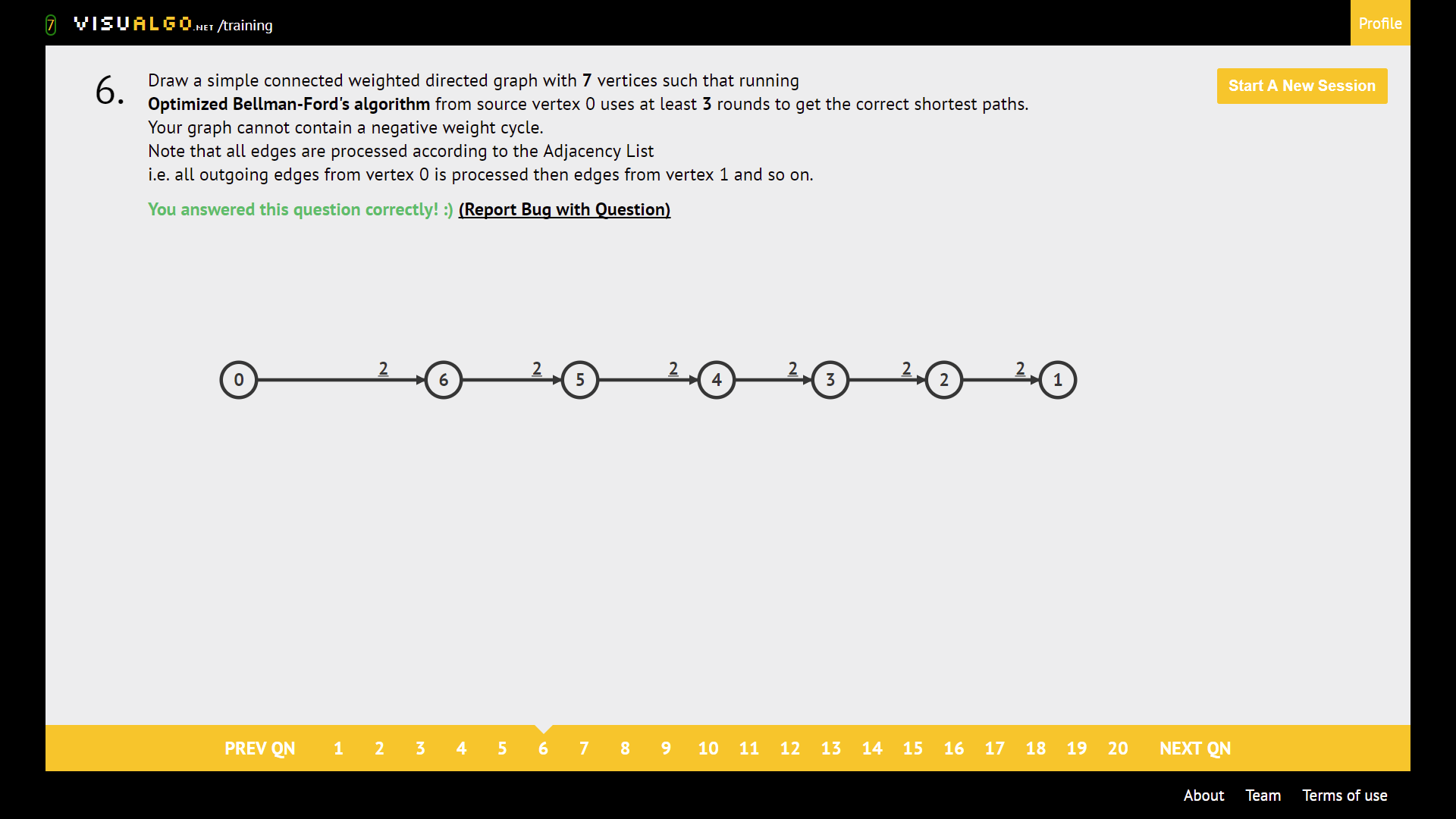
Create a negative cycle. **MAKE SURE THE WEIGHTS ARE DISTINCT**



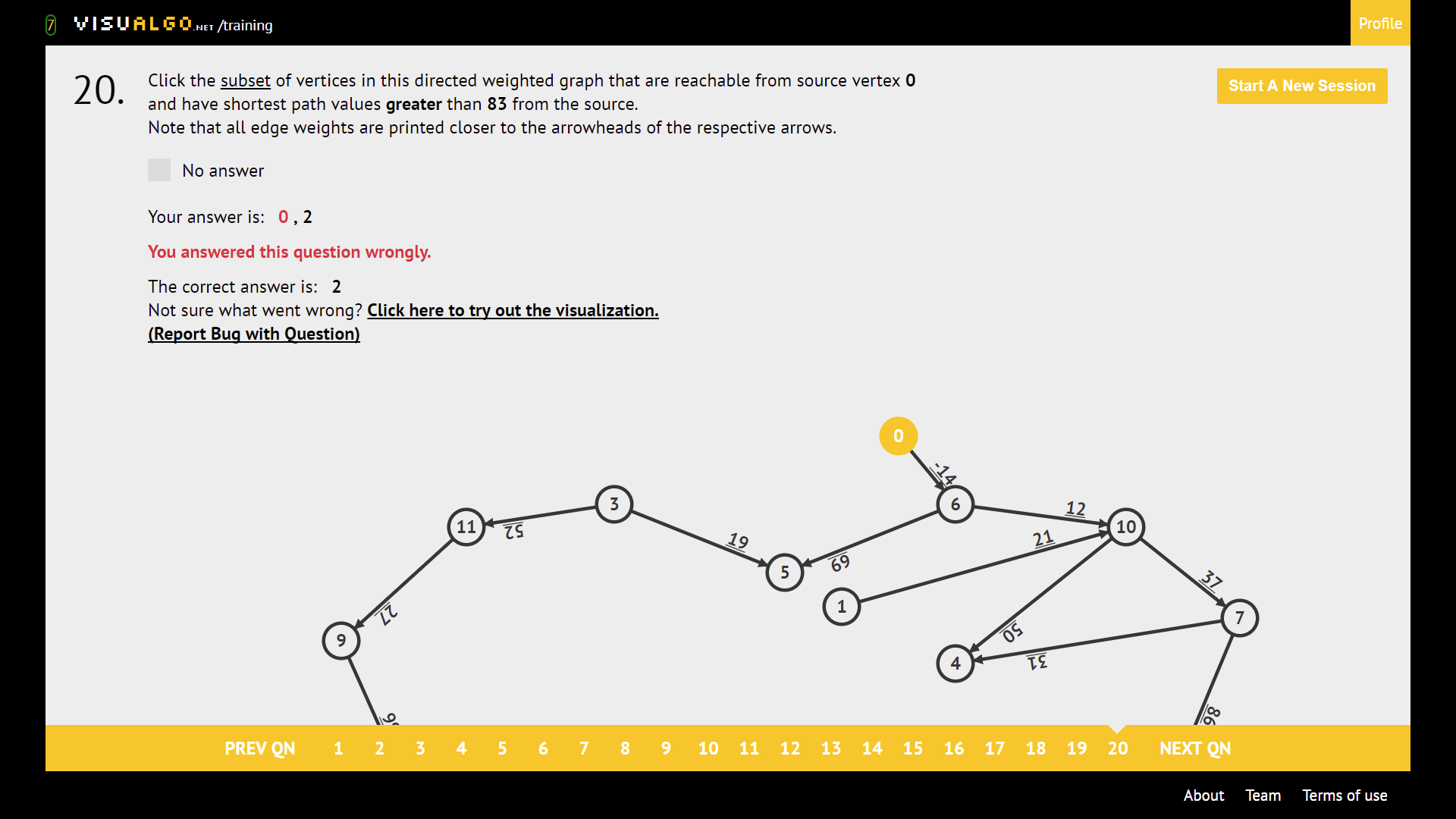
Topological sort.



Dijkstra Killer. **THERE ARE (E-V+1) TRIANGLES.**



Bellman-Ford Killer.



Note path weight criteria.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bellman-Ford | Original Dijkstra | Modified Dijkstra |
| Terminate | Always | Always | Does not terminate when there’s negative weight cycle |
| Wrong when there’s | Negative weight cycle | Negative weight | Negative weight cycle (it doesn’t even terminate in the first place ☹ ) |

# **Tables**

Max Heap max swaps (min swaps = 0. what if it’s already a heap?)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Elements | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Swaps | 4 | 4 | 7 | 7 | 8 | 8 | 10 | 10 | 11 |

Max Heap max comparisons

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Elements | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Compare | 7 | 8 | 11 | 14 | 15 | 16 | 18 | 20 | 21 |

Max Heap Min Comparisons: N-1 (min comparisons is NOT ZERO)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Elements | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Compare | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

# **Binary Search Tree: how many permutations**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Elements | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Catalan # | 1 | 2 | 5 | 14 | 42 | 132 | 429 | 1430 | 4862 |

**AVL Tree: Minimum # of vertices (Height(n) = height(n-1) + height(n-2) + 1)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Height | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Vertices | 1 | 2 | 4 | 7 | 12 | 20 | 33 | 54 | 88 | 143 | 232 | 376 | 609 | 986 | 1596 | 2583 |

# **Graph Structures**

Neighbours Frequently Enumerated AND **HAS ENOUGH MEMORY: AL + AM** (We need V^2 memory for AM)

Neighbours Frequently Enumerated AND **NOT ENOUGH MEMORY:** **AL ONLY** (memory < V^2 )

Existence of edge(u,v): **AM ONLY**

Edges need to be sorted: **EDGE LIST ONLY**

**Number of spanning trees in complete graph n^(n-2)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Vertices | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Ans | 3 | 16 | 453 | 1296 | 16807 | 262144 | 4782969 | 100000000 | 2357947691 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bellman-Ford | Original Dijkstra | Modified Dijkstra |
| Terminate | Always | Always | Does not terminate when there’s negative weight cycle |
| Wrong when there’s | Negative weight cycle | Negative weight | Negative weight cycle |

**Bitmasking/UFDS tallest tree**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 2^N | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **%11** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |  |  |  |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |  |  |  |
|  | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |  |  |  |
|  | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |  |  |  |
|  | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 |  |  |  |
|  | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 |  |  |  |
|  | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 |  |  |  |
|  | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 |  |  |  |
|  | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |  |  |  |
|  | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 |  |  |  |
|  | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **%13** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |  |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |  |
|  | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |  |
|  | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 |  |
|  | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 |  |
|  | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |  |
|  | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |  |
|  | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |  |
|  | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |  |
|  | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **%12** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |  |  |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |  |  |
|  | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |  |  |
|  | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |  |  |
|  | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |  |
|  | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |  |  |
|  | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |  |  |
|  | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 |  |  |
|  | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |  |  |
|  | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 |  |  |
|  | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **%14** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|  | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|  | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 |
|  | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
|  | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 |
|  | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 |
|  | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 |
|  | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |