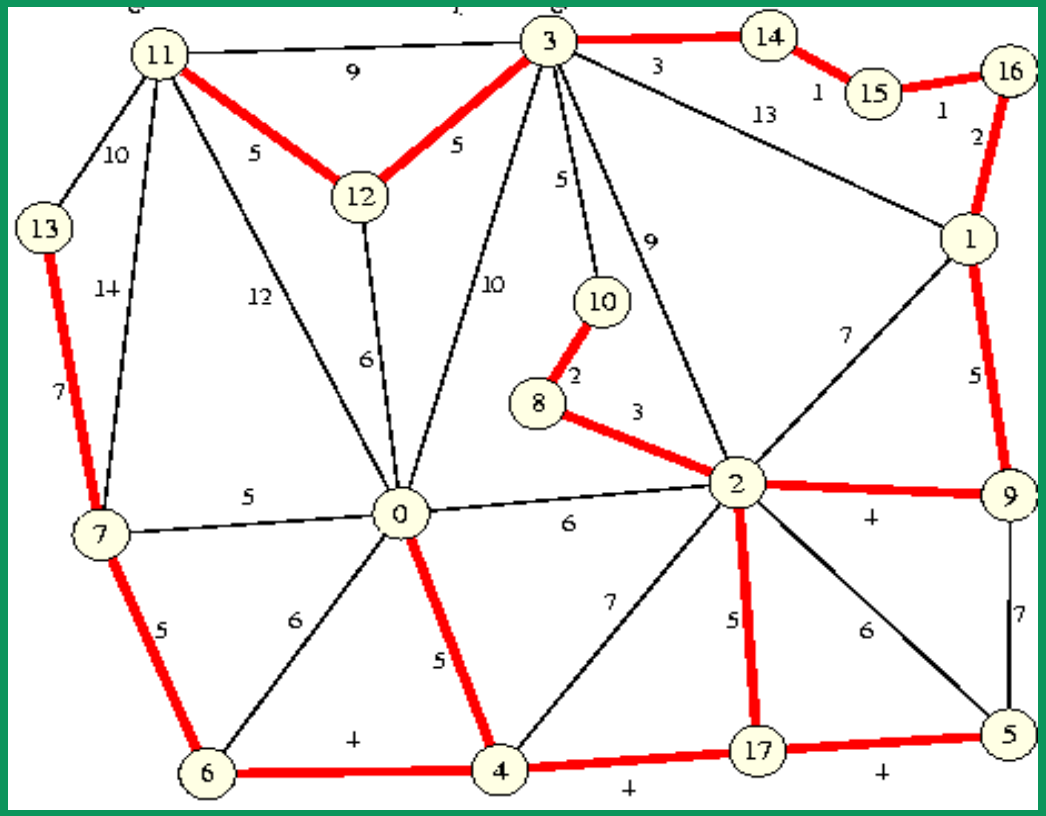
A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees

David R. Karger, Stanford
Philip N. Klein, Brown
Robert E. Tarjan, Princeton

Presented By: Claire Le Goues
Theory lunch
August 14, 2008

Minimum Spanning Tree (MST)



(the weight of this tree is 65)

- **Definition:** a subset of edges in an undirected, weighted graph such that all the vertices are touched by at least one of the edges and the combined weight of all the edges is minimal

MST: Why?

- Cable/phone line layouts
- Airplane route scheduling
- Electricity grids
- …and so on.
- Doing it quickly and efficiently is important!

Previous (Deterministic) Approaches

- Greedy algorithms, polynomial time:
  - Baruvka (1926)
  - Prim's and Kruskal's algorithms.
- Better algorithms, O(nlogn):
  - Chazelle
  - Gabow et al.
  - etc

Can we do better?

SURE!

(probabilistically)

Introducing:

RANDOMNESS

A Sorting Digression

- Quicksort (vs. Mergesort, for example)
- Pick a random pivot point and recurse on parts.
- If we pick the point properly, it's much better!
- But we need to know how/what to pick, and how to combine the subparts
- KKT's algorithm is analogous.

Preliminaries: Cut and Cycle Properties

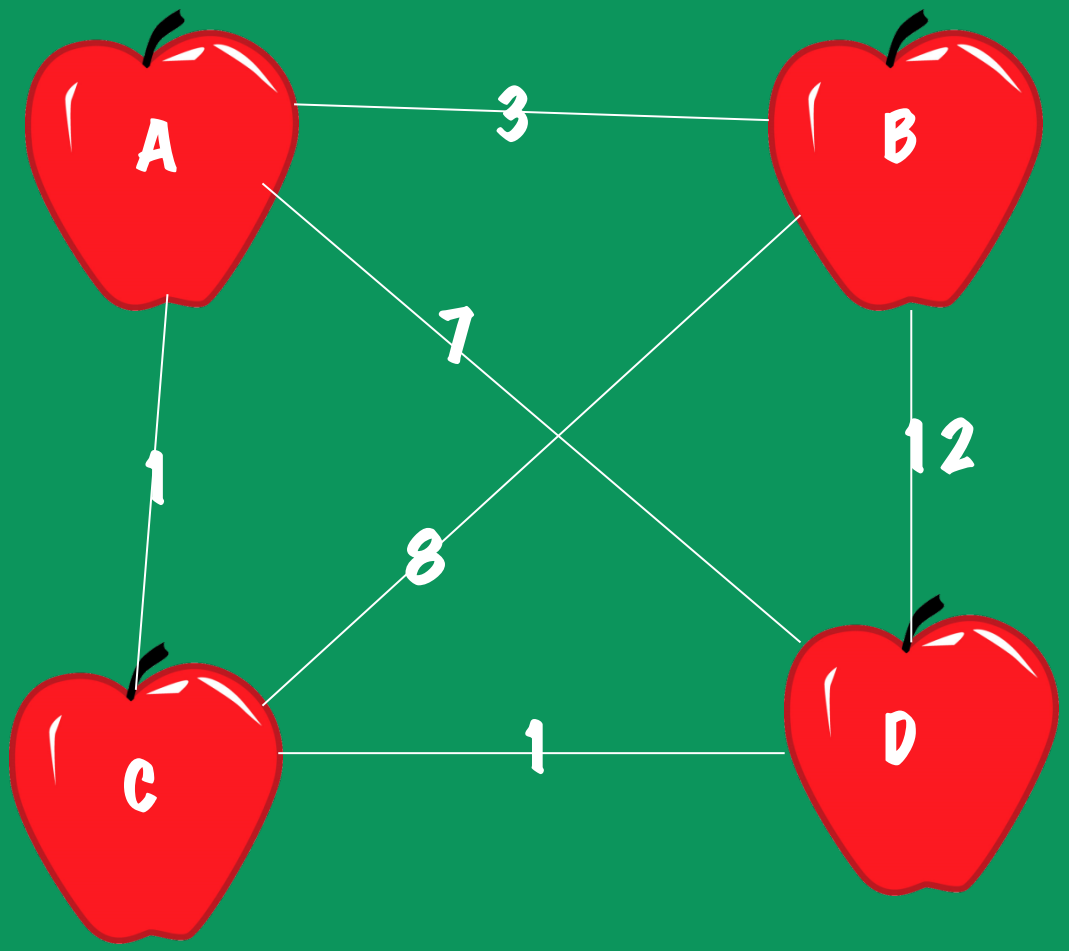
- **Cut property:** For any proper nonempty subset X of the vertices, the lightest edge with exactly one endpoint in X belongs to the minimum spanning tree

- **Cycle property:** For any cycle C in a graph, the heaviest edge in C does not appear in the minimum spanning tree

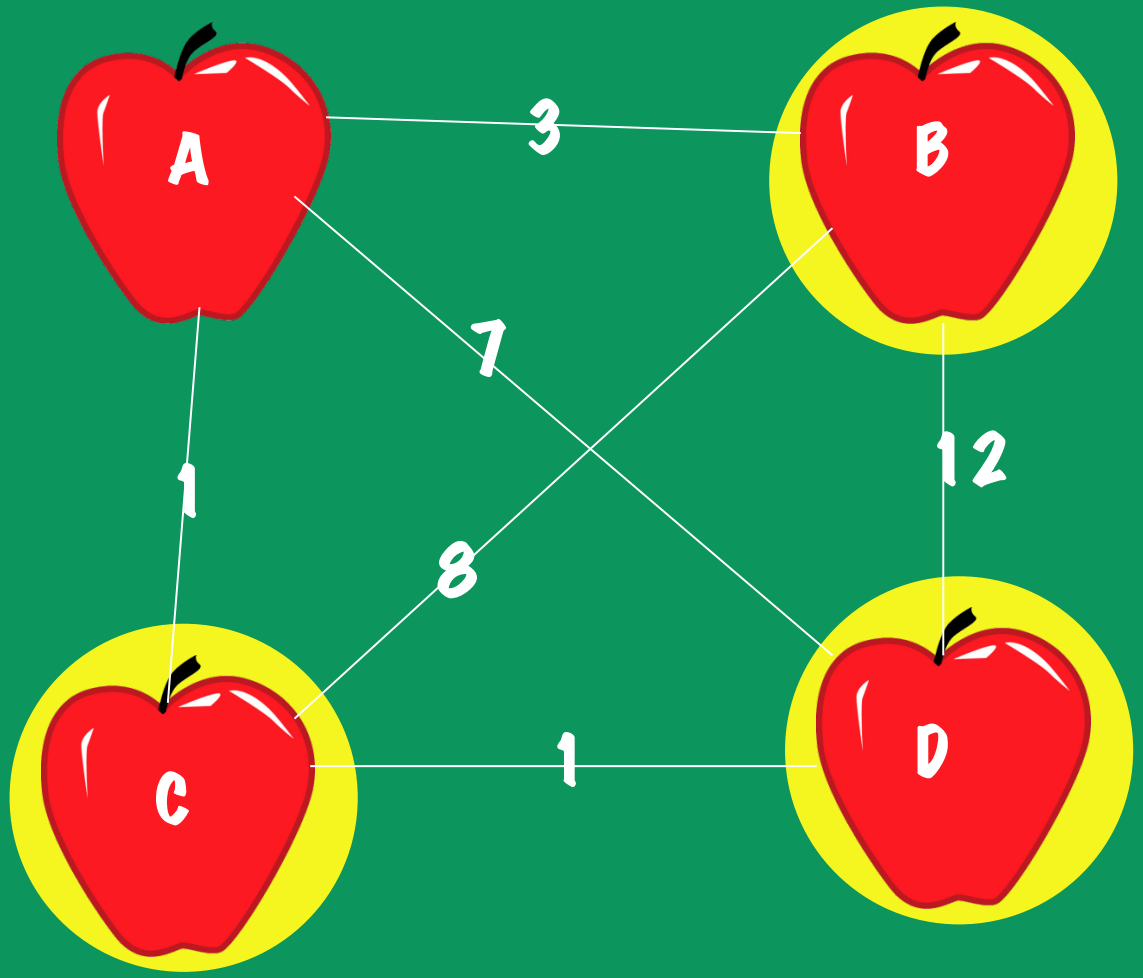
More preliminaries

- $w(x,y)$ is the weight of the edge $\{x, y\}$
- Where F is a forest in G, $F(x,y)$ is the path connecting x and y in F, and $w_F(x,y)$ is the maximum weight of an edge on $F(x,y)$
- An edge (x,y) is **F-heavy** if $w(x,y) > w_F(x,y)$ and **F-light** otherwise.
- *F-heavy edges don't belong in an MST!*

F-Heavy Edges

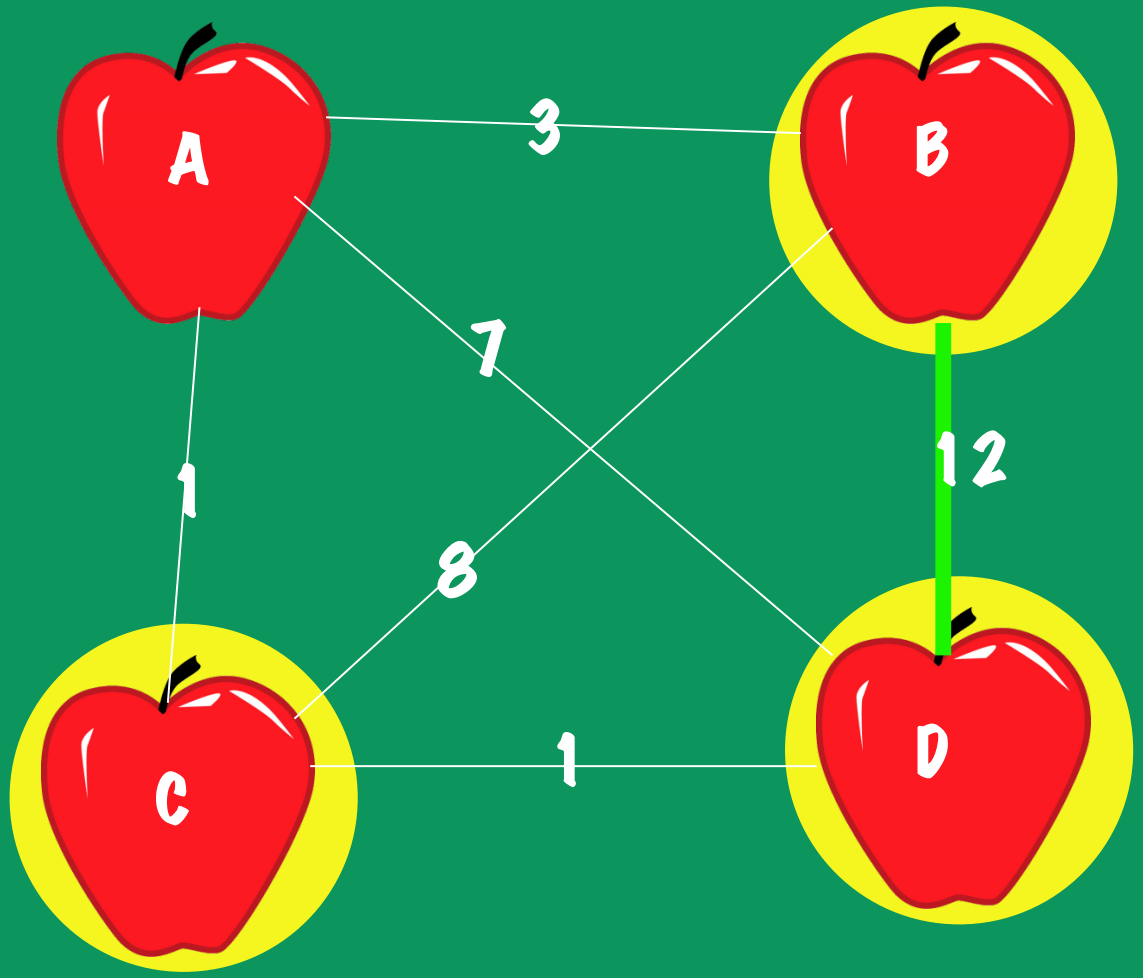


F-Heavy Edges

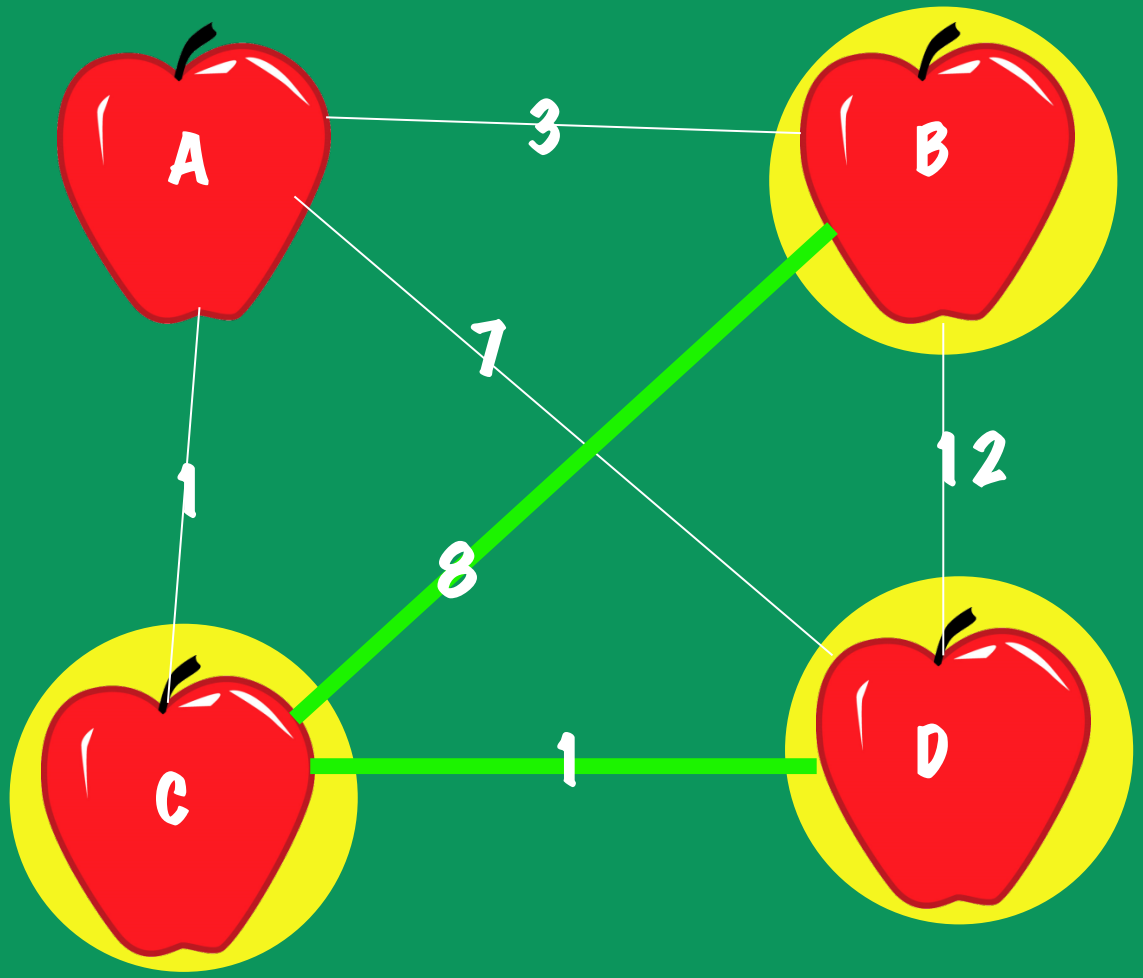


F-Heavy Edges

w(B,D) = 12

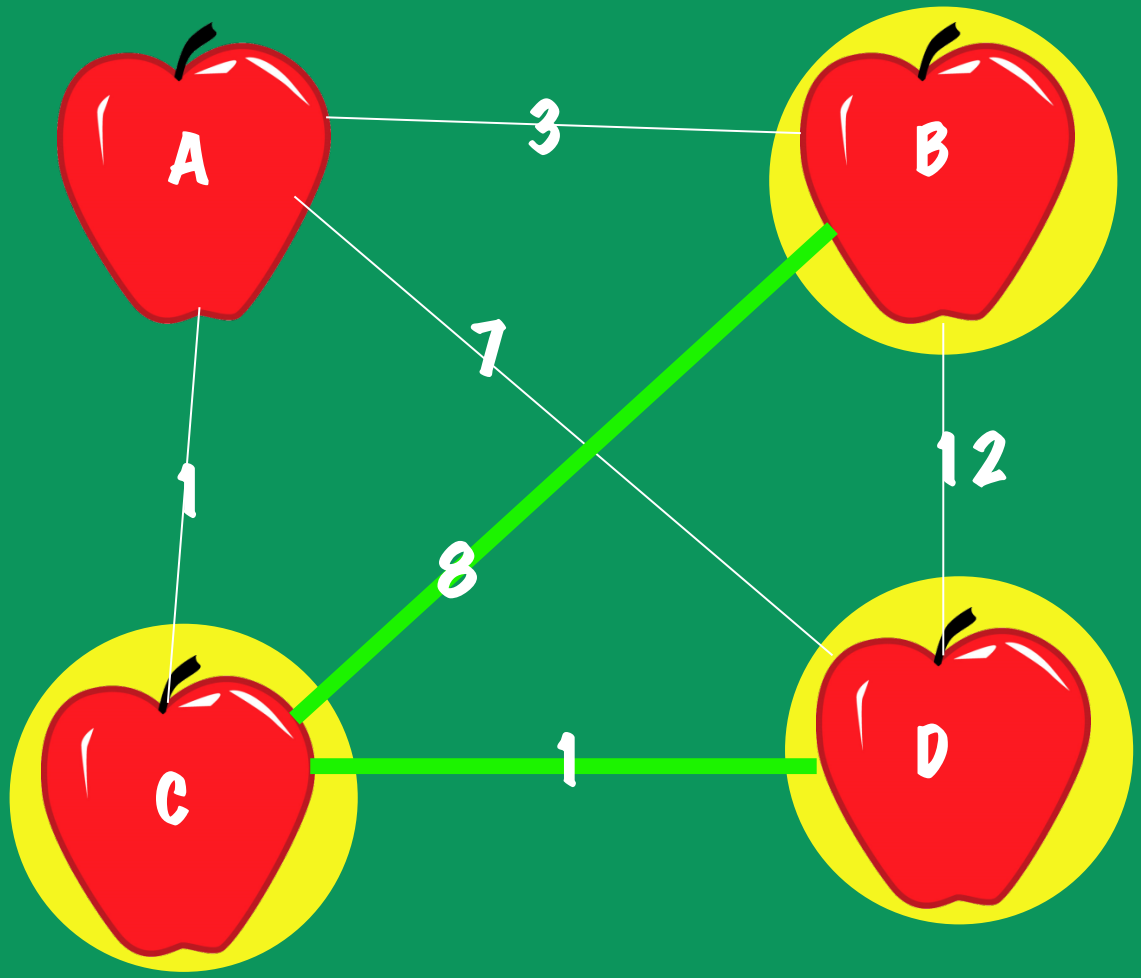


F-Heavy Edges



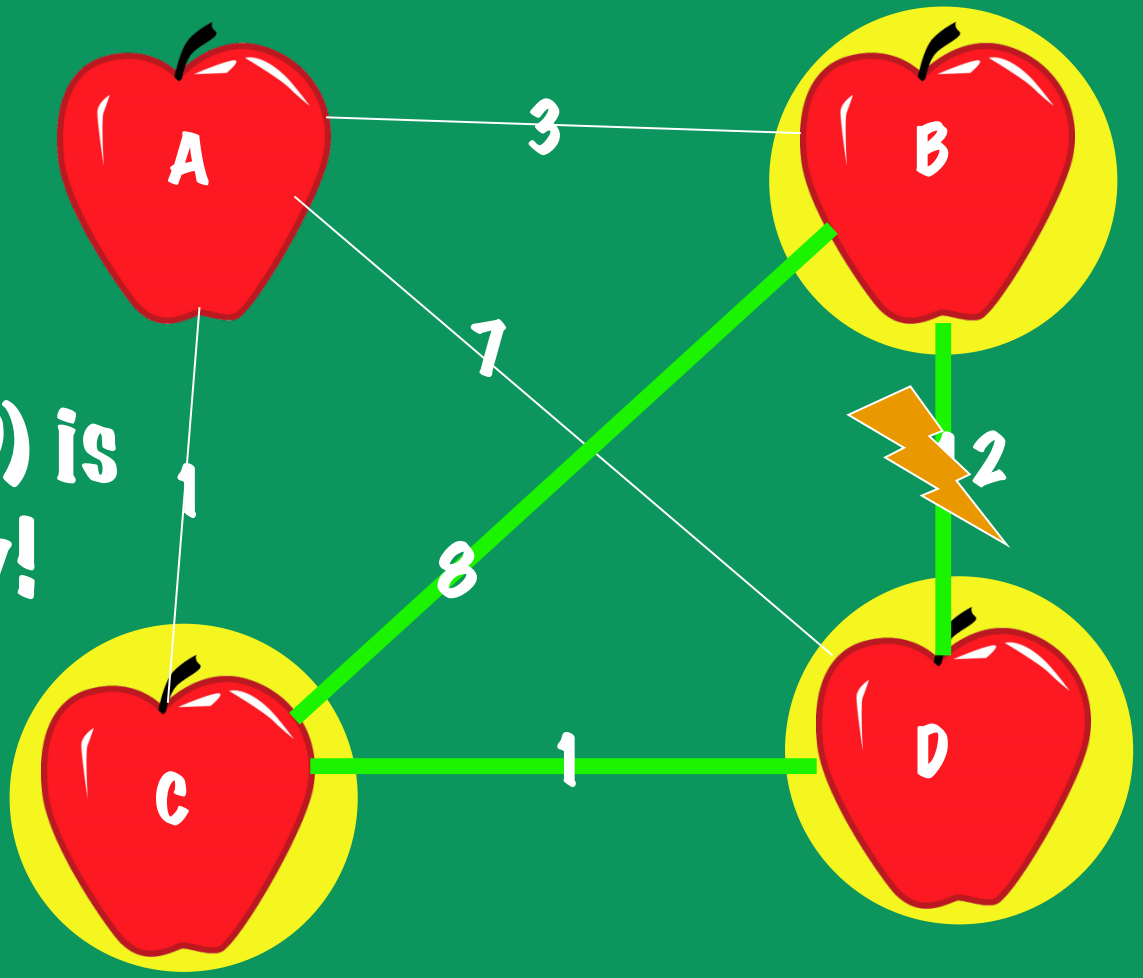
F-Heavy Edges

$W_f(B,D) = 8$

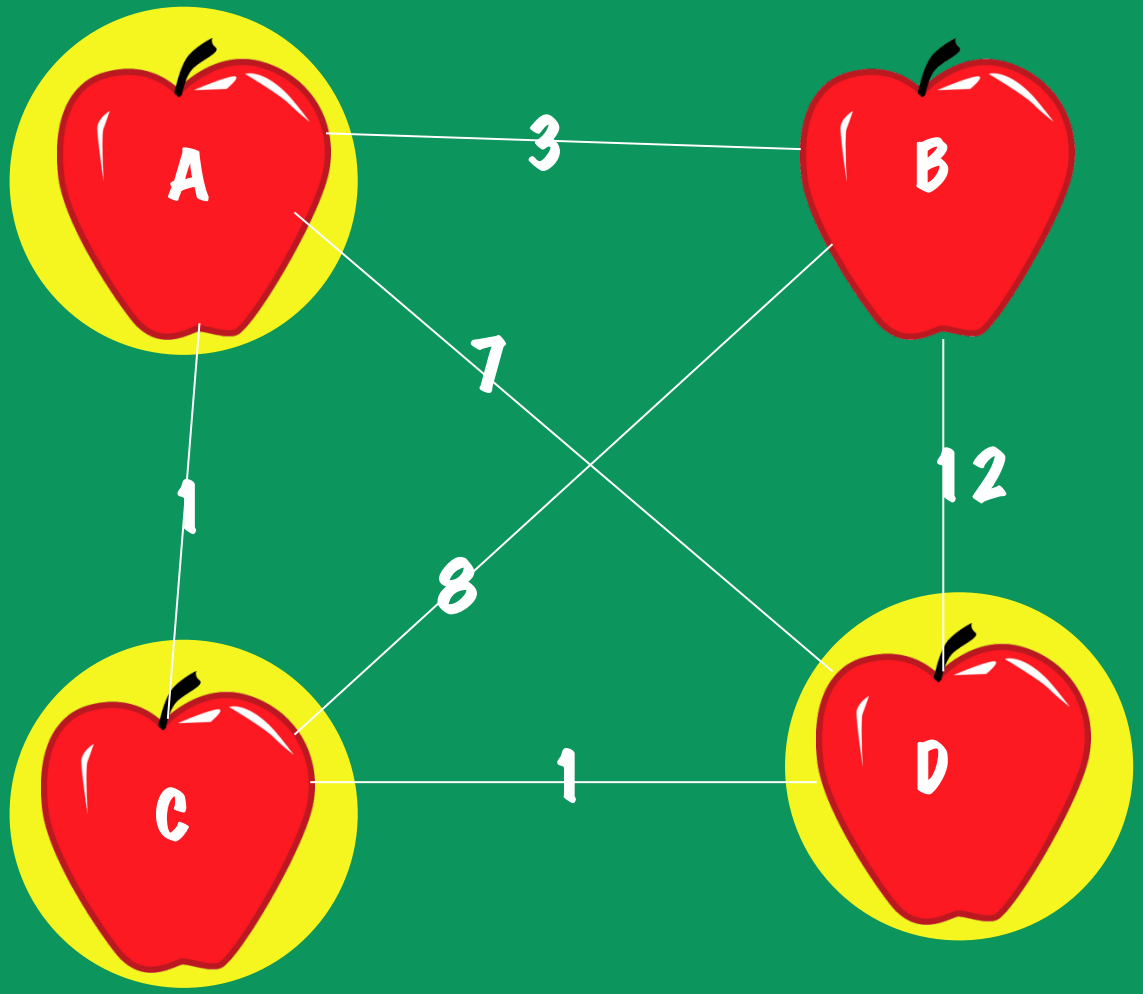


F-Heavy Edges

12 > 8,
So w(B,D) is
F-Heavy!

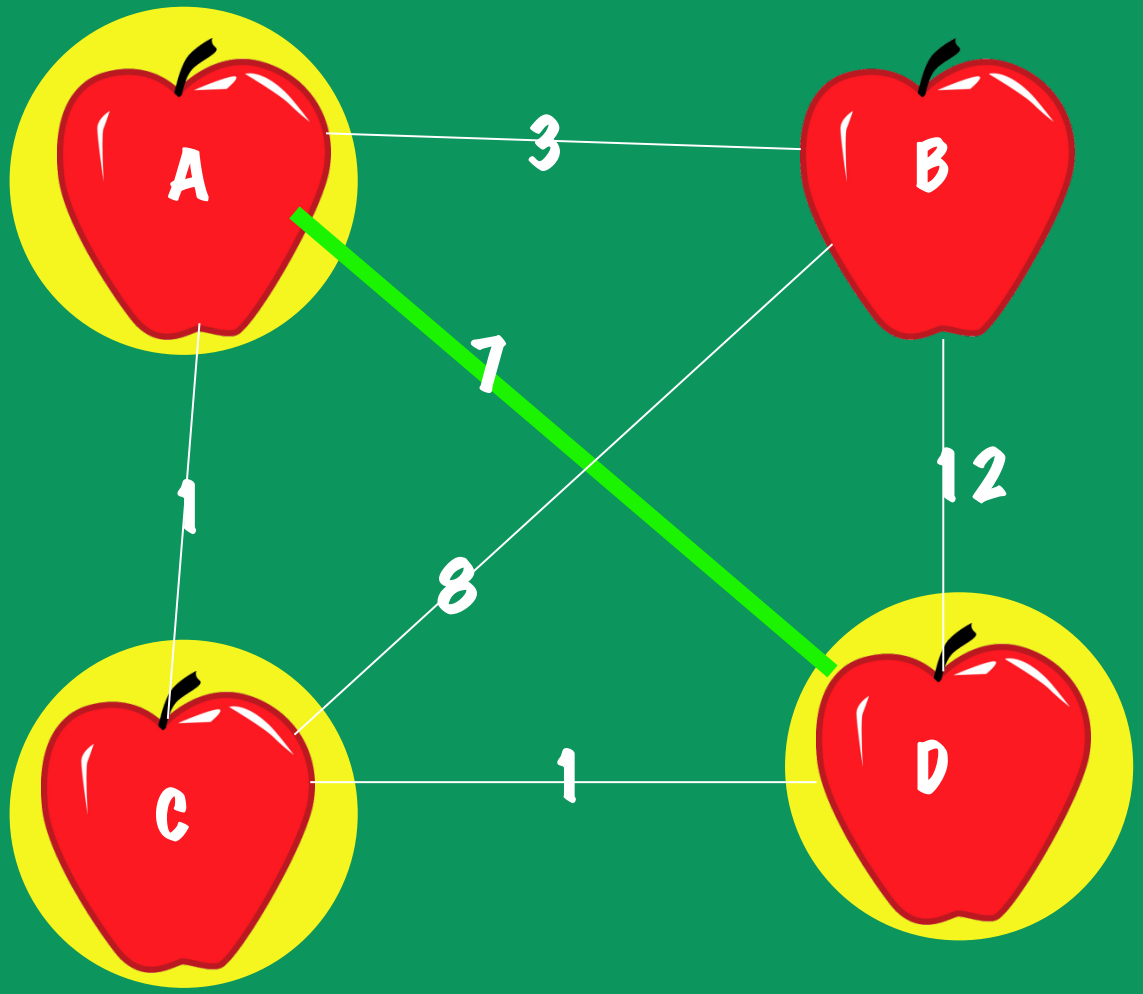


F-Heavy Edges



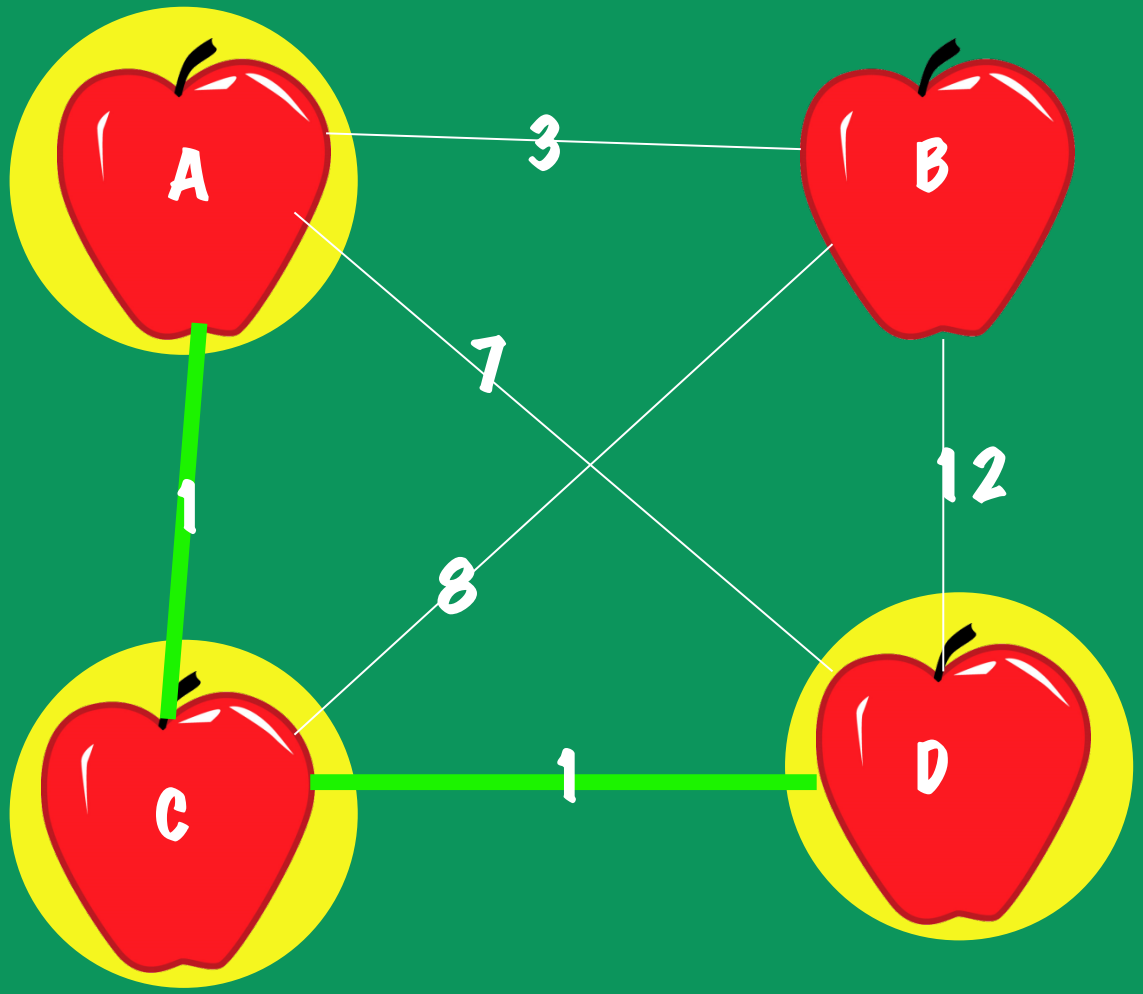
F-Heavy Edges

w(A,D) = 7

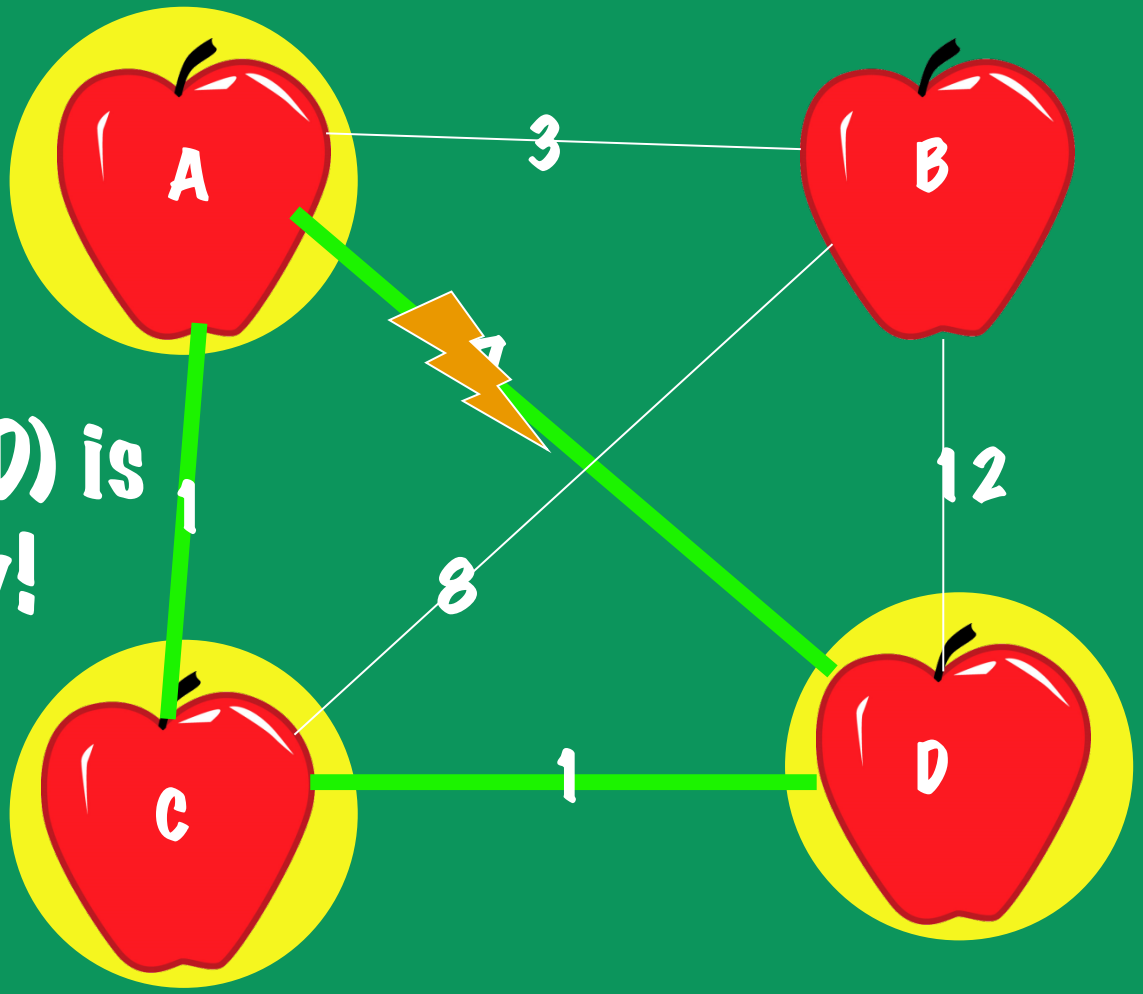


F-Heavy Edges

$w_f(A,D) = 1$

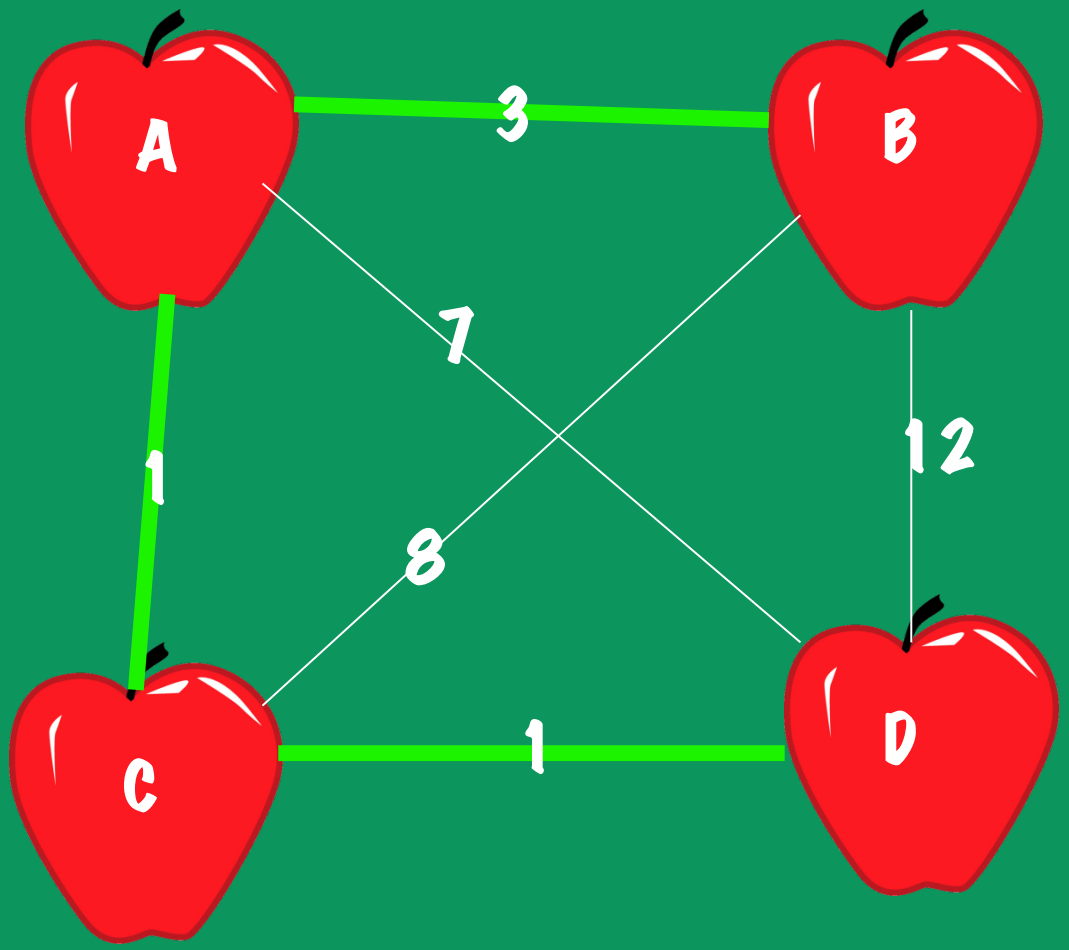


F-Heavy Edges



7 > 1,
So w(A,D) is
F-Heavy!

F-Heavy Edges - The Final MST

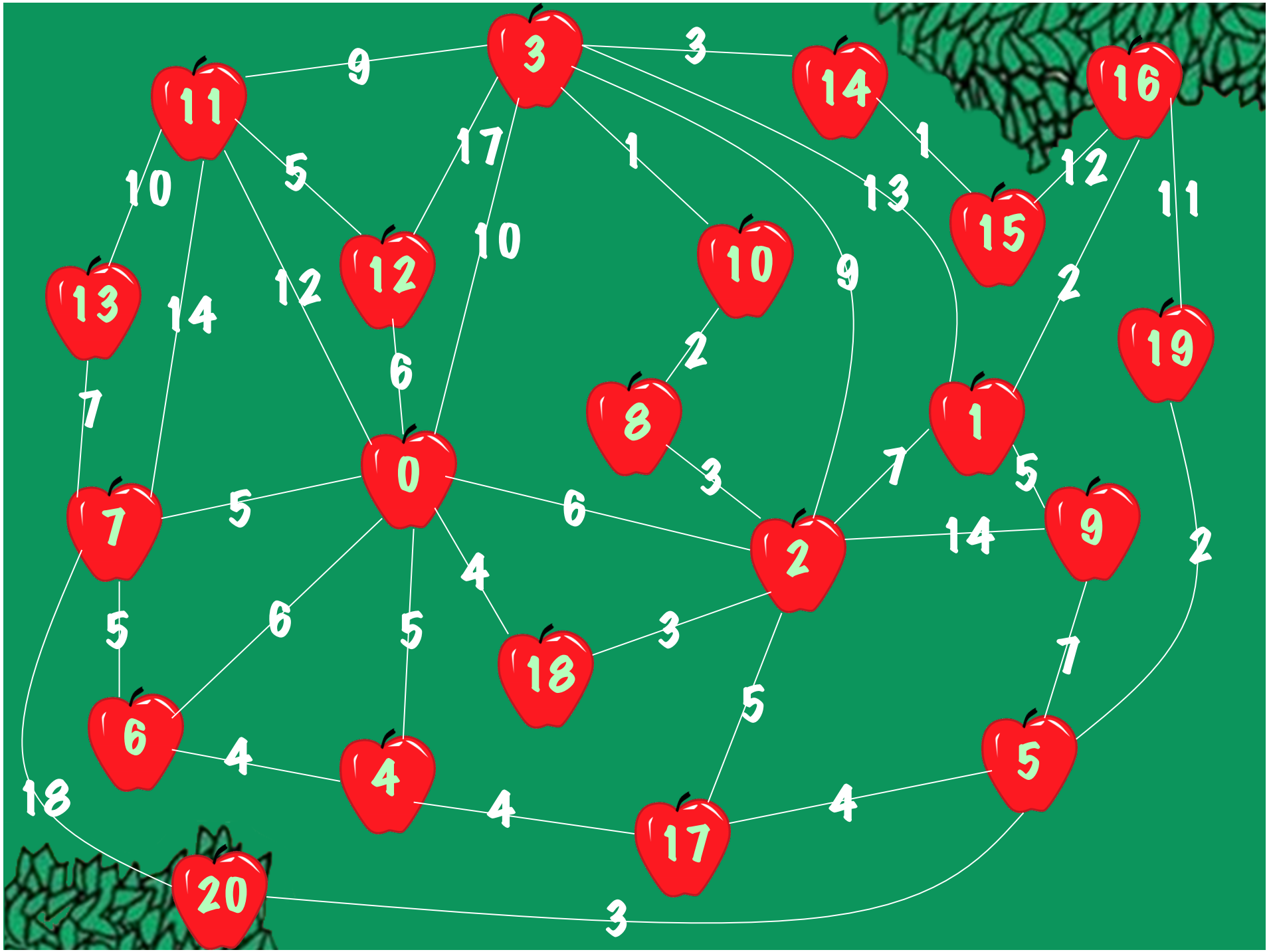


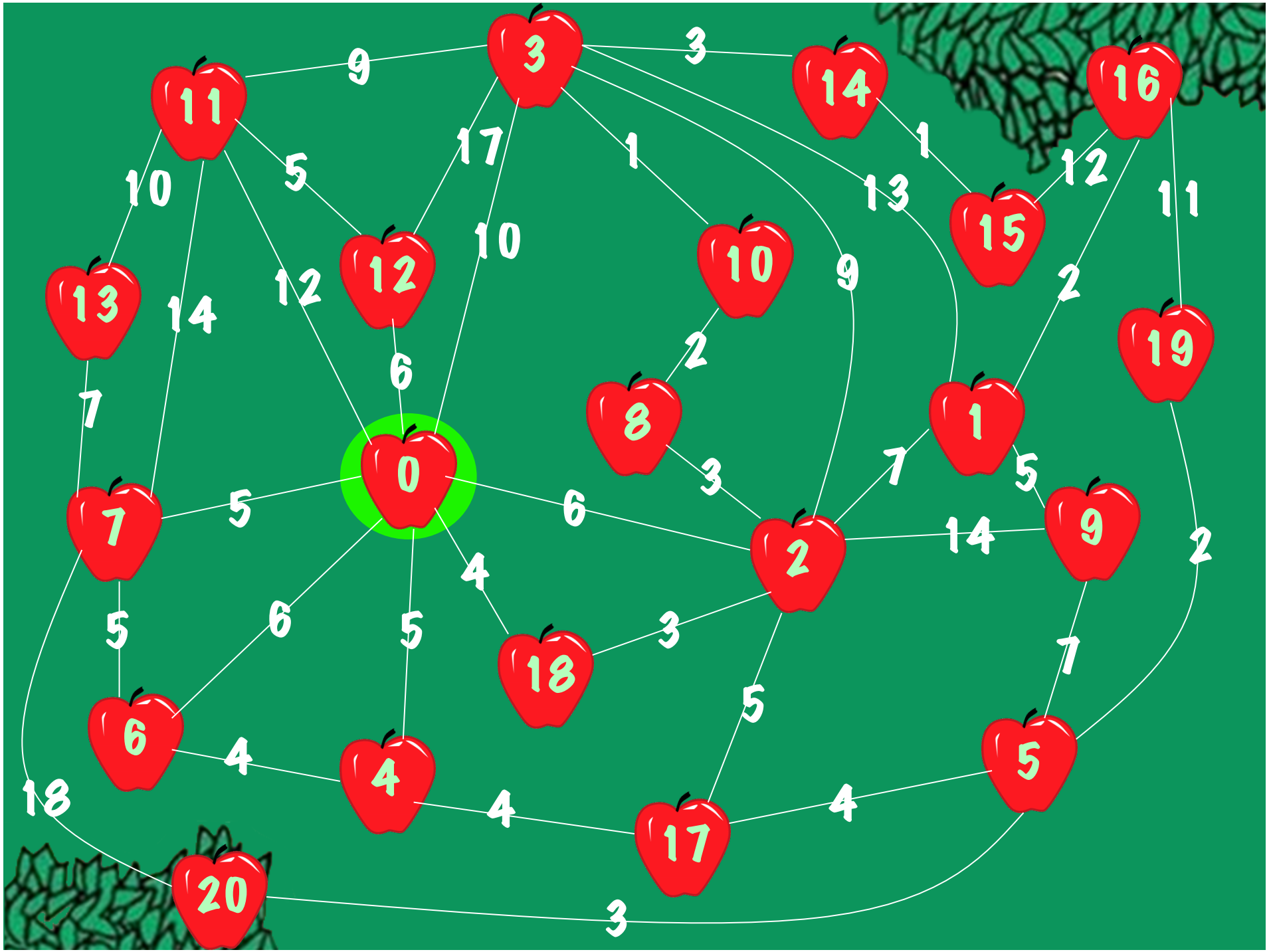
One more thing: Boruvka Step

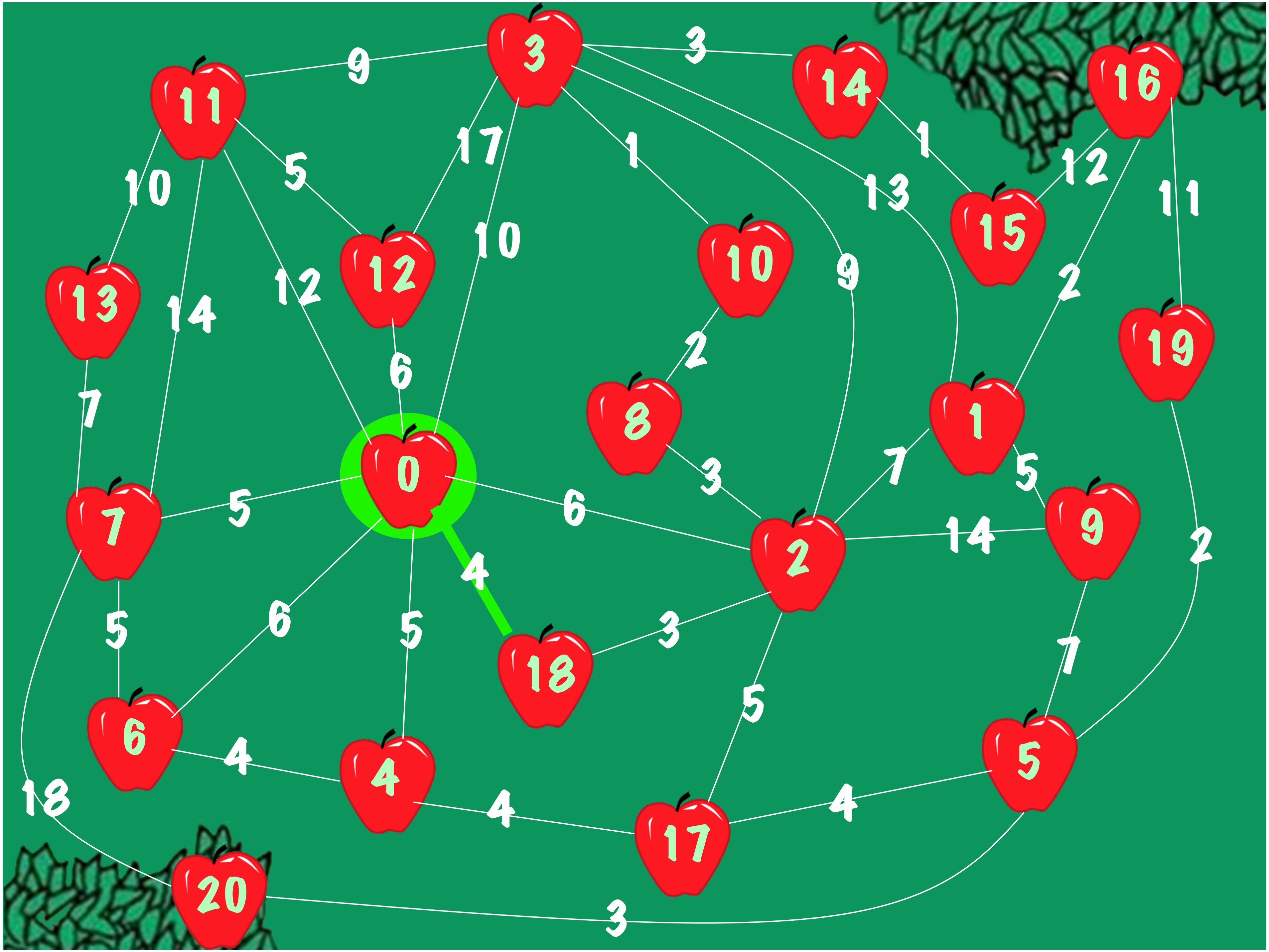
- For each vertex v, select the minimum-weight edge incident to v.
- The selected edges defined connected components; coalesce these components into single vertices.
- Delete all resulting loops, isolated vertices, and all but the lowest-weight edge among each set of multiple edges.

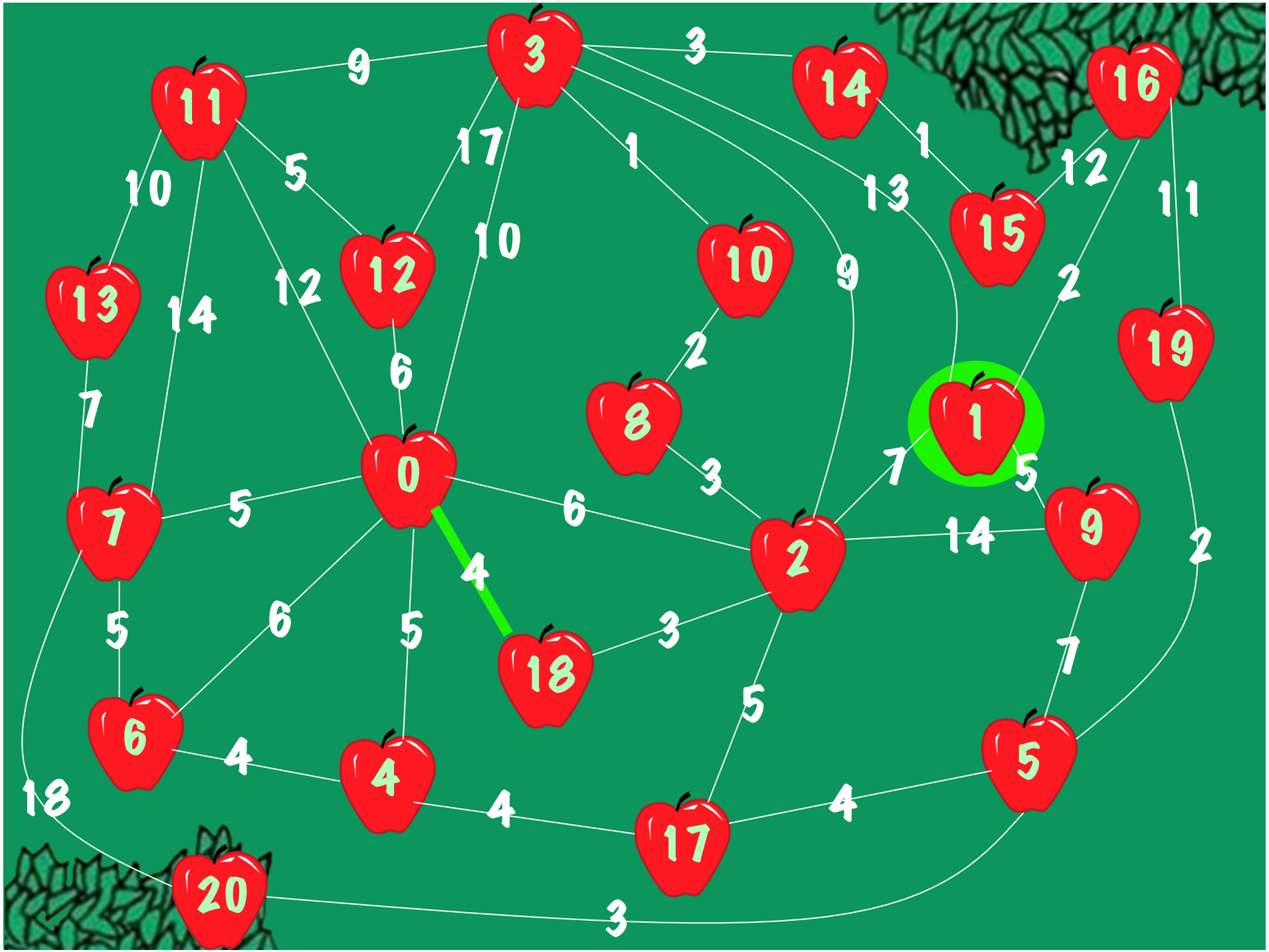
The Algorithm

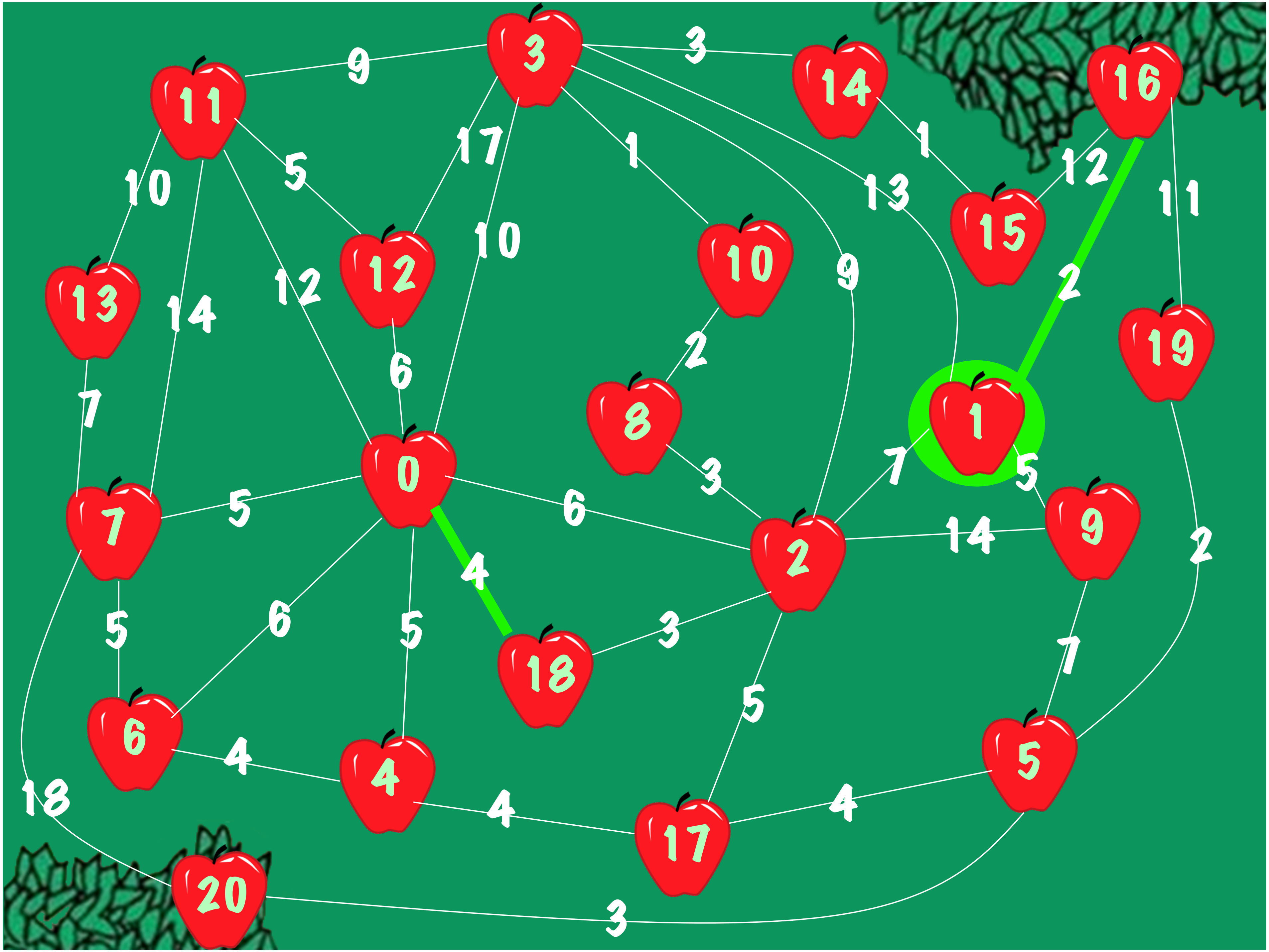
- STEP 1: Perform two Boruvka steps.
- STEP 2: Choose a subgraph H by selecting each edge independently with probability 1/2. Apply the algorithm to the subgraph, producing a minimum spanning forest F of H. Throw away all **F-heavy** edges in the entire graph.
- STEP 3: Recurse on the remaining graph to compute a spanning forest F'. Return edges contracted in STEP 1 along with the edges of F'

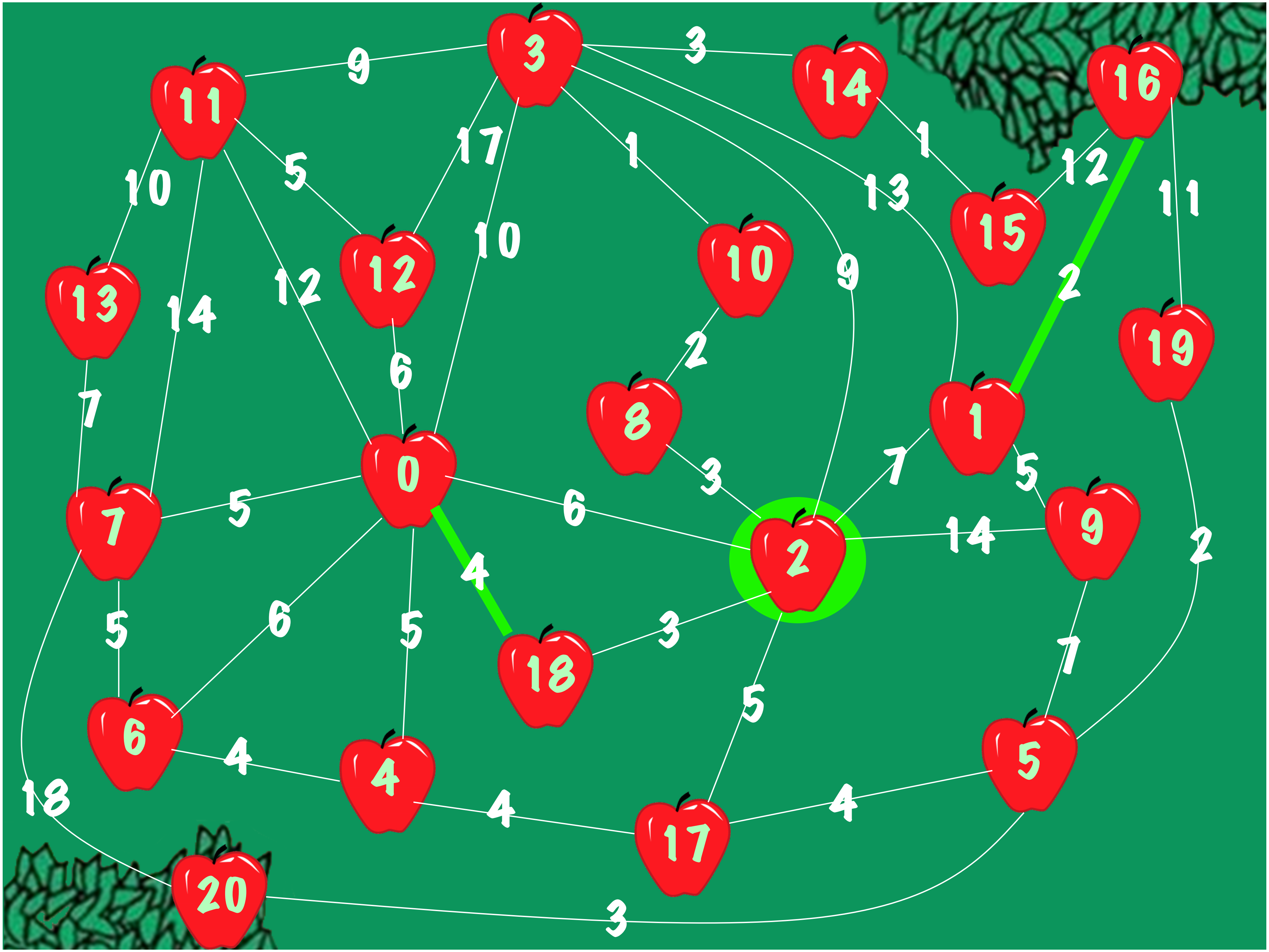

3
11
14
16
15
12
10
13
19
8
1
0
7
2
9
18
6
4
17
5
20
9
3
14
1
10
5
17
1
13
12
11
10
12
14
6
9
2
2
7
5
14
2
7
5
6
4
5
3
7
5
4
4
4
18
3

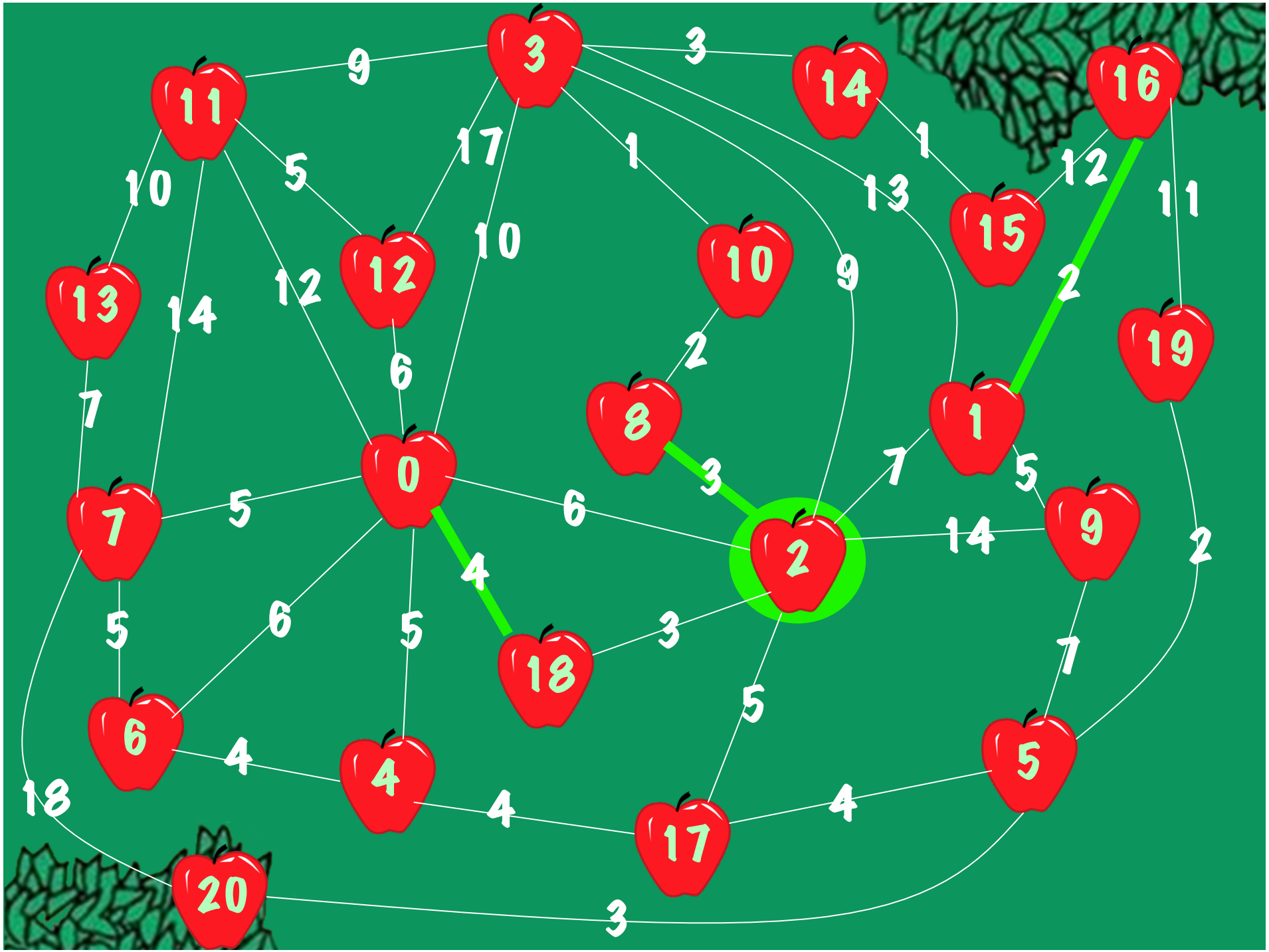

3
11
14
16
15
12
13
10
19
8
1
0
7
2
9
18
6
4
17
5
20
9
3
14
1
10
17
1
13
12
5
12
11
10
13
14
12
6
9
2
2
7
5
7
5
6
14
2
4
5
6
5
3
7
5
4
4
18
4
3

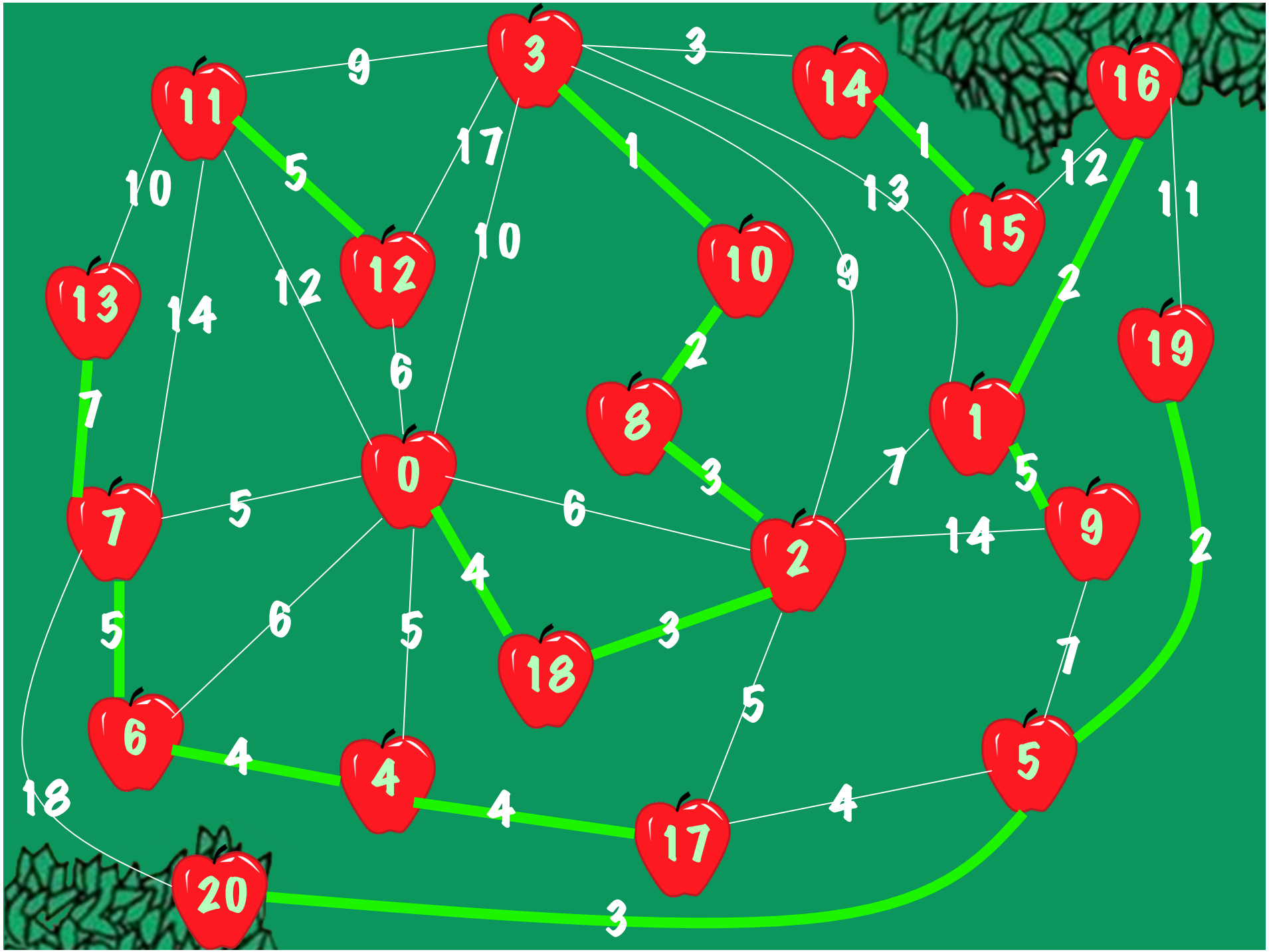

3
9
3
14
16
11
17
1
1
5
12
10
13
11
15
10
12
10
13
12
14
9
2
19
2
6
8
7
1
0
7
5
3
5
6
7
9
2
14
2
4
5
6
5
3
18
7
5
6
4
4
5
4
18
17
20
3

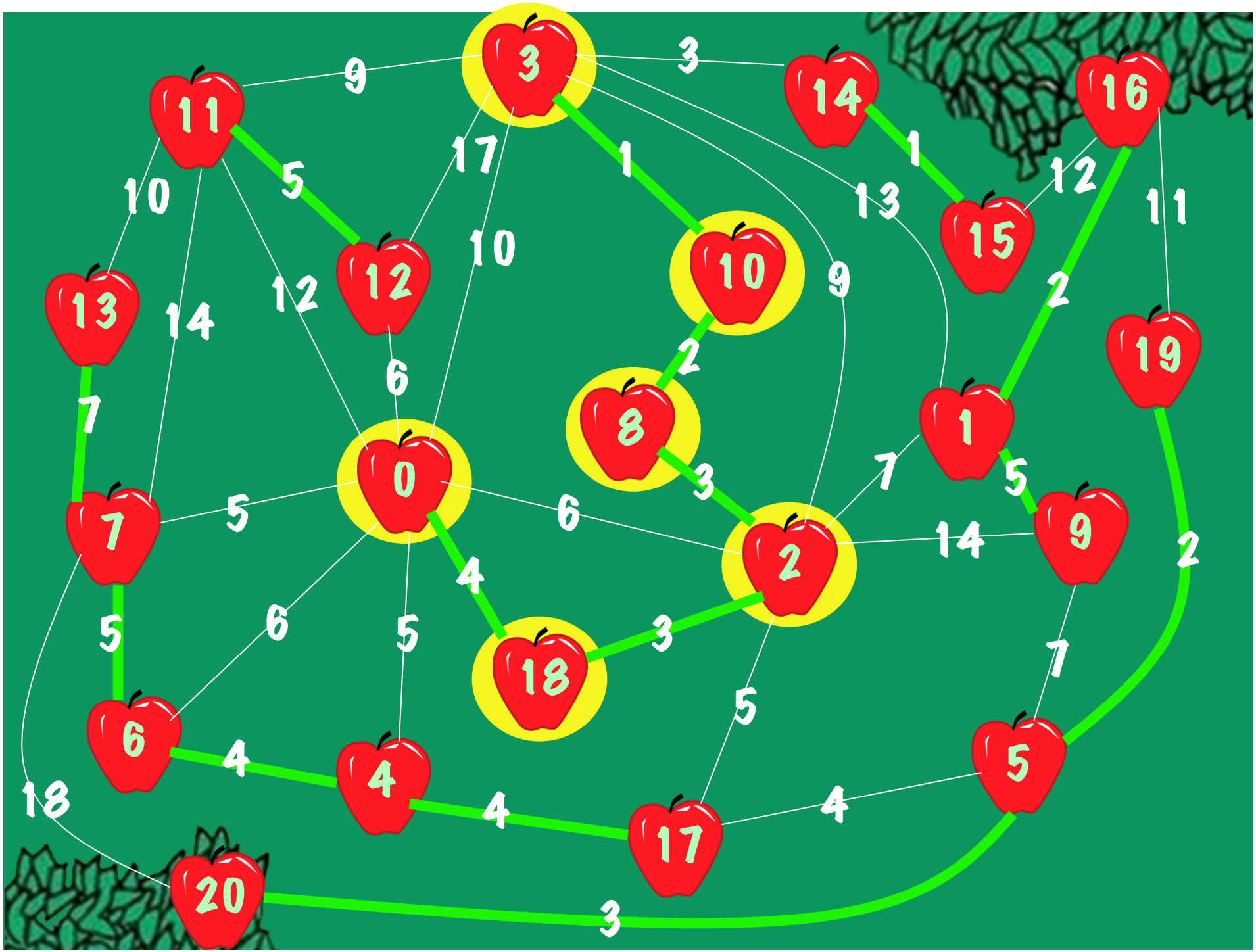

3
11
14
16
15
12
10
13
12
10
8
1
19
0
7
2
9
18
6
4
17
5
20
9
3
14
1
10
5
17
1
13
12
10
9
2
19
7
14
12
6
7
5
6
4
2
14
2
5
6
5
4
3
7
5
6
4
4
4
18
3

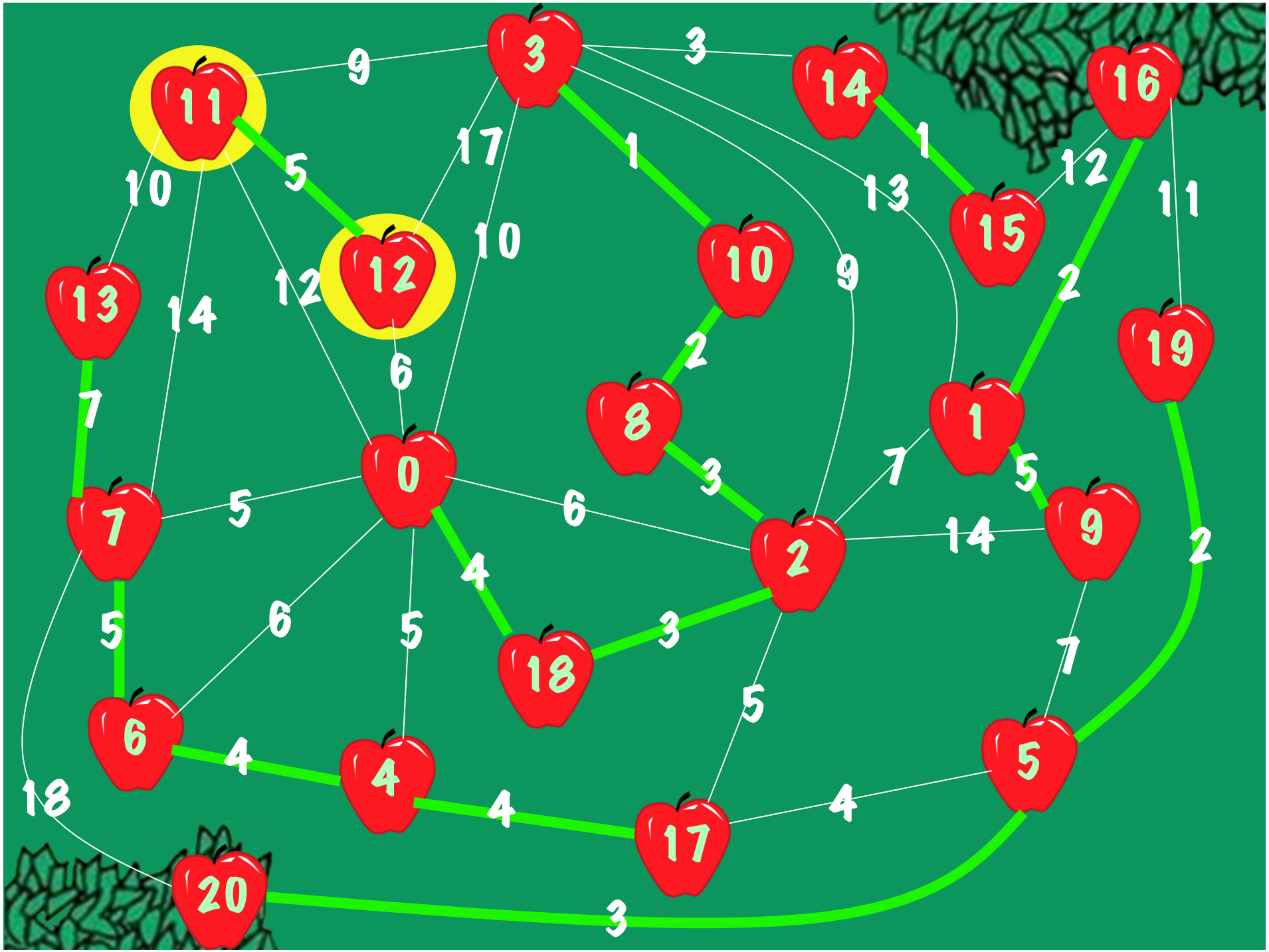

3
9
3
14
16
11
17
1
5
10
1
12
13
15
10
2
11
13
14
12
12
10
9
19
2
6
8
1
7
0
7
3
5
5
6
2
14
9
2
4
6
5
18
3
5
7
7
5
6
5
4
4
4
17
4
18
20
3

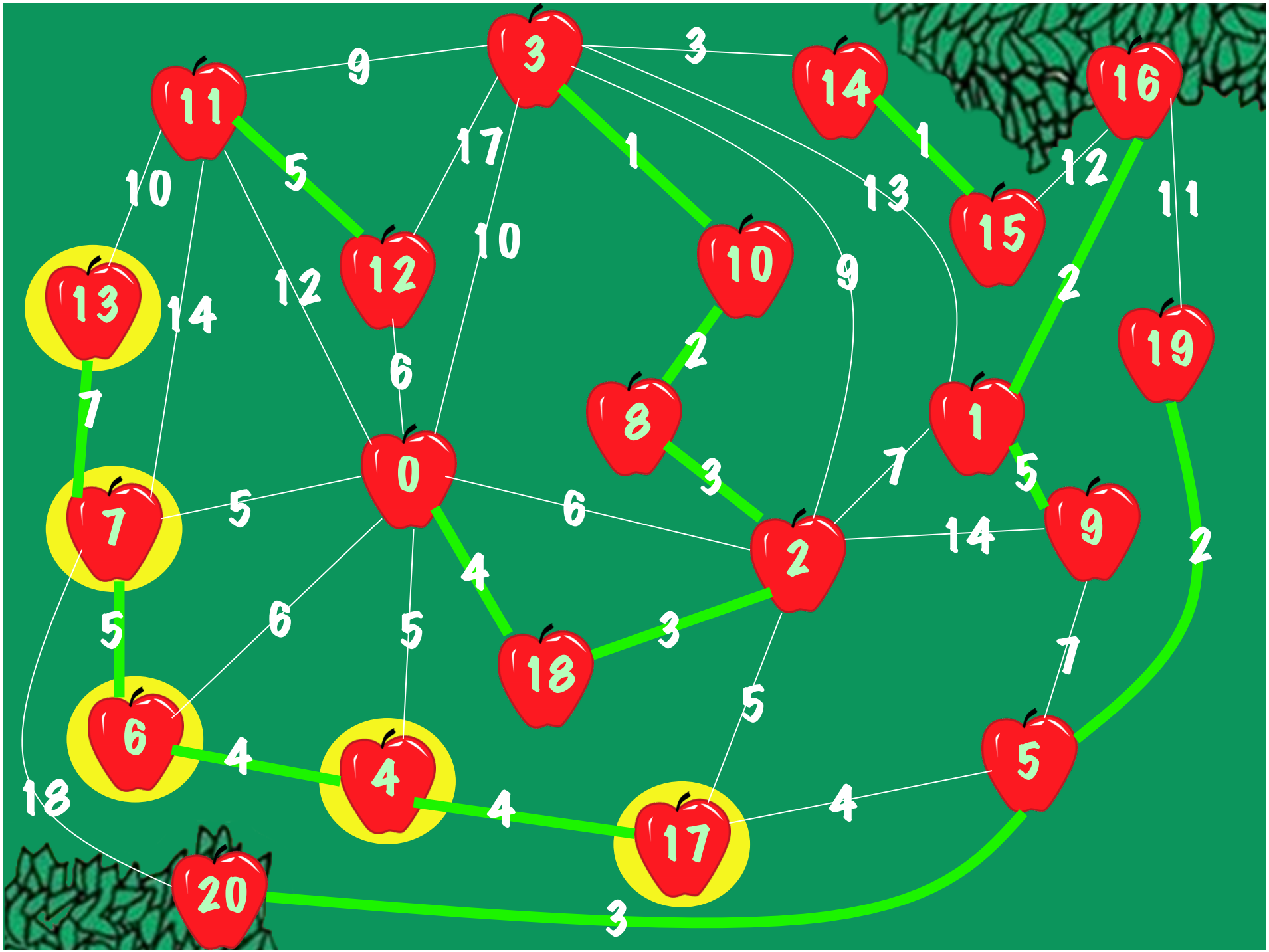

3
9
3
14
16
11
17
1
5
12
10
13
11
10
15
12
10
2
13
14
12
9
19
2
6
8
7
1
7
5
3
0
7
5
6
14
9
2
2
4
5
6
5
3
7
18
5
6
4
5
18
4
4
17
4
20
3

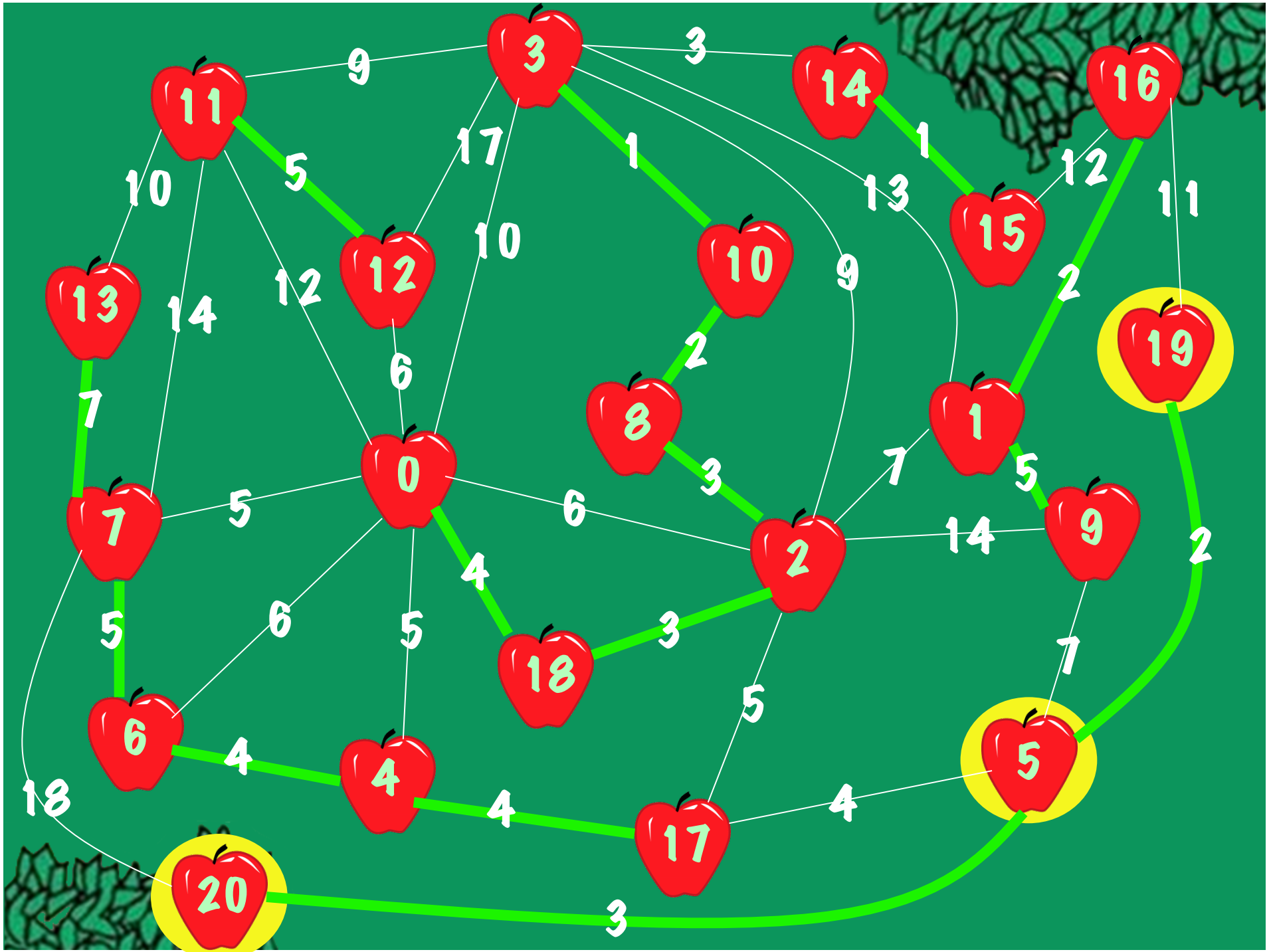

3
11
14
16
15
12
10
13
19
8
1
0
7
2
9
18
6
4
17
5
20
9
3
3
10
5
17
1
13
1
12
11
10
14
12
6
7
2
2
7
5
5
14
2
4
6
5
3
7
5
4
4
4
18
3

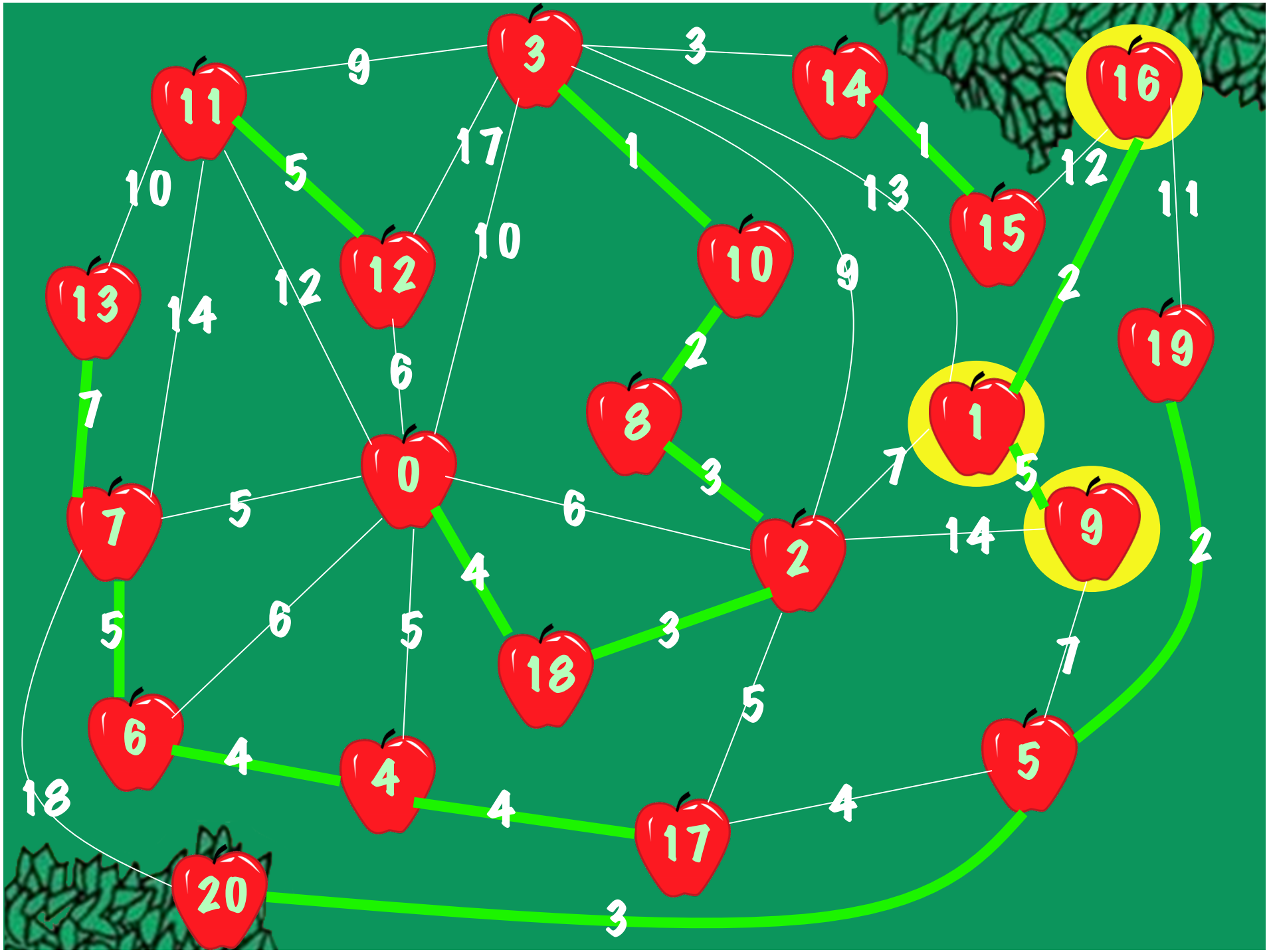

3
11
14
16
12
15
10
13
19
8
1
0
7
9
2
18
6
4
17
5
20
9
3
14
16
10
5
17
1
13
12
11
13
14
12
10
9
12
2
6
2
19
7
7
5
6
5
14
2
5
6
5
4
3
7
5
6
4
4
4
18
3

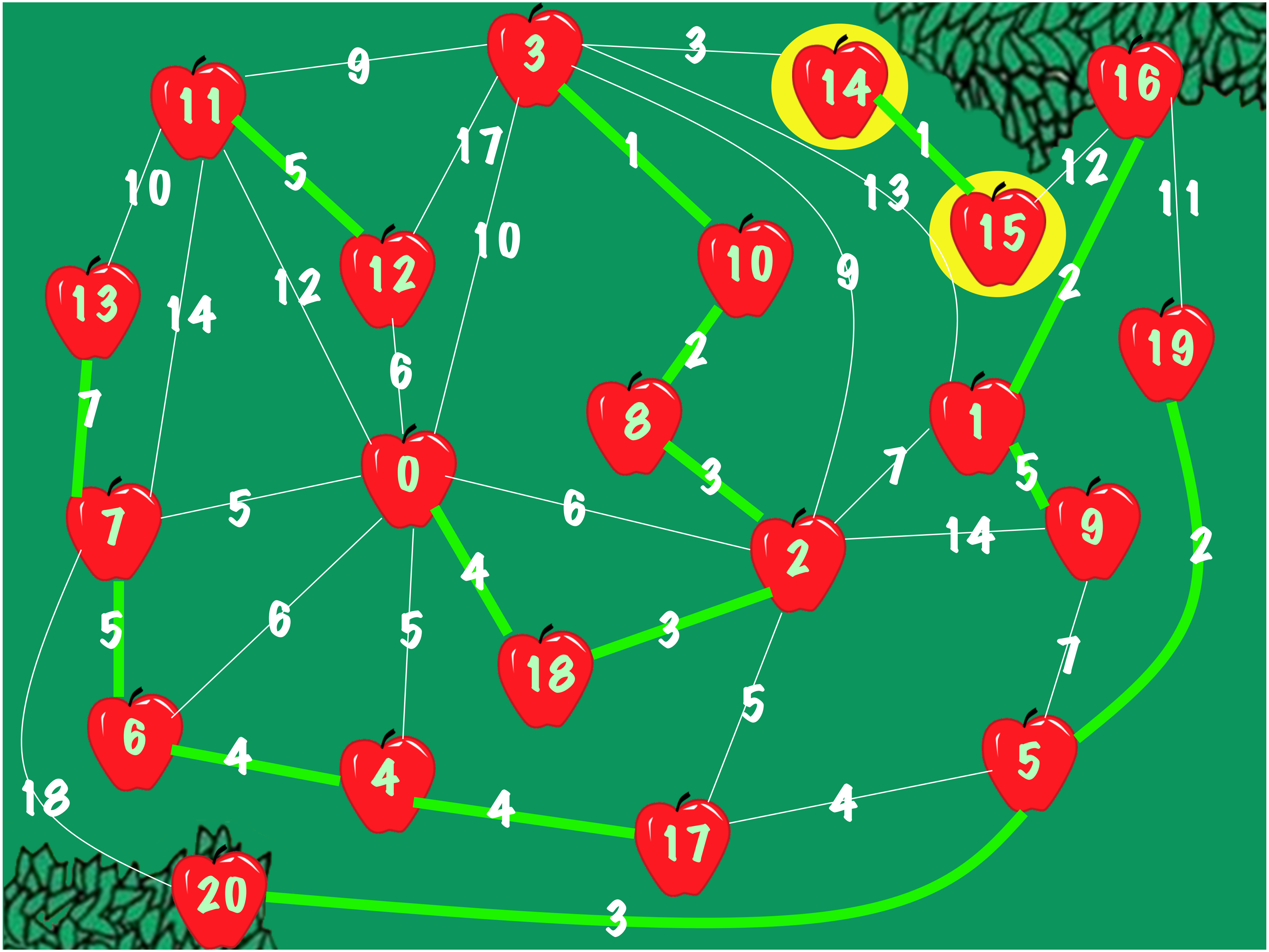

3
11
14
16
15
10
12
13
8
1
19
0
7
9
2
18
6
4
17
5
20
9
3
14
1
16
10
5
17
1
13
12
11
13
14
12
10
9
12
2
6
2
7
5
6
14
2
7
5
4
6
5
3
7
5
4
4
4
18
3


11
3
14
16
15
12
10
13
19
8
1
0
7
9
2
18
6
4
17
5
20
9
3
3
10
5
17
1
13
1
12
11
10
12
13
14
6
9
2
7
2
5
7
5
6
14
2
4
5
6
5
3
7
5
4
4
4
18
3

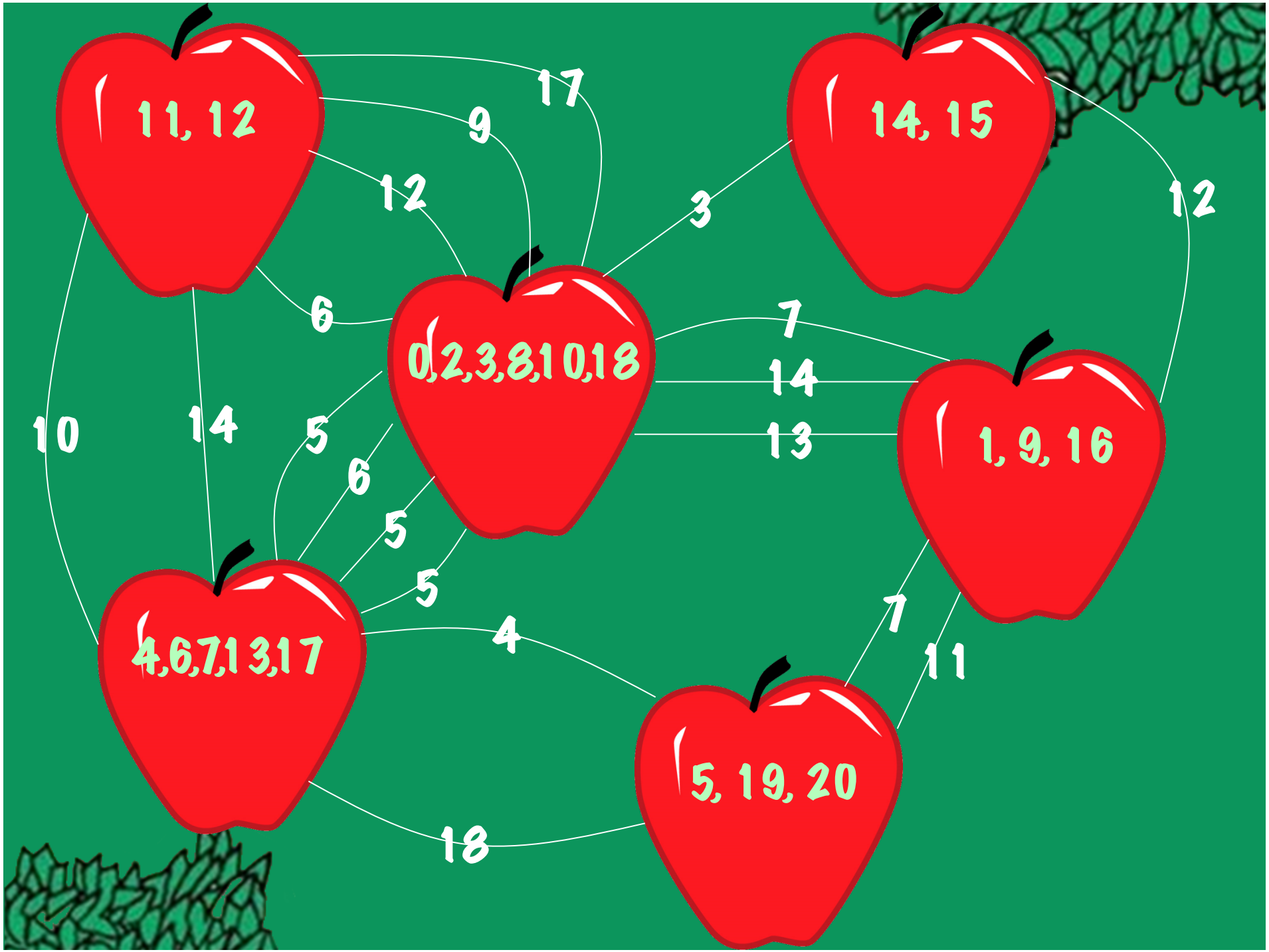

3
11
14
16
15
12
10
13
0
8
1
19
7
2
9
18
6
4
17
5
20
9
3
3
10
5
17
1
13
1
12
11
10
12
14
6
2
7
5
6
7
5
14
2
4
6
5
3
7
5
4
4
4
18
3

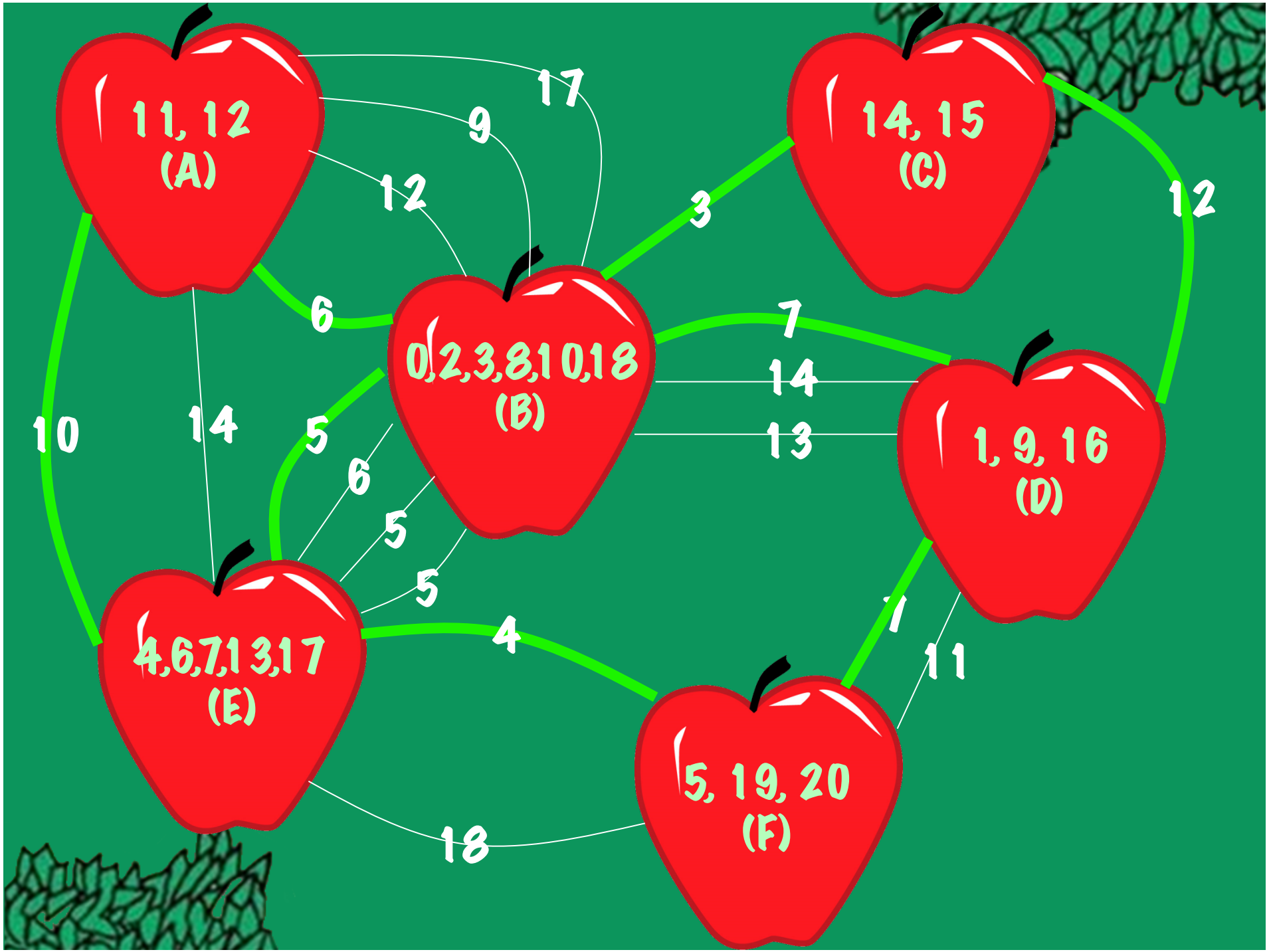

3
11
14
16
15
12
10
13
19
8
1
0
7
9
2
18
6
4
17
5
20
9
3
3
10
5
17
1
13
1
12
11
13
14
12
10
9
2
2
6
7
5
7
5
6
4
2
14
4
6
5
3
5
7
2
18
4
4
4
5
3

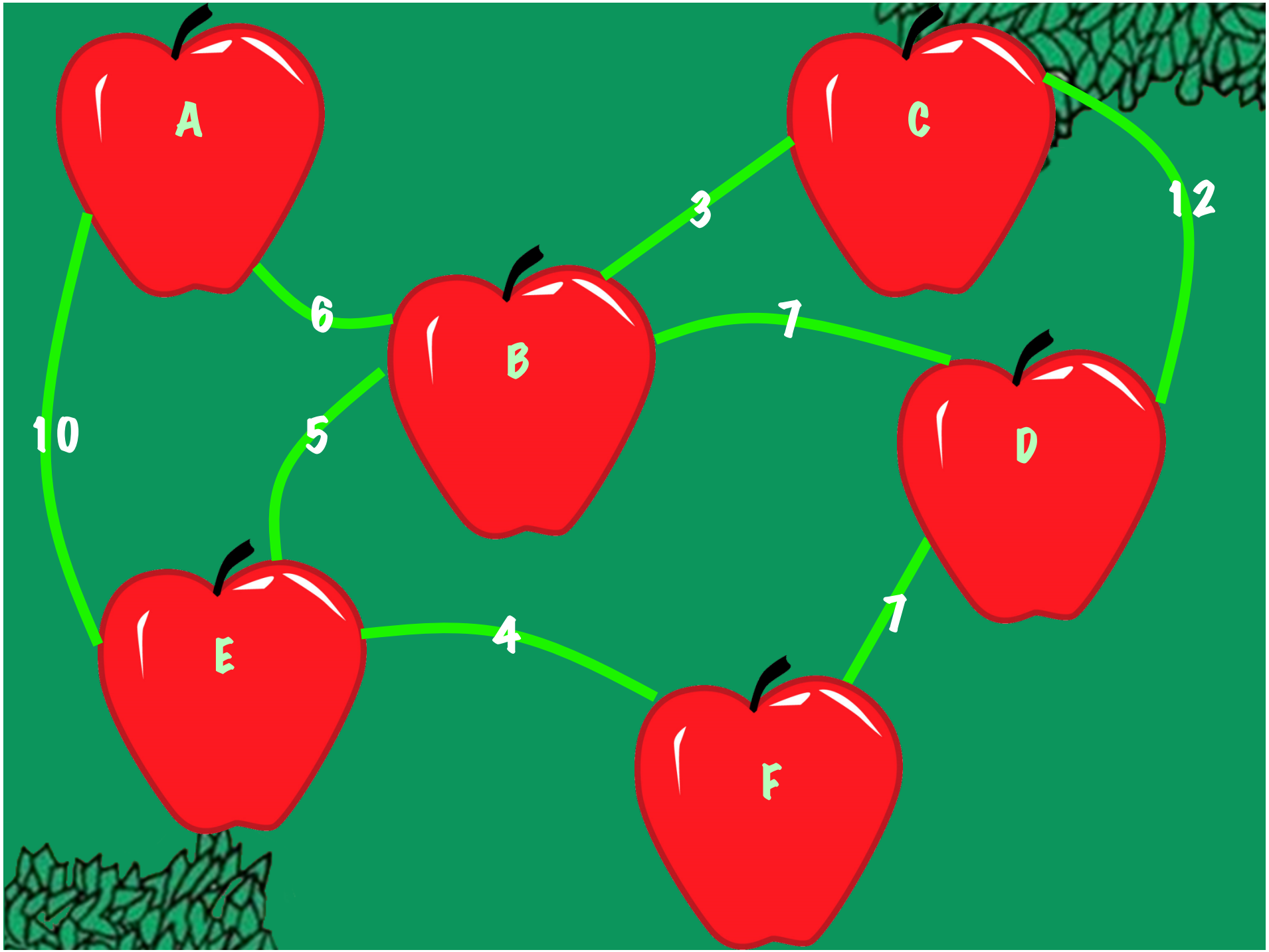

3
11
14
16
9
3
10
5
17
1
13
1
12
11
15
13
12
10
10
9
2
14
6
2
19
8
1
7
0
7
5
6
3
5
9
14
2
4
5
6
5
18
3
7
6
4
5
18
4
4
17
4
20
3

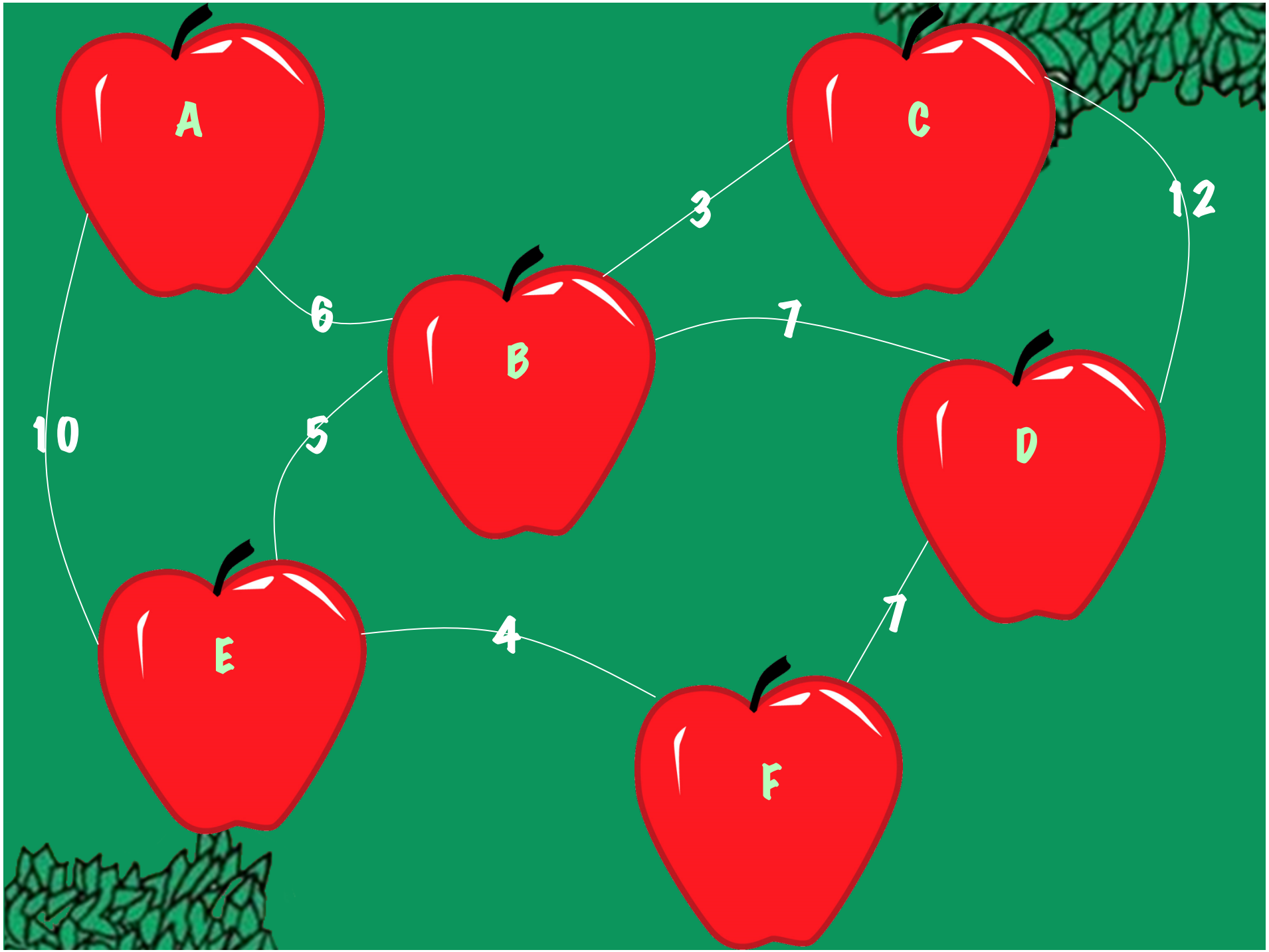


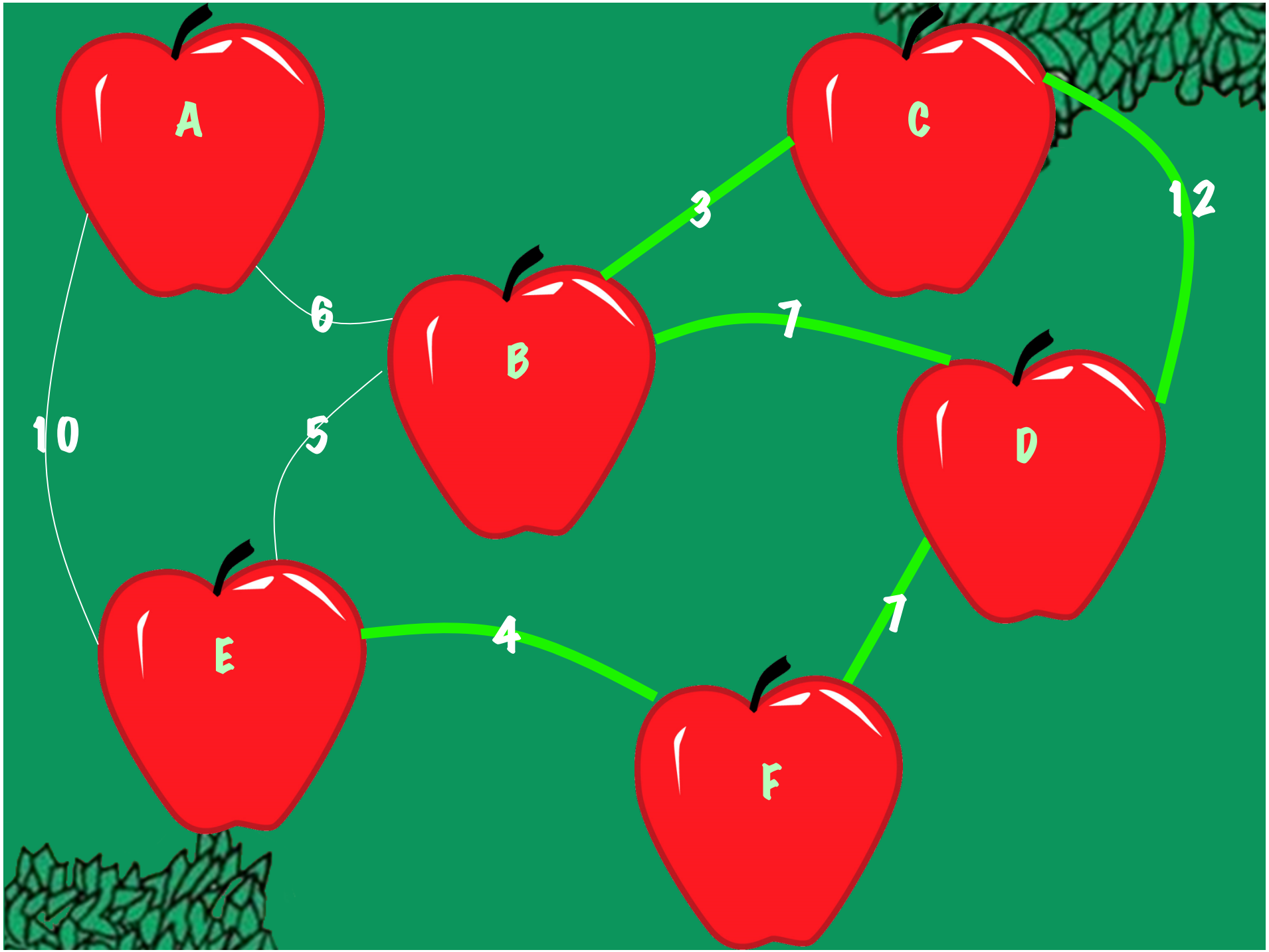
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

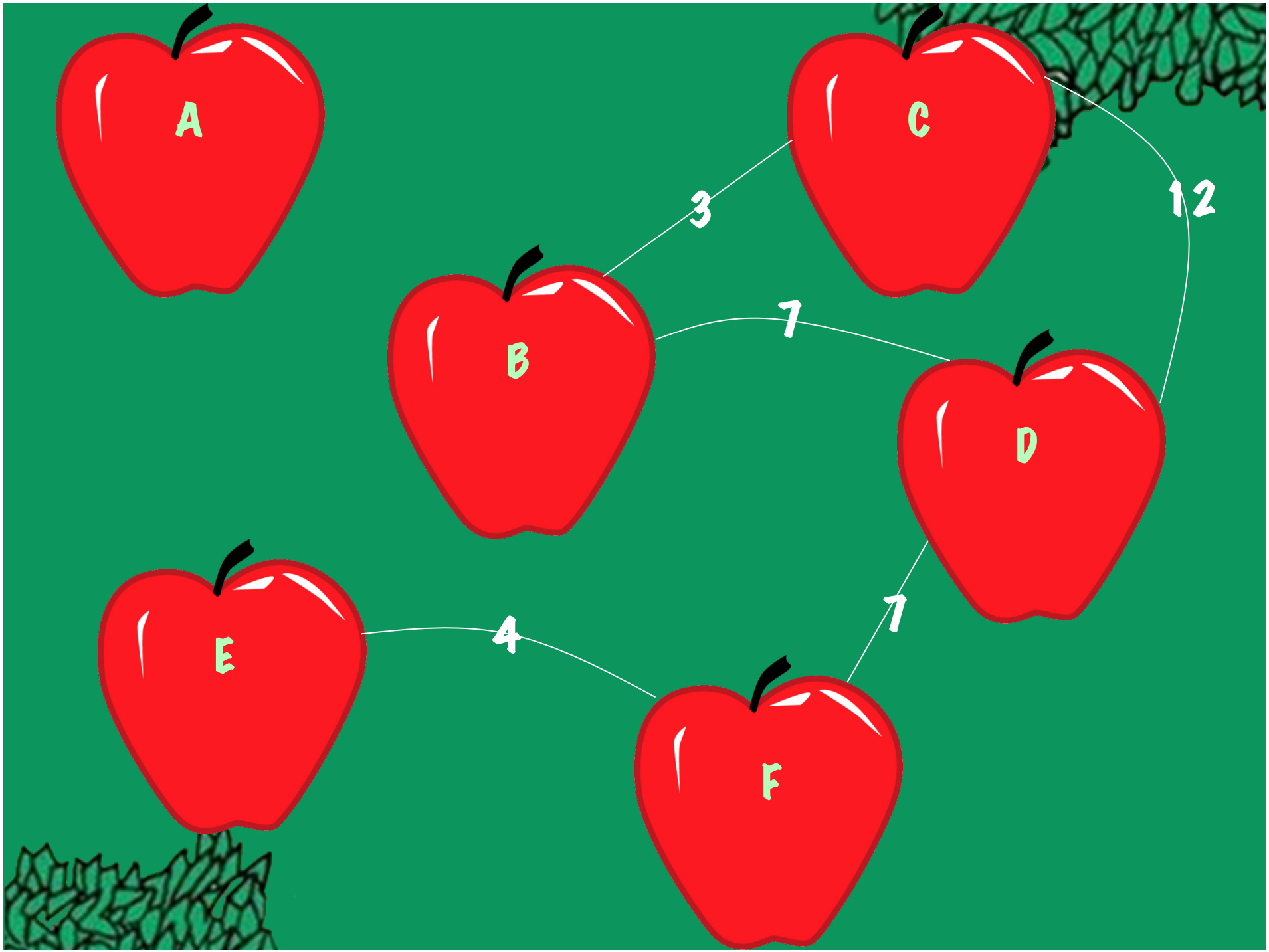

11, 12
14, 15
0, 2, 3, 8, 10, 18
1, 9, 16
4, 6, 7, 13, 17
5, 19, 20
17
9
12
3
12
6
7
14
13
10
14
5
6
5
5
4
7
11
18

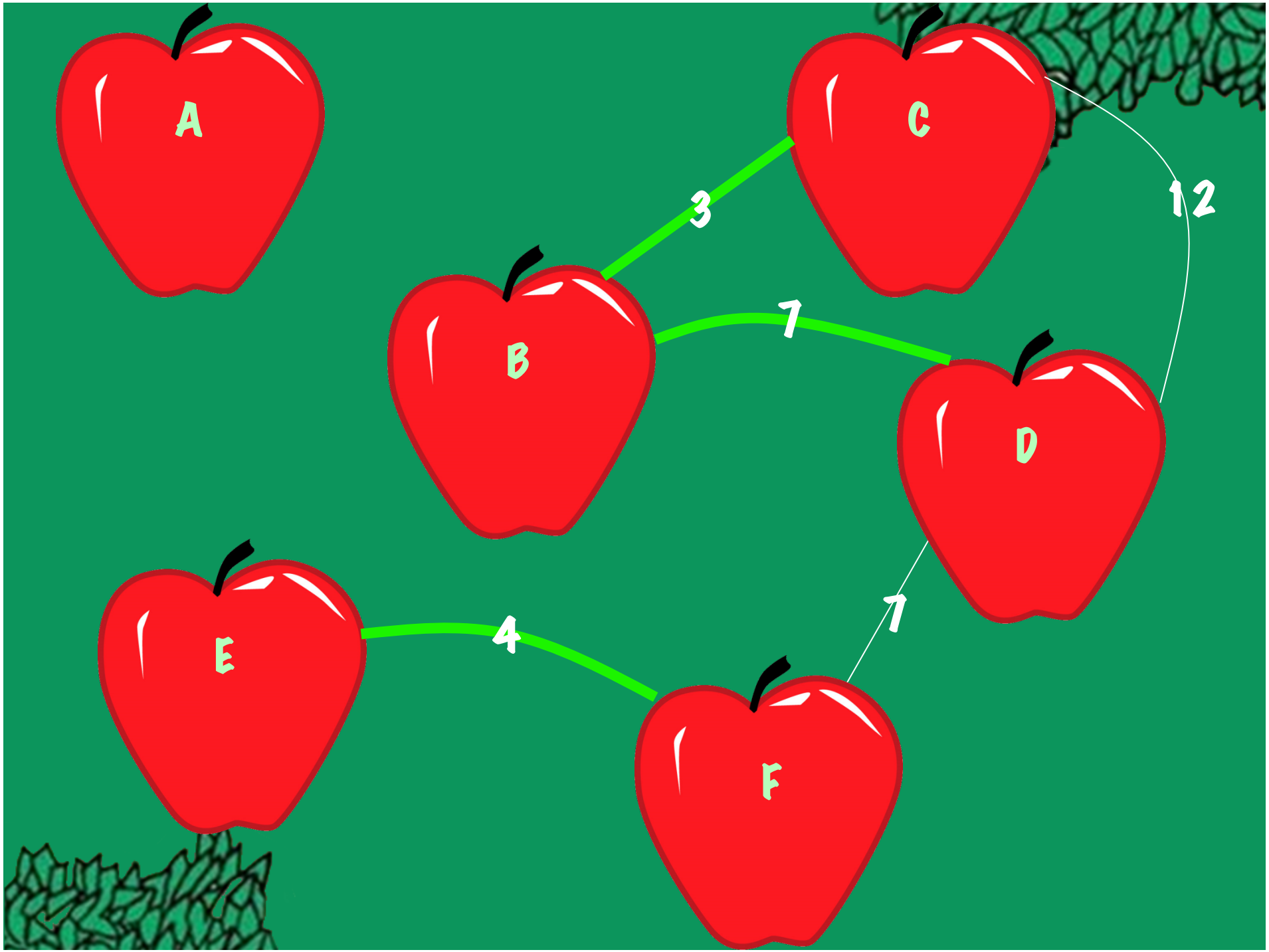

11, 12
(A)
14, 15
(C)
0,2,3,8,10,18
(B)
1, 9, 16
(D)
4,6,7,13,17
(E)
5, 19, 20
(F)
17
9
12
3
12
6
7
14
13
10
14
5
6
5
5
4
1
11
18

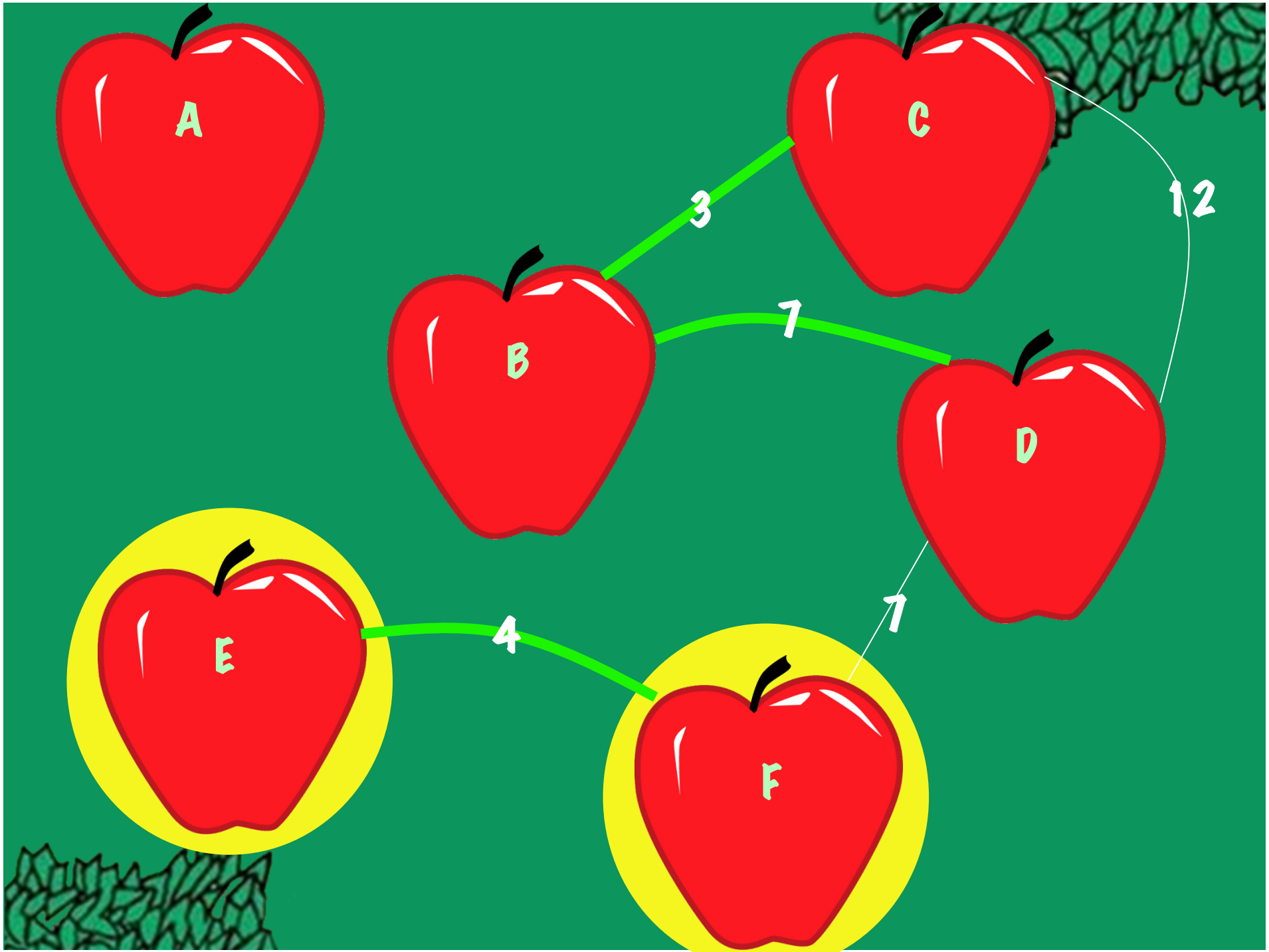

A
C
B
D
E
F
3
12
6
7
10
5
4
7

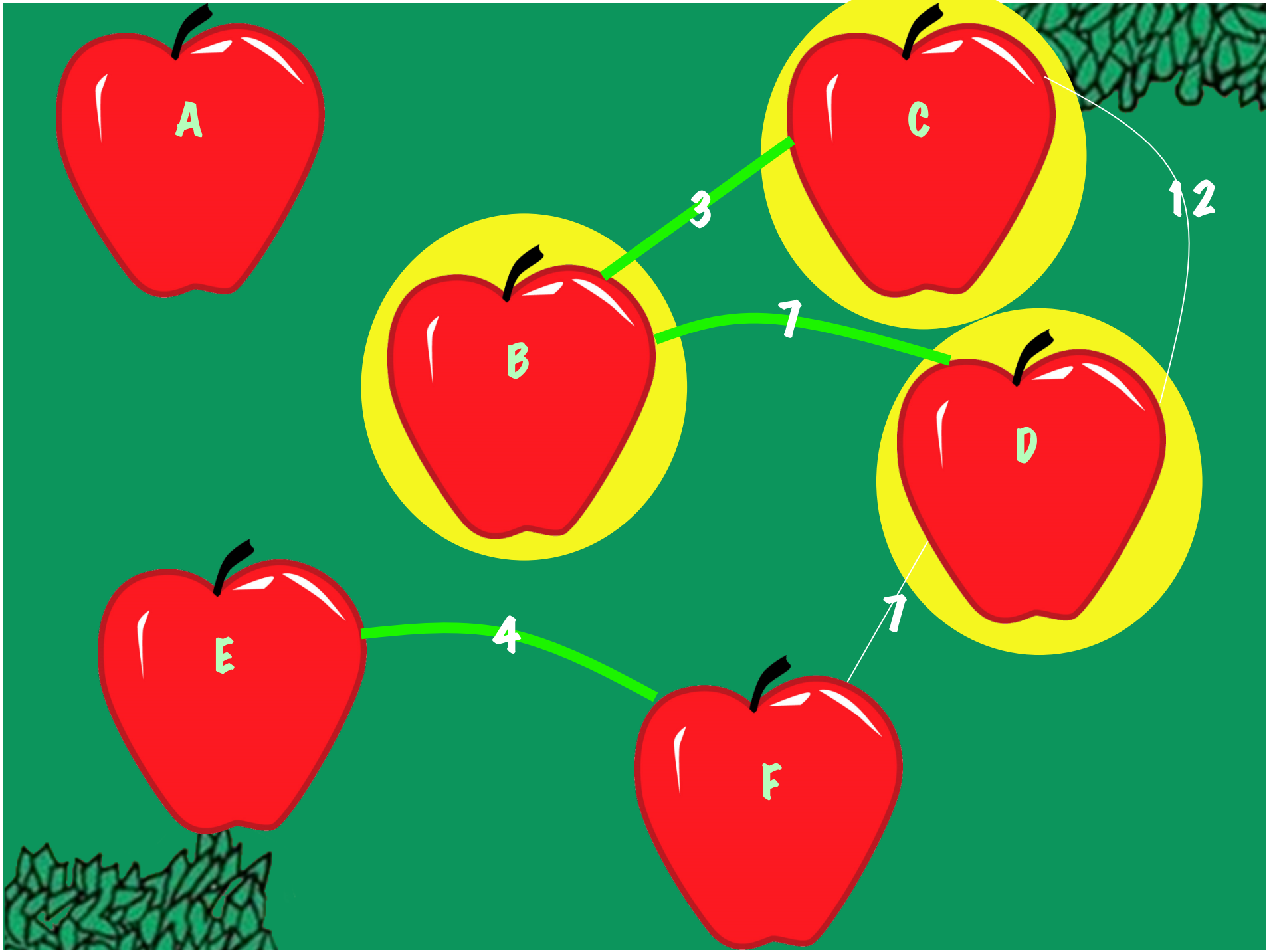

A
C
B
D
E
F
3
12
6
7
10
5
4
7

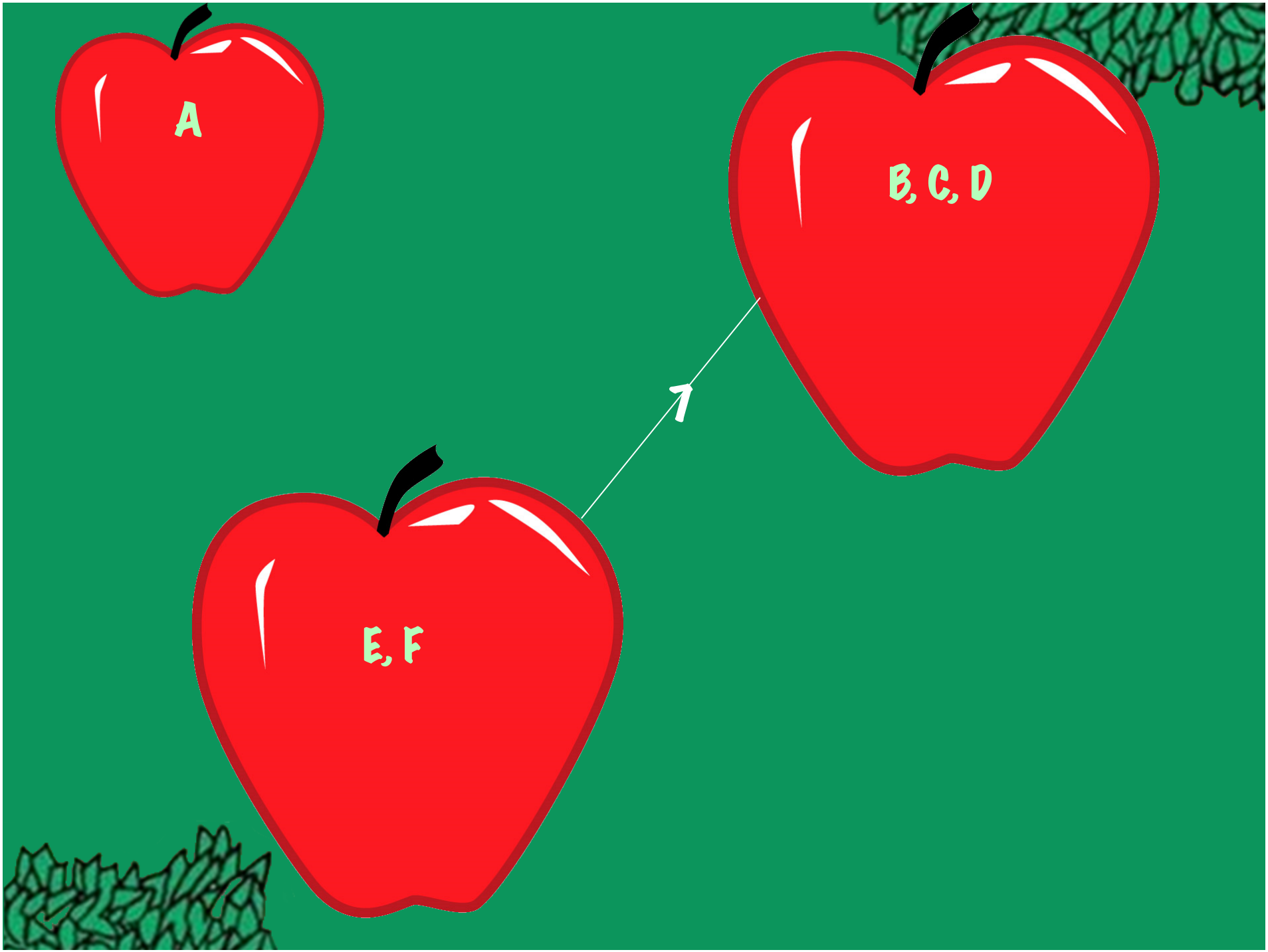

A
C
B
D
E
F
3
12
6
7
10
5
4
7

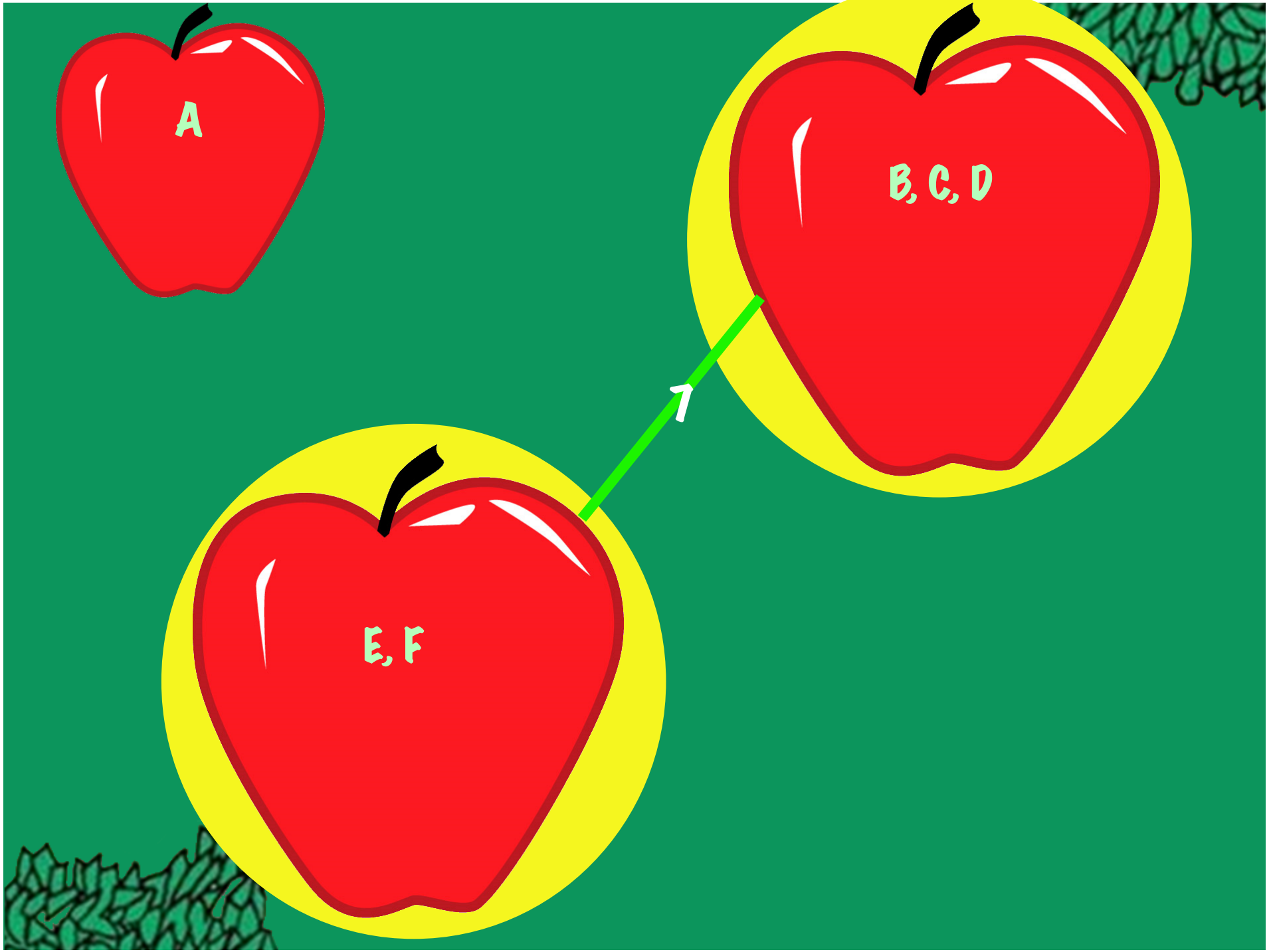

A
B
C
D
E
F
3
12
7
4
7

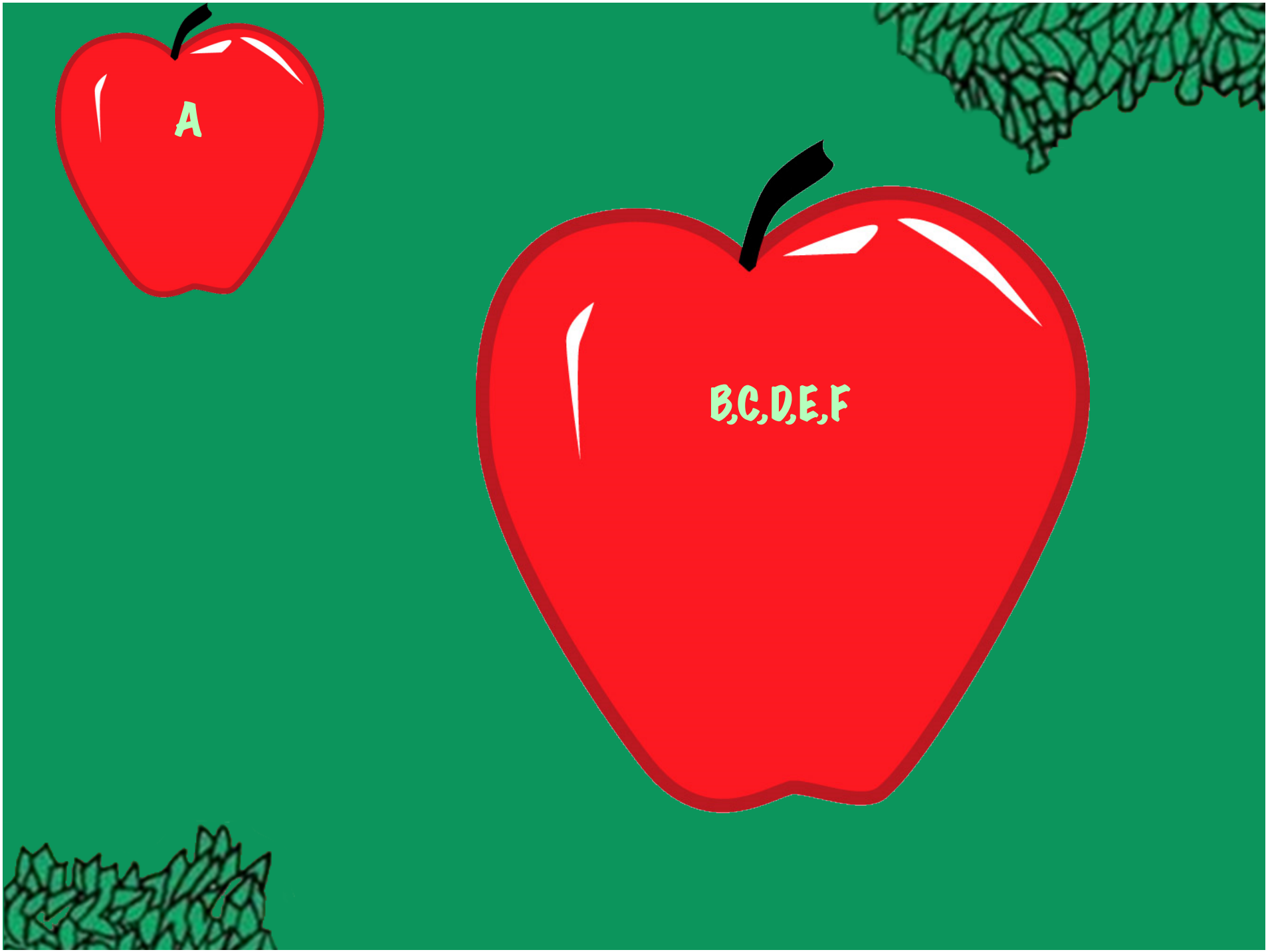

A
C
3
12
B
7
D
E
4
7
F

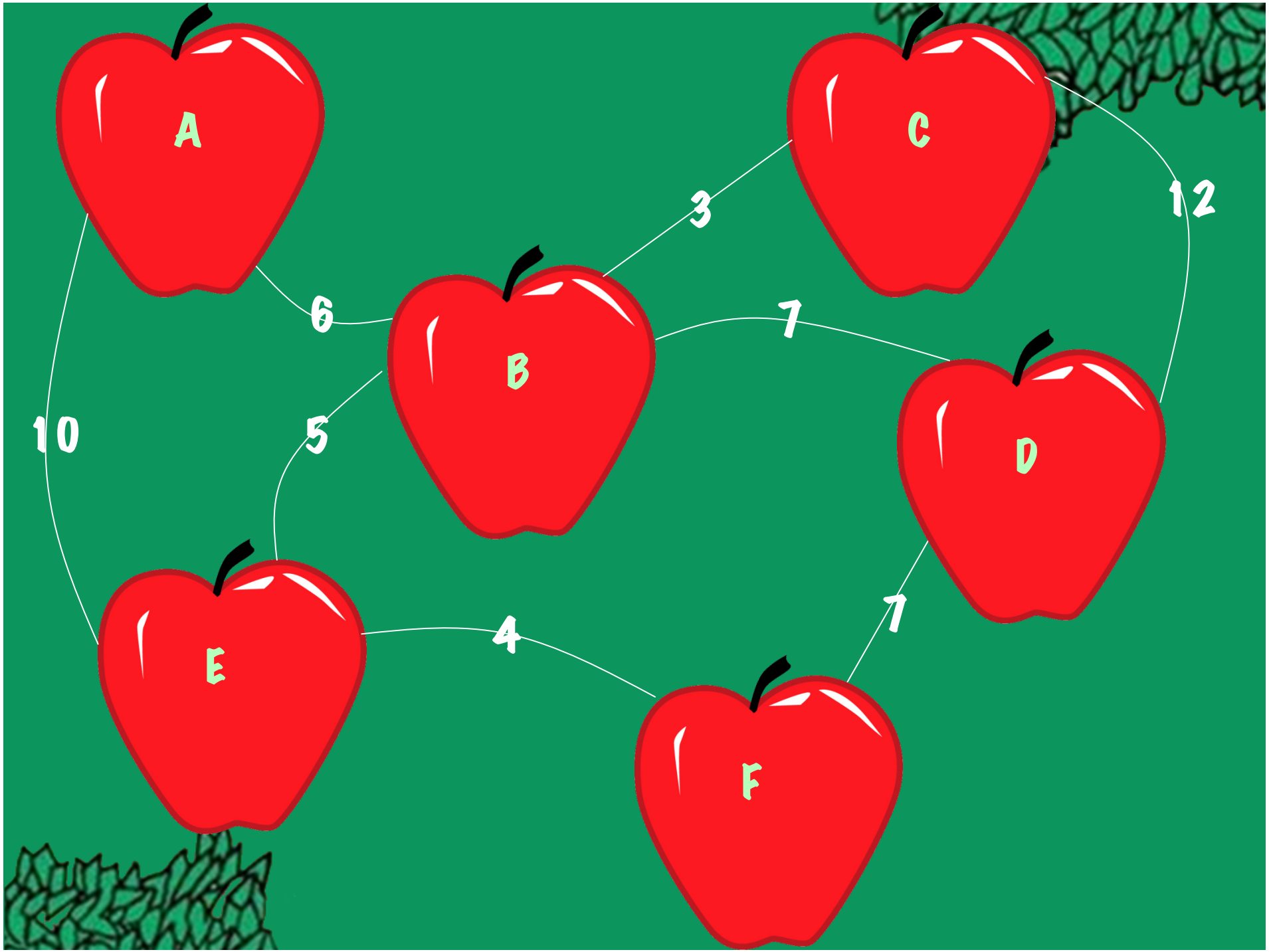

A
C
3
12
B
7
D
E
4
7
F

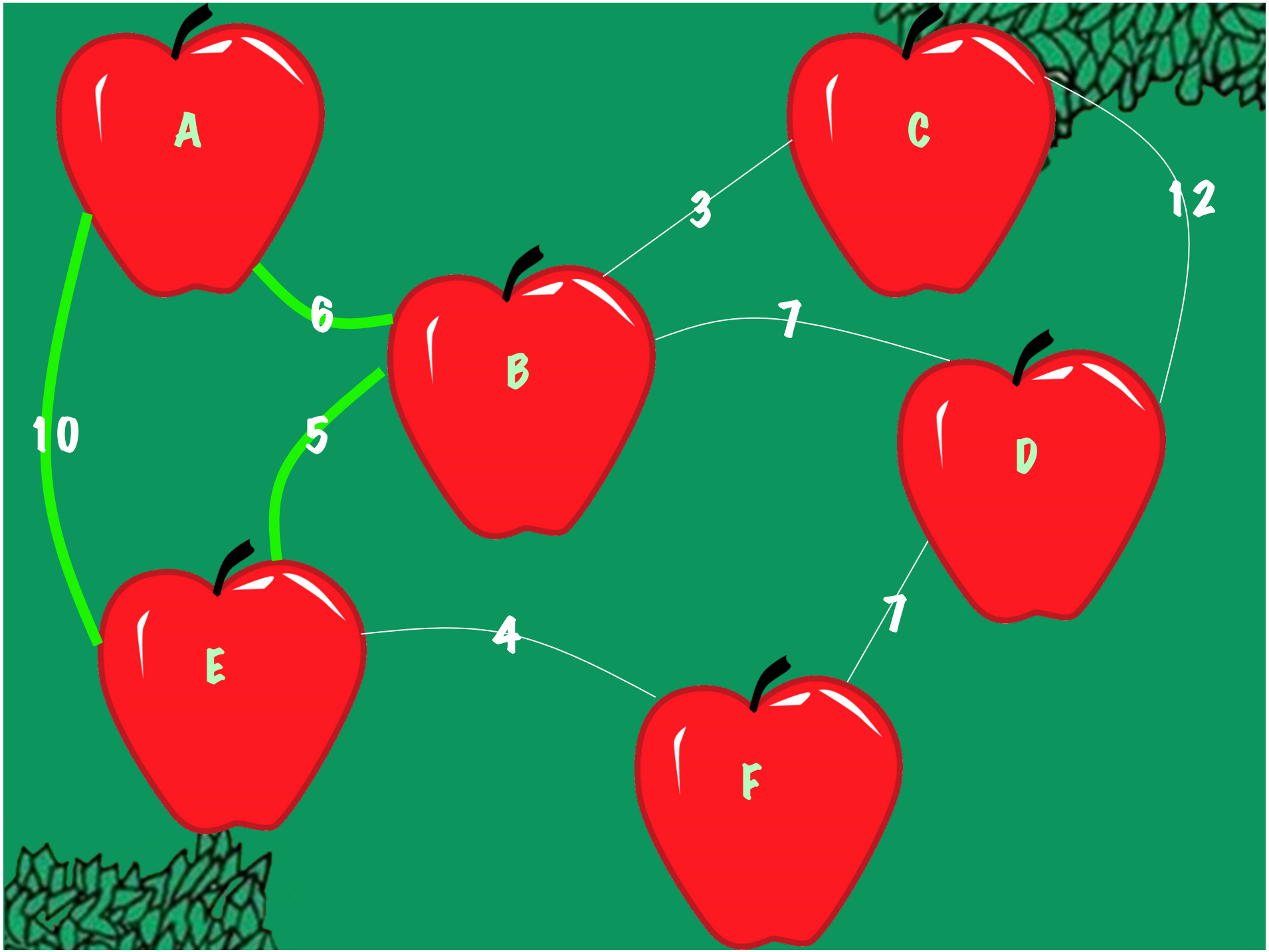

A
C
B
D
E
F
3
12
7
4
7

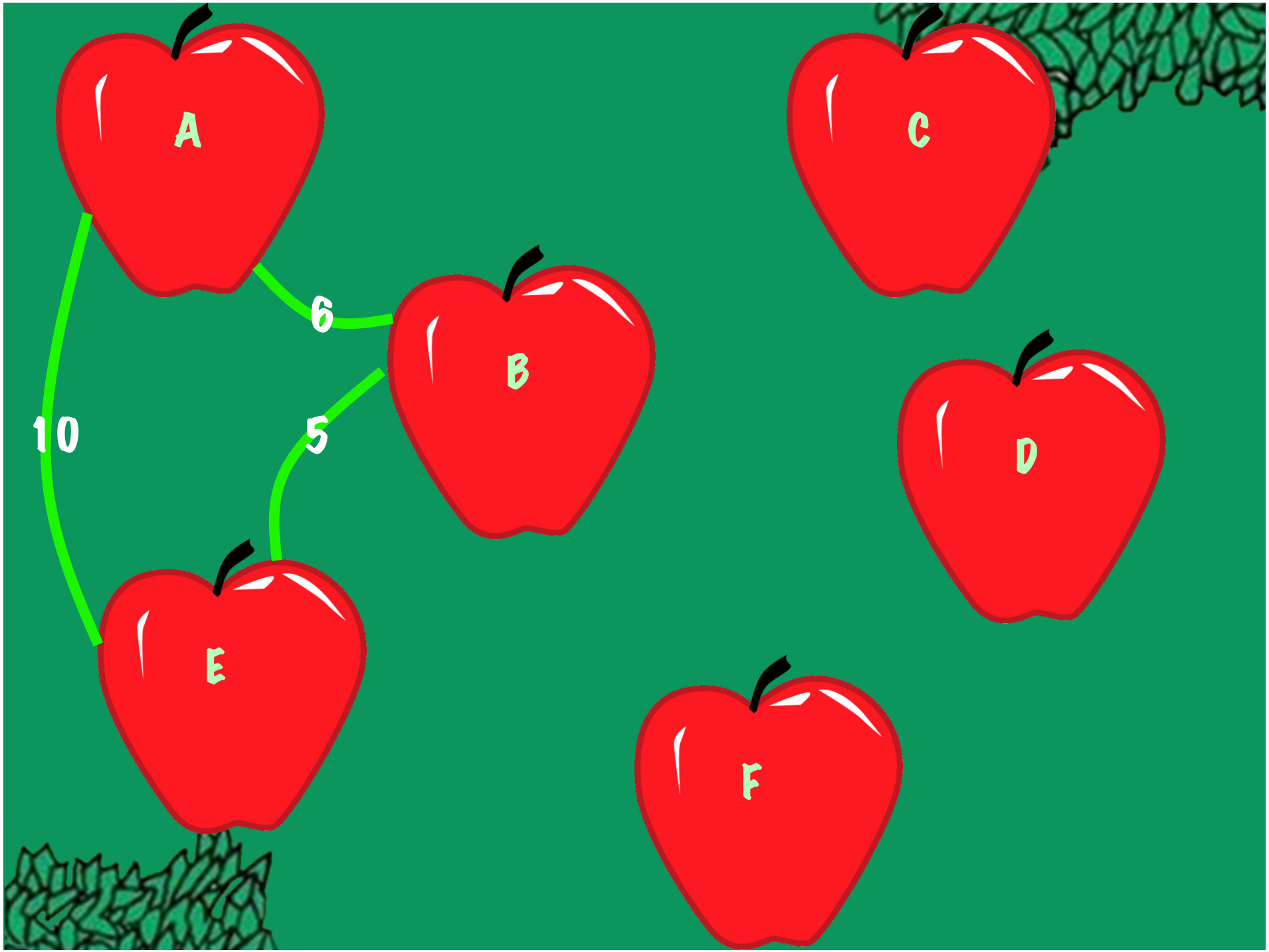

A
B, C, D
E, F

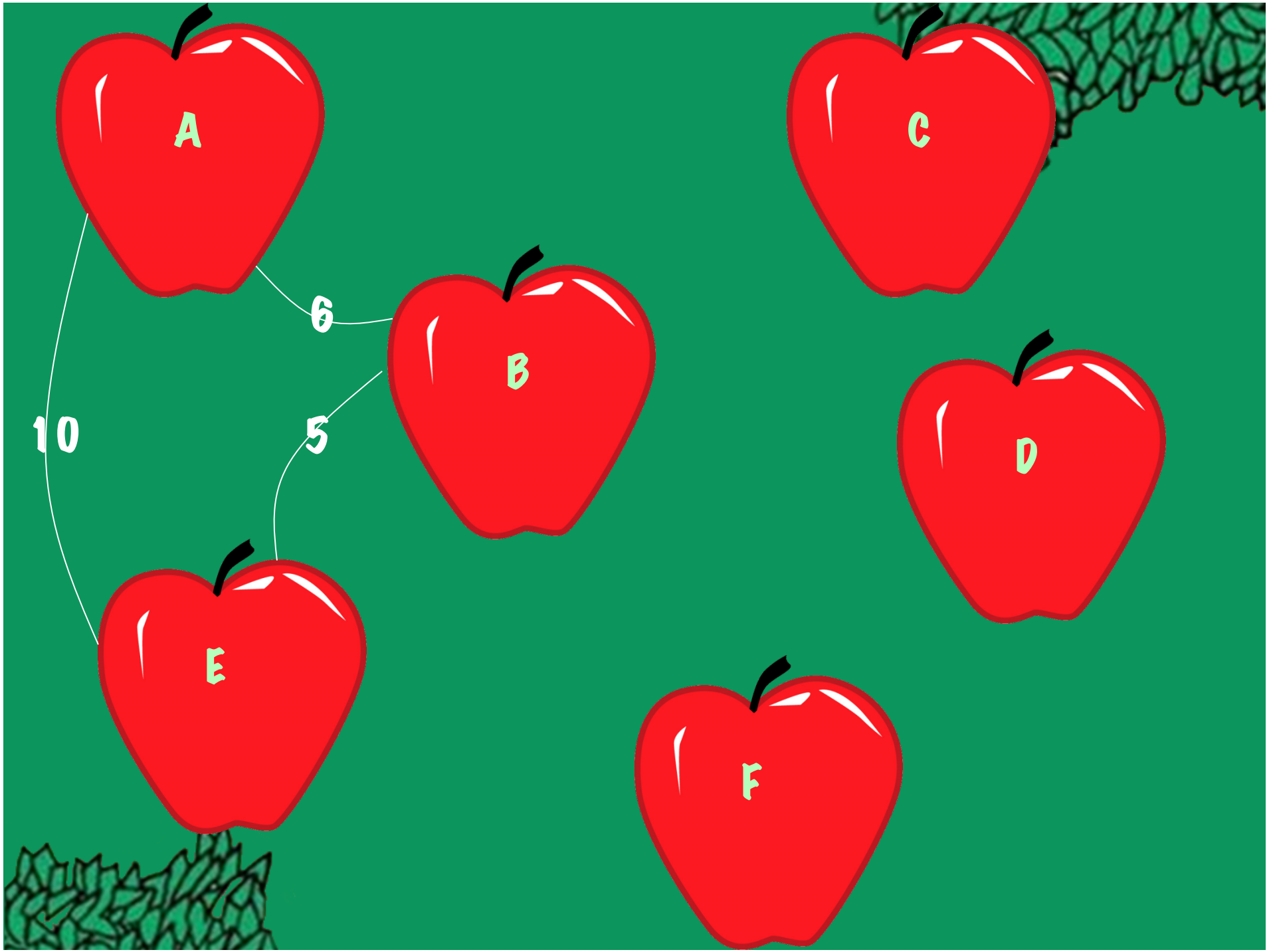

A
B, C, D
E, F

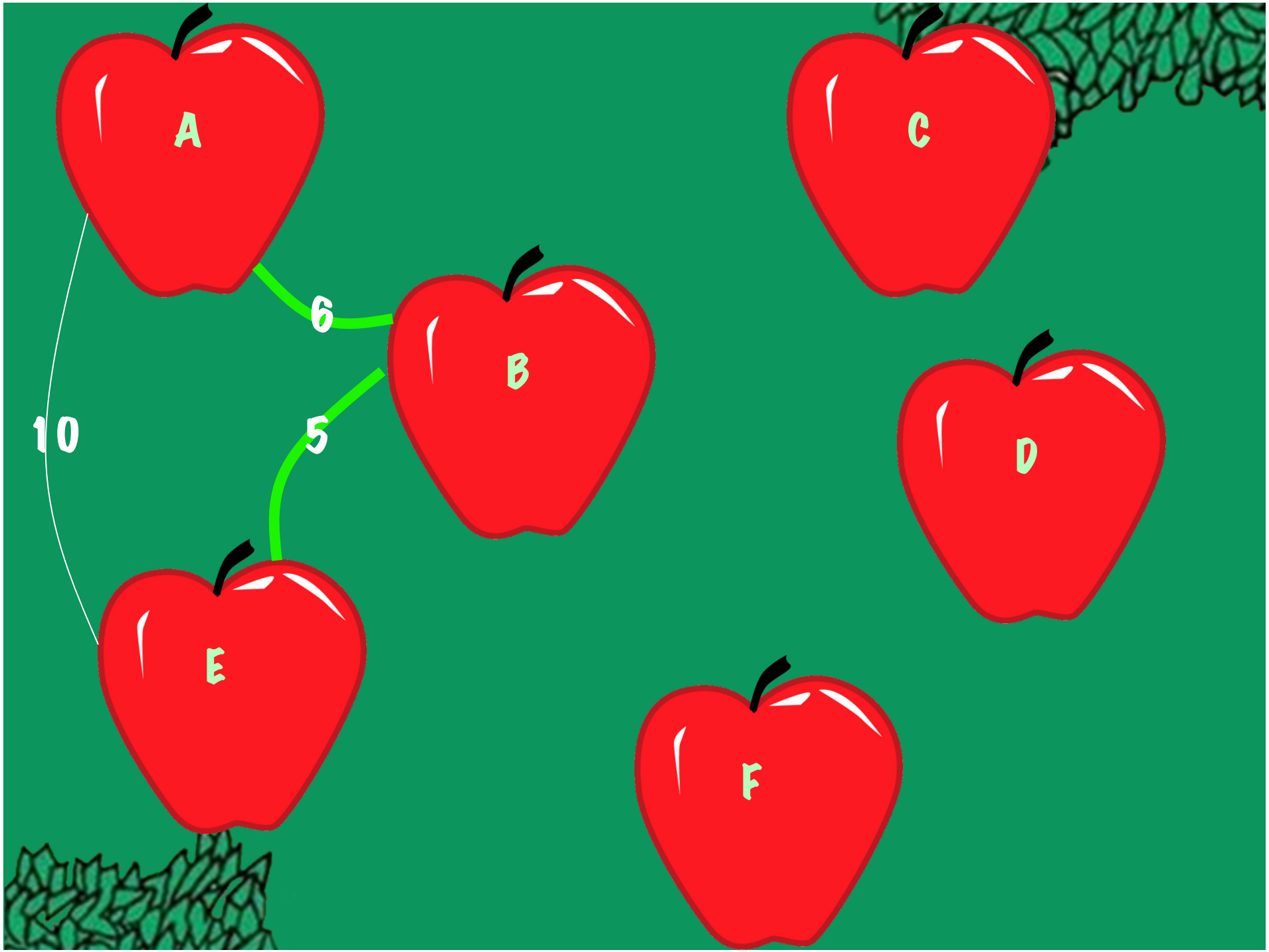

A
B,C,D,E,F

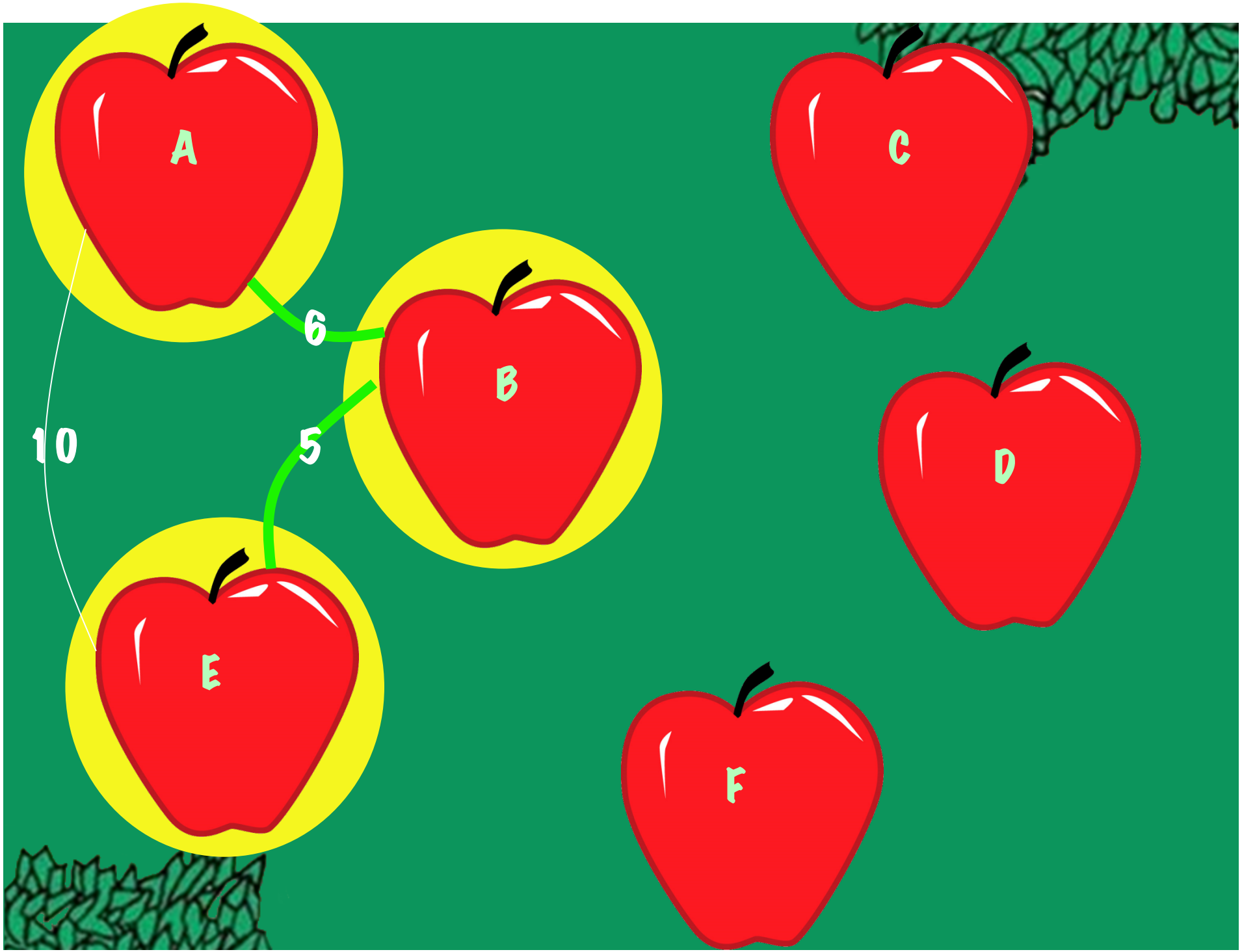

A
C
B
D
E
F
3
12
6
7
10
5
4
7

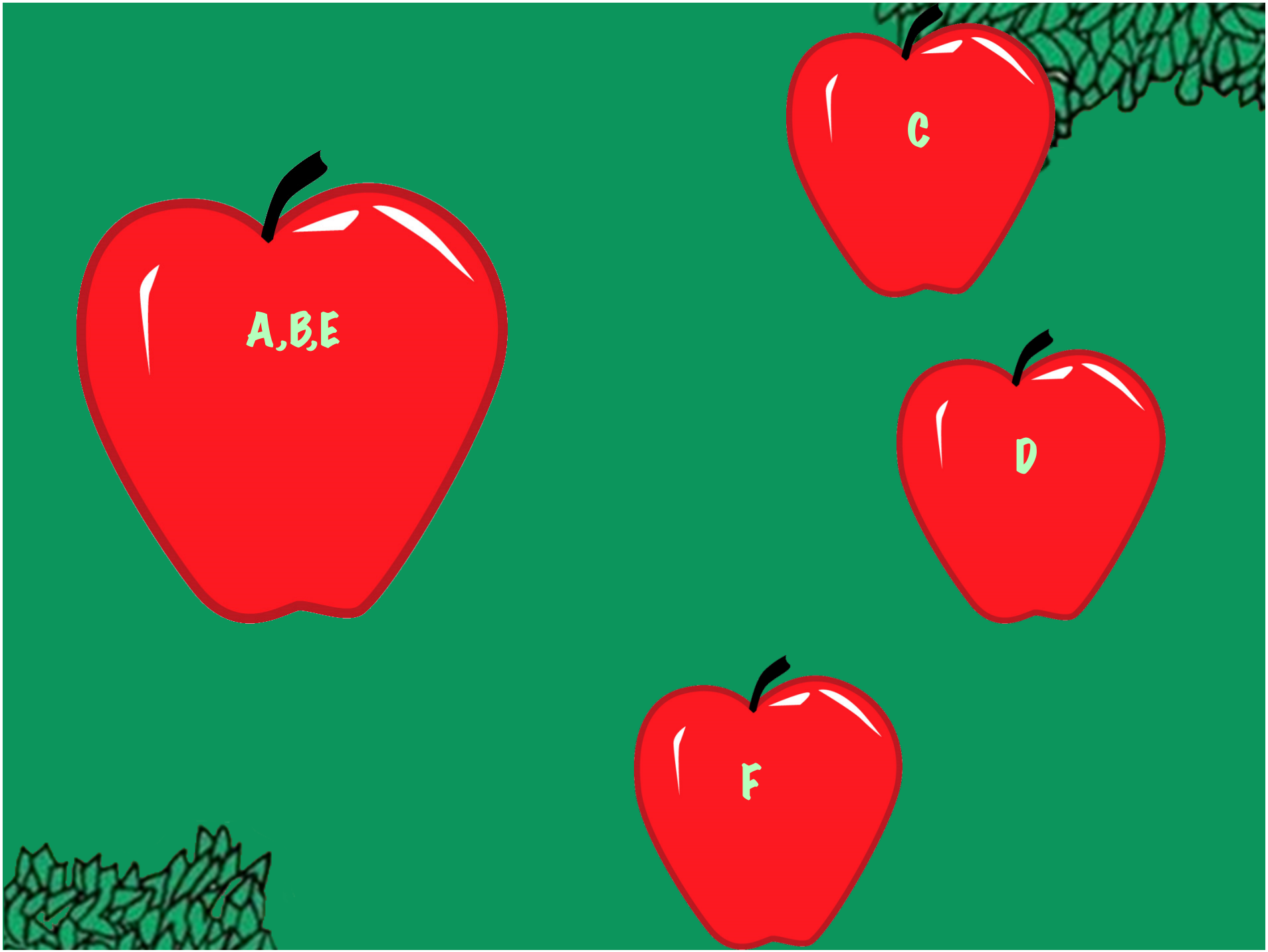

A
C
B
D
E
F
3
12
6
7
10
5
4
7

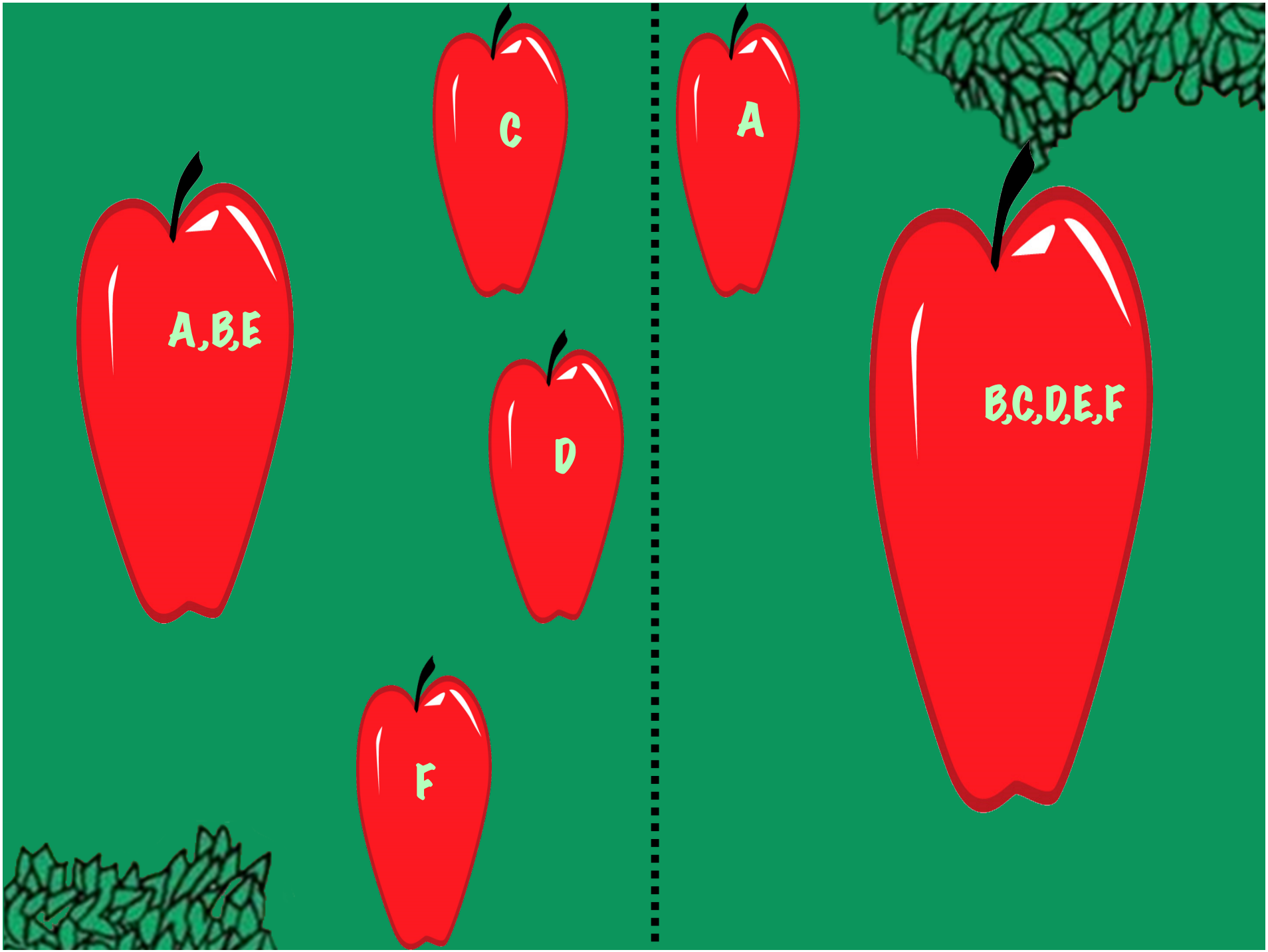

A
C
6
B
10
5
D
E
F

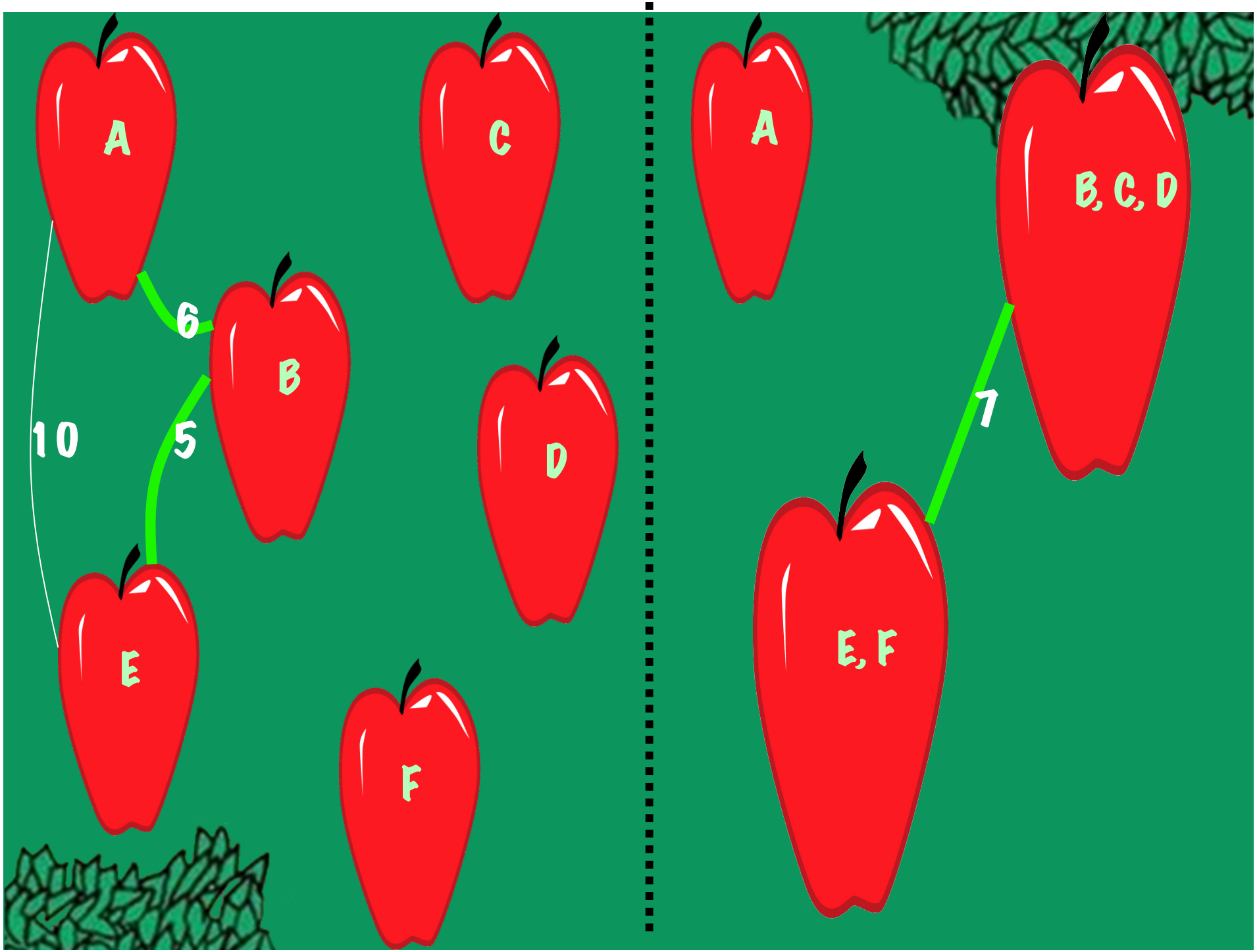

A
C
6
B
10
5
D
E
F

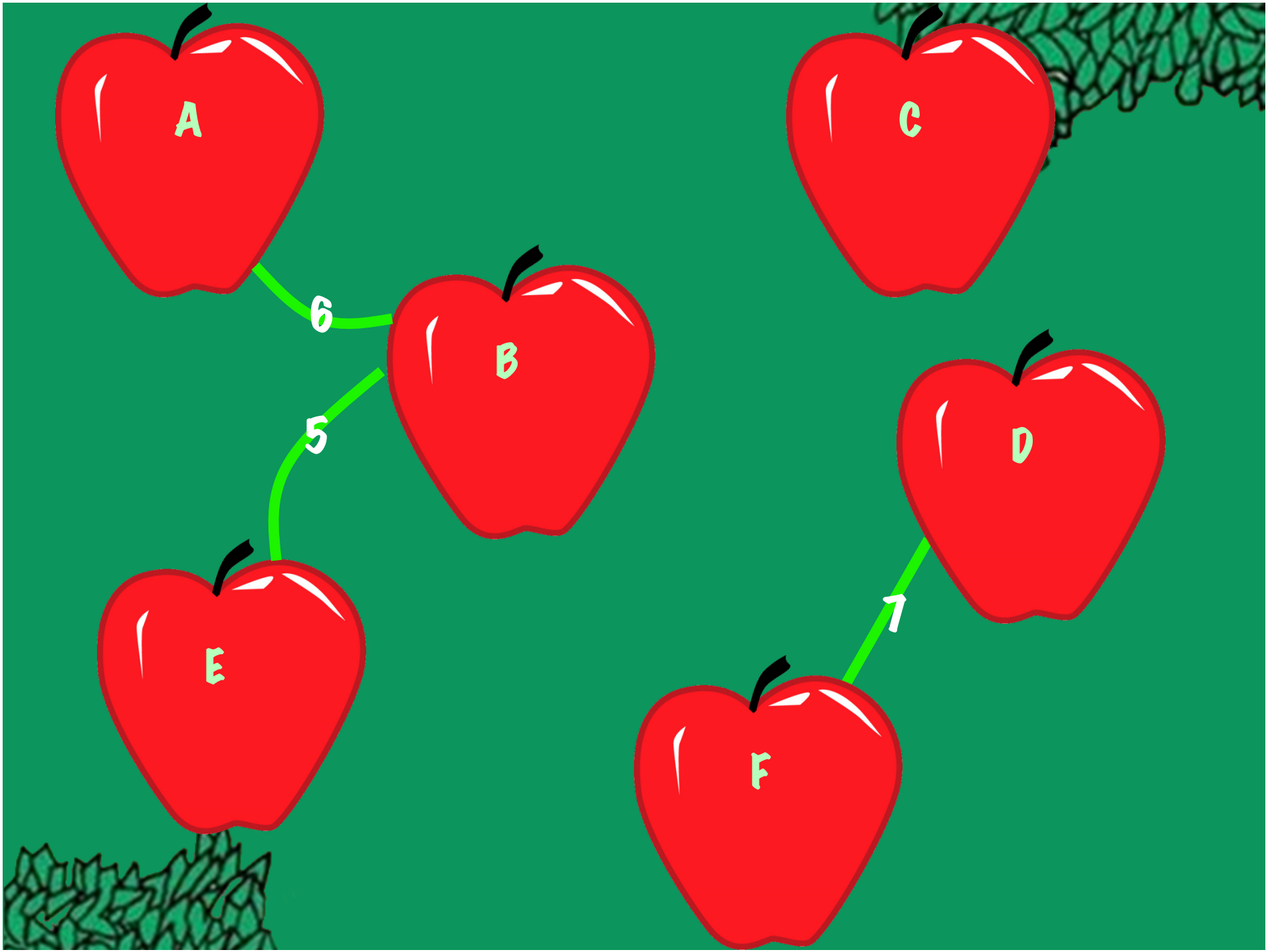

A
C
6
B
10
5
D
E
F

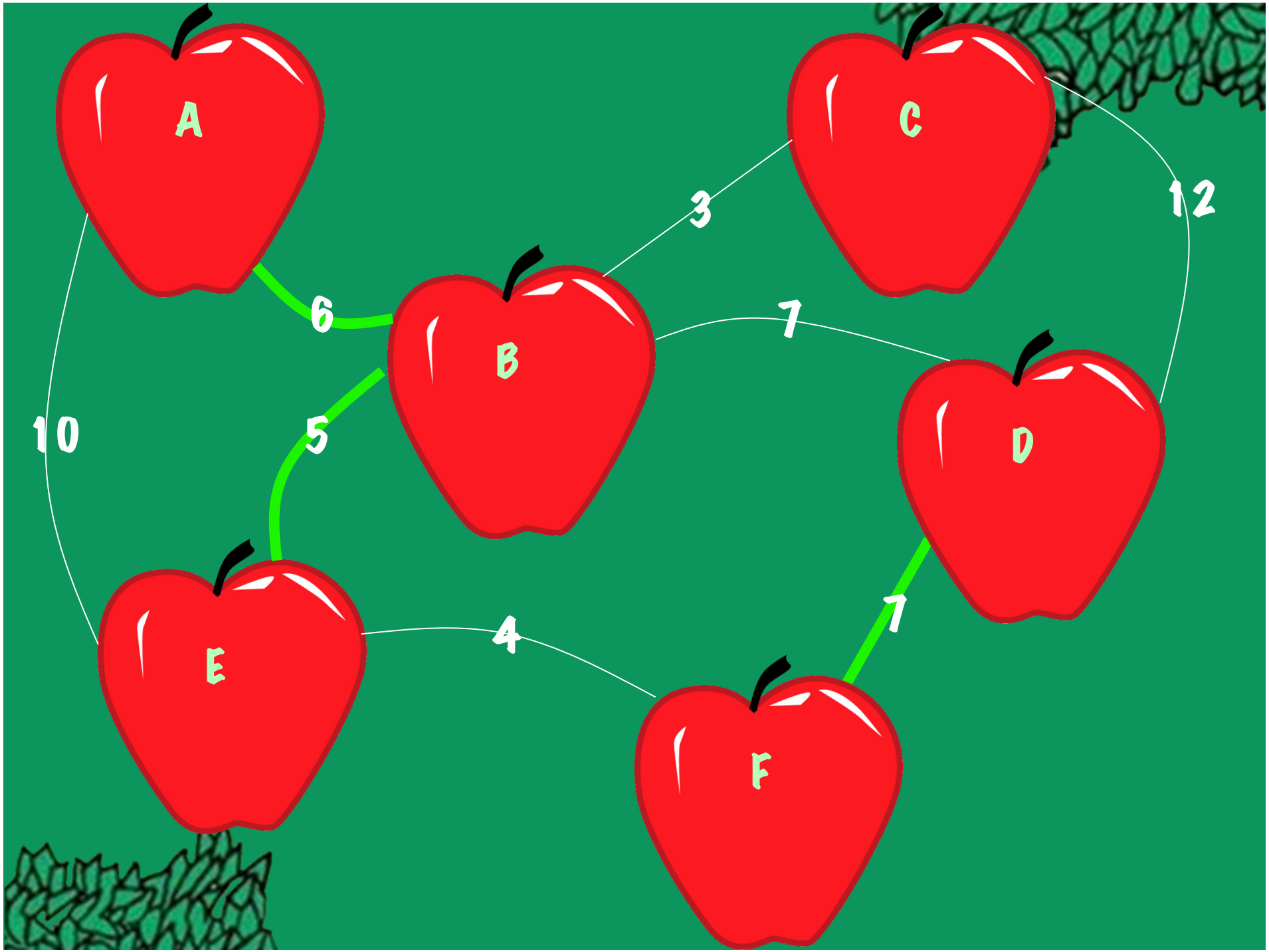

A
B
C
D
E
F
6
5
10

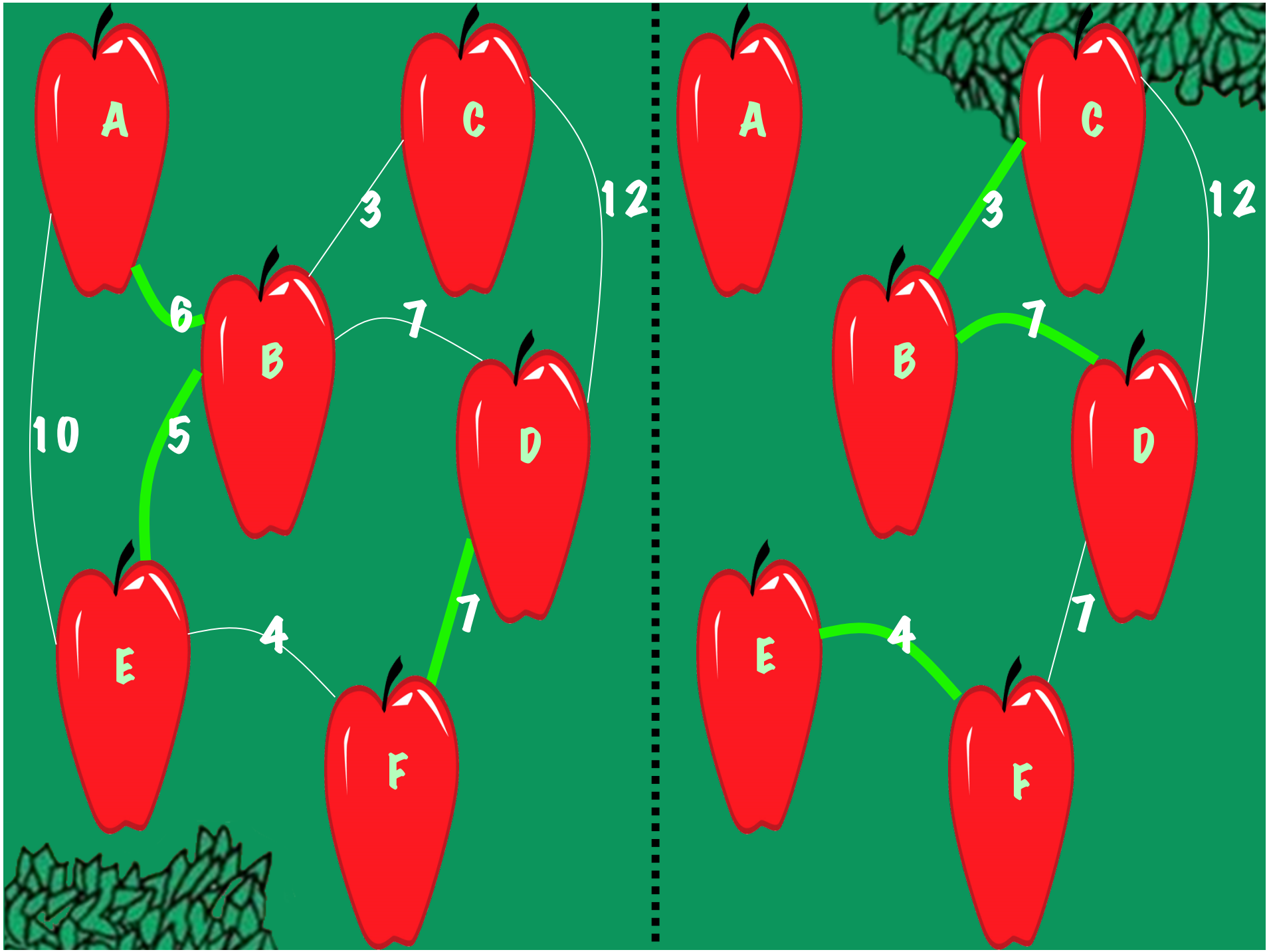

A,B,E
C
D
F

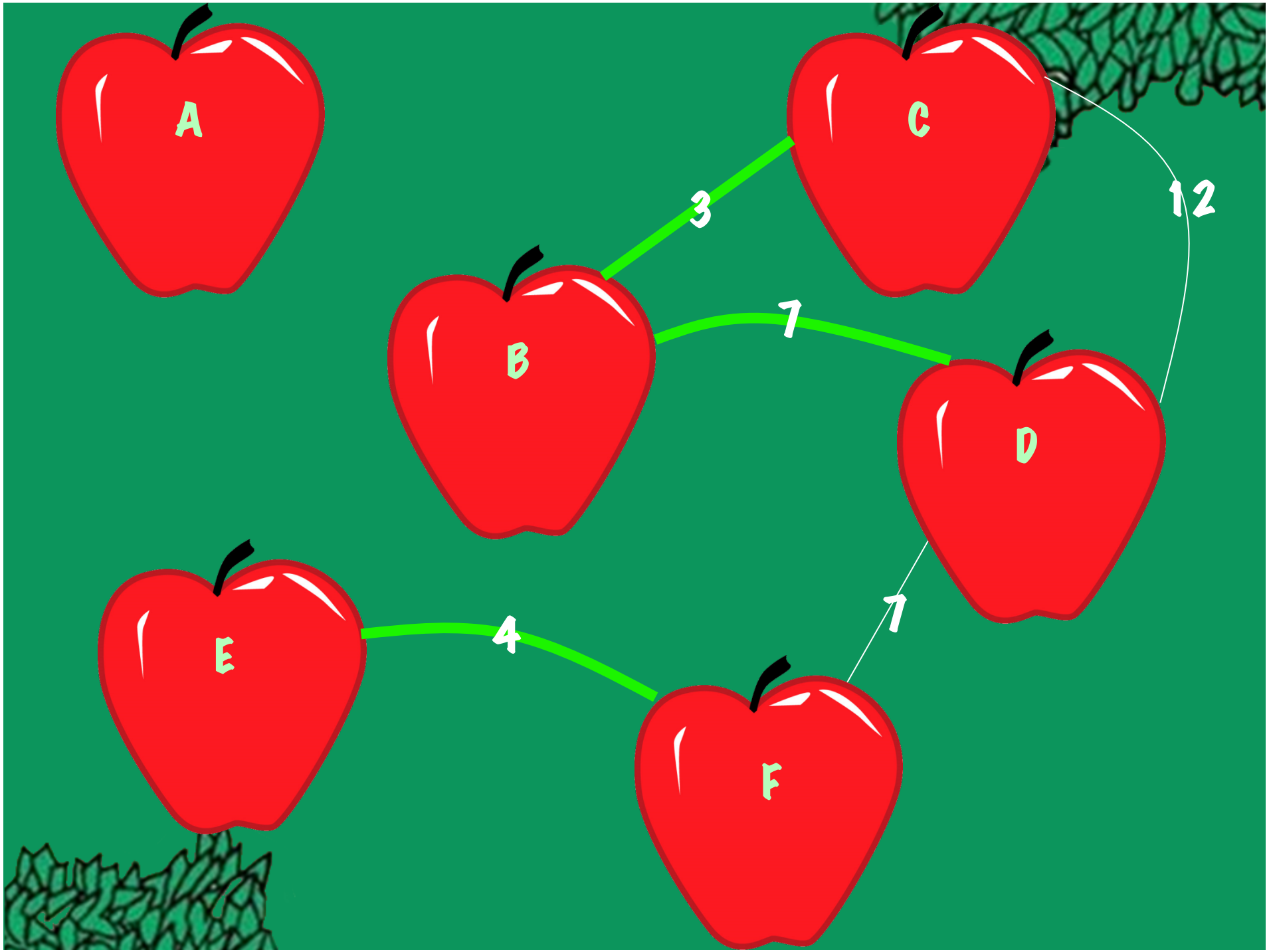

A,B,E
C
D
F
A
B,C,D,E,F

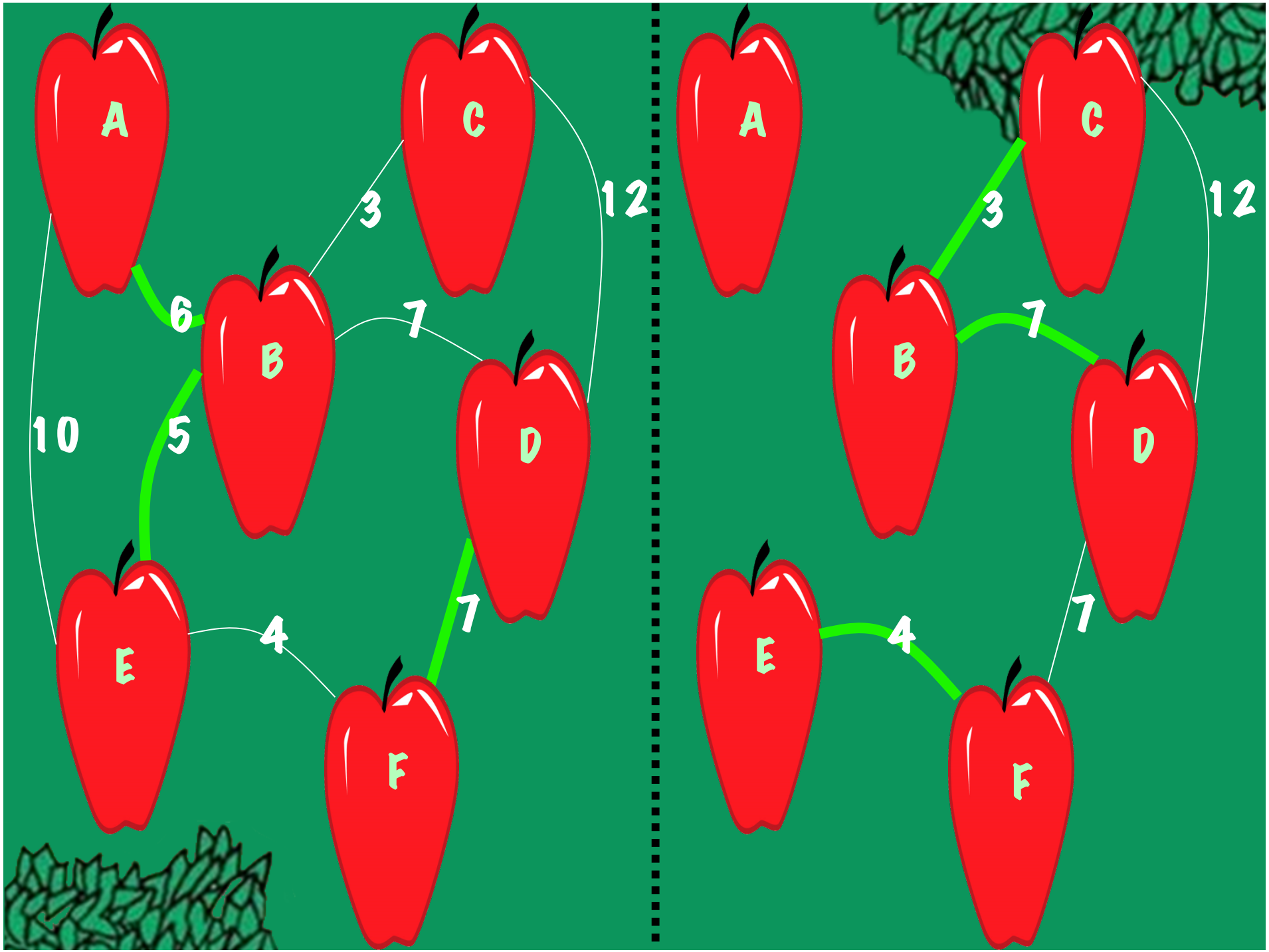

A
C
6
B
10
5
D
E
F
A
B, C, D
7
E, F

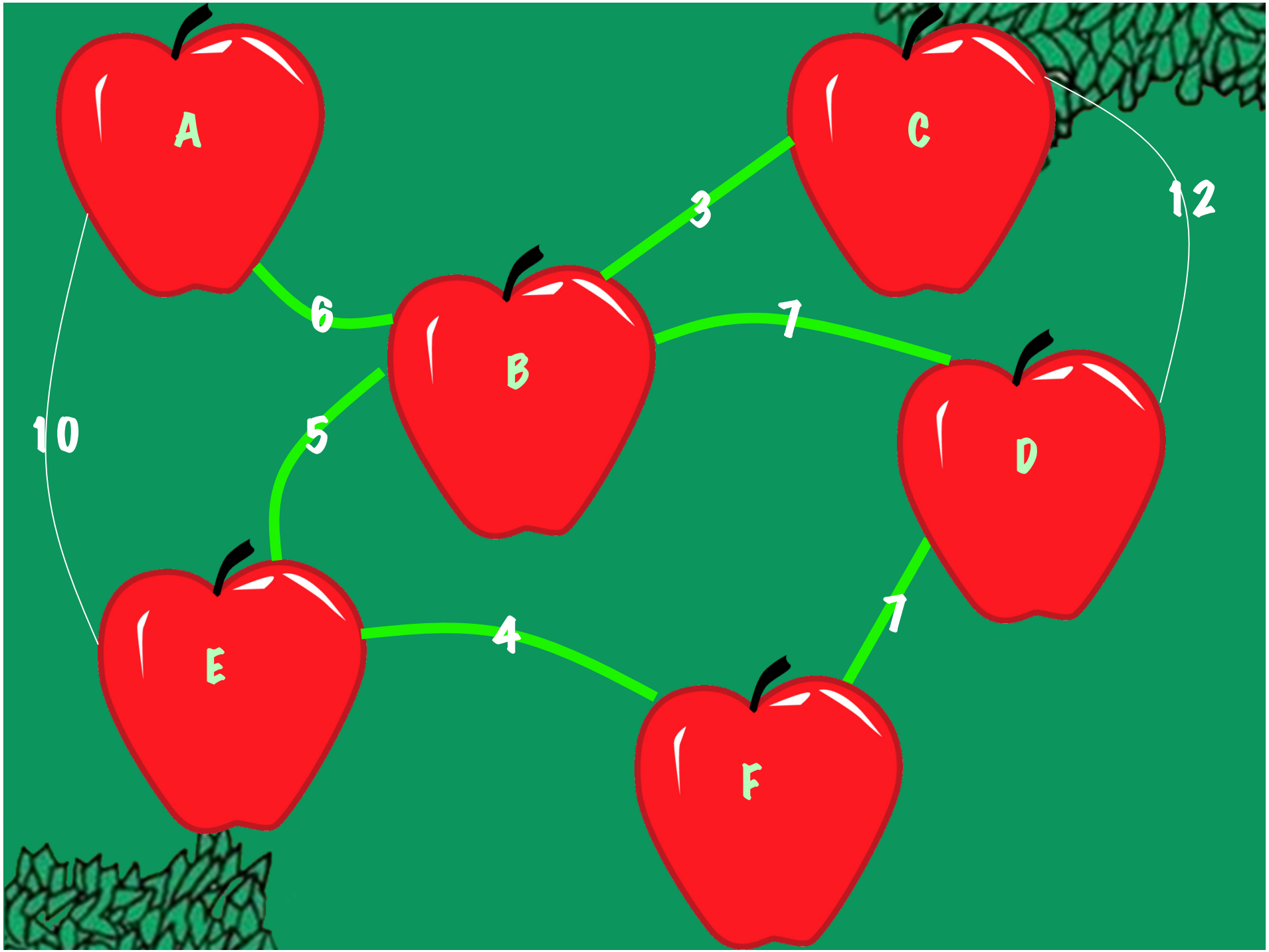

A
C
6
B
5
D
E
7
F

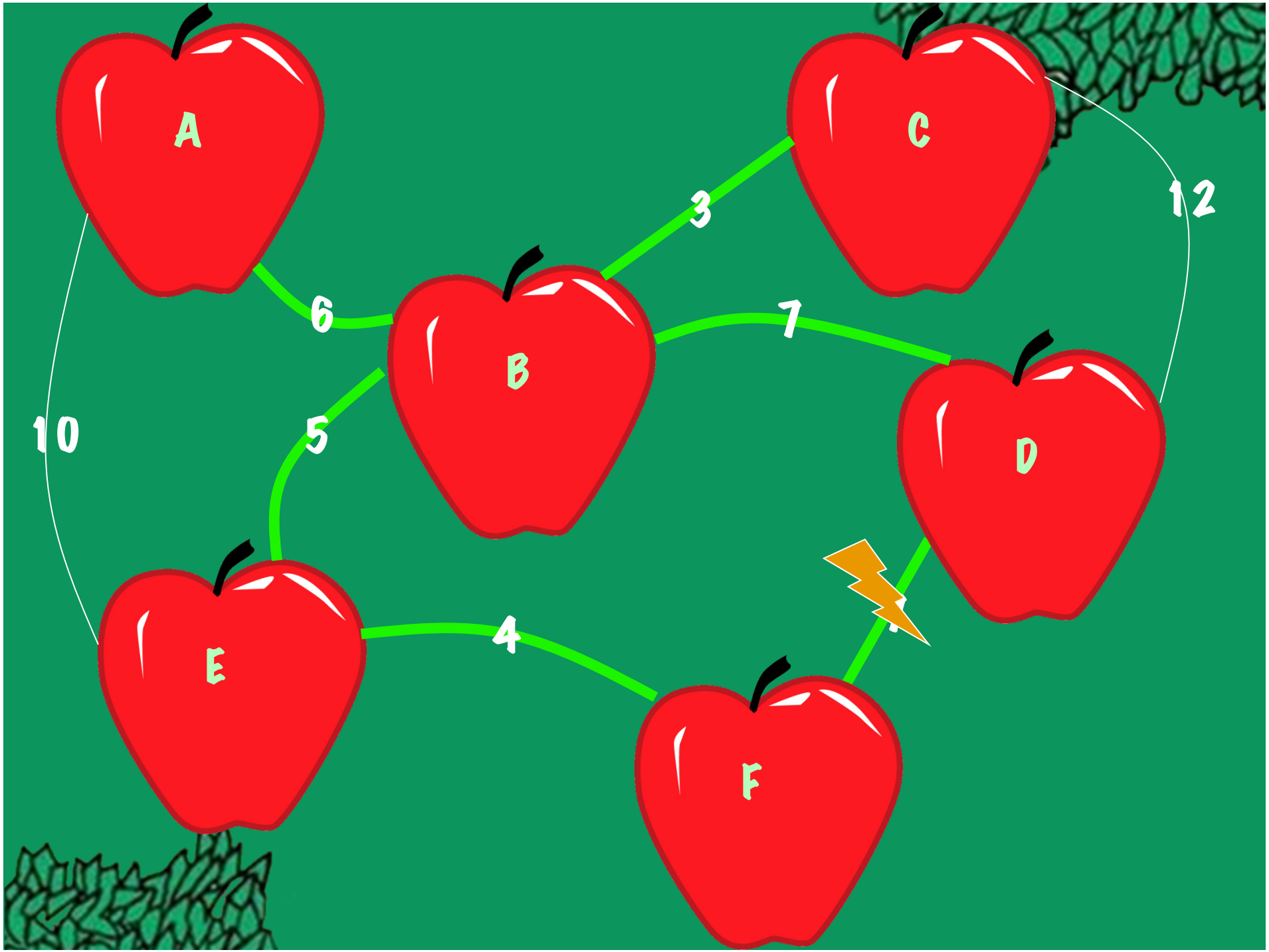

A
C
B
D
E
F
6
3
12
7
10
5
4
7

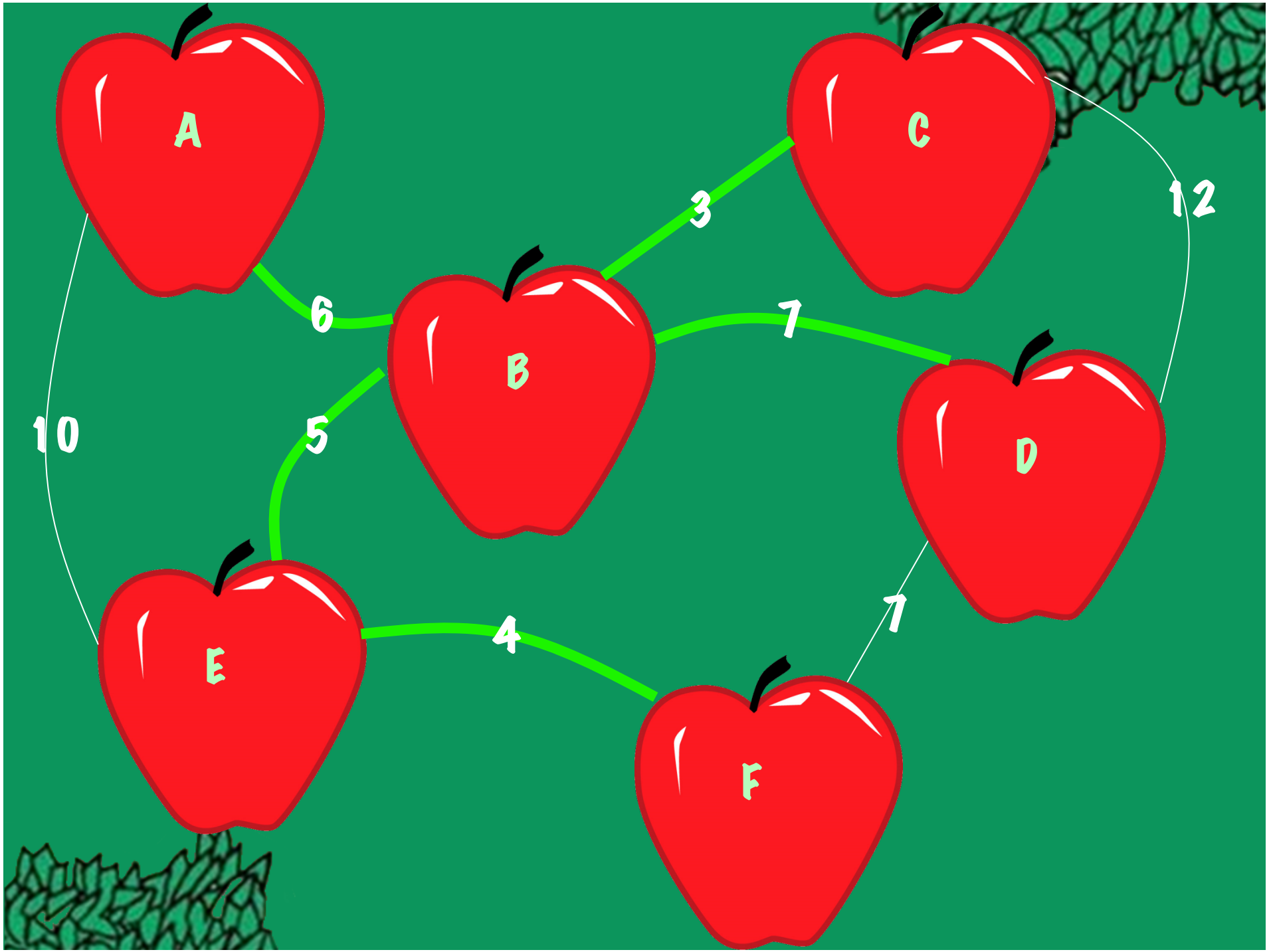

A
C
B
D
E
F
6
3
12
7
10
5
4
7
A
C
B
D
E
F
3
12
7
4
7

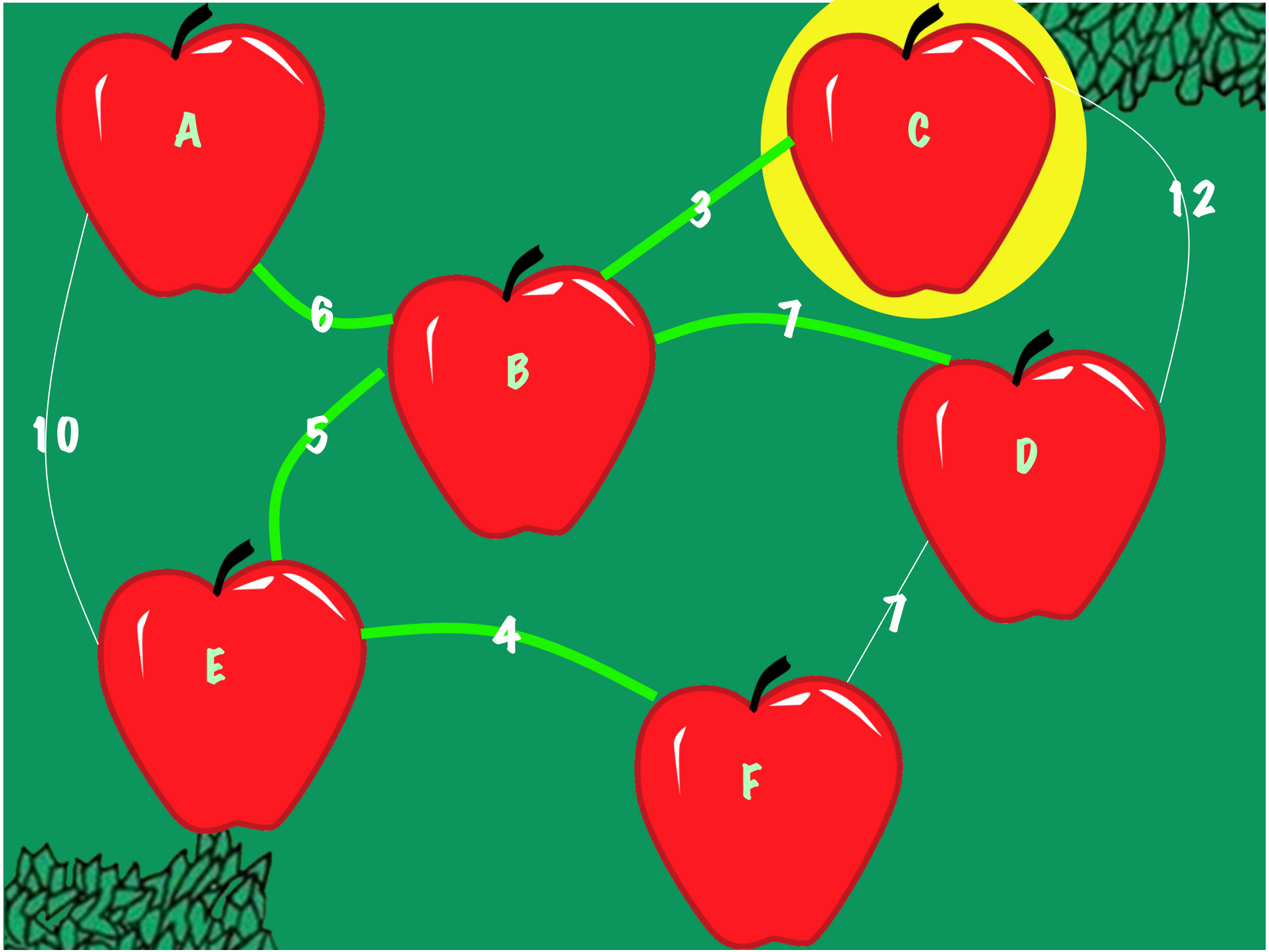

A
C
3
12
B
7
D
E
4
7
F

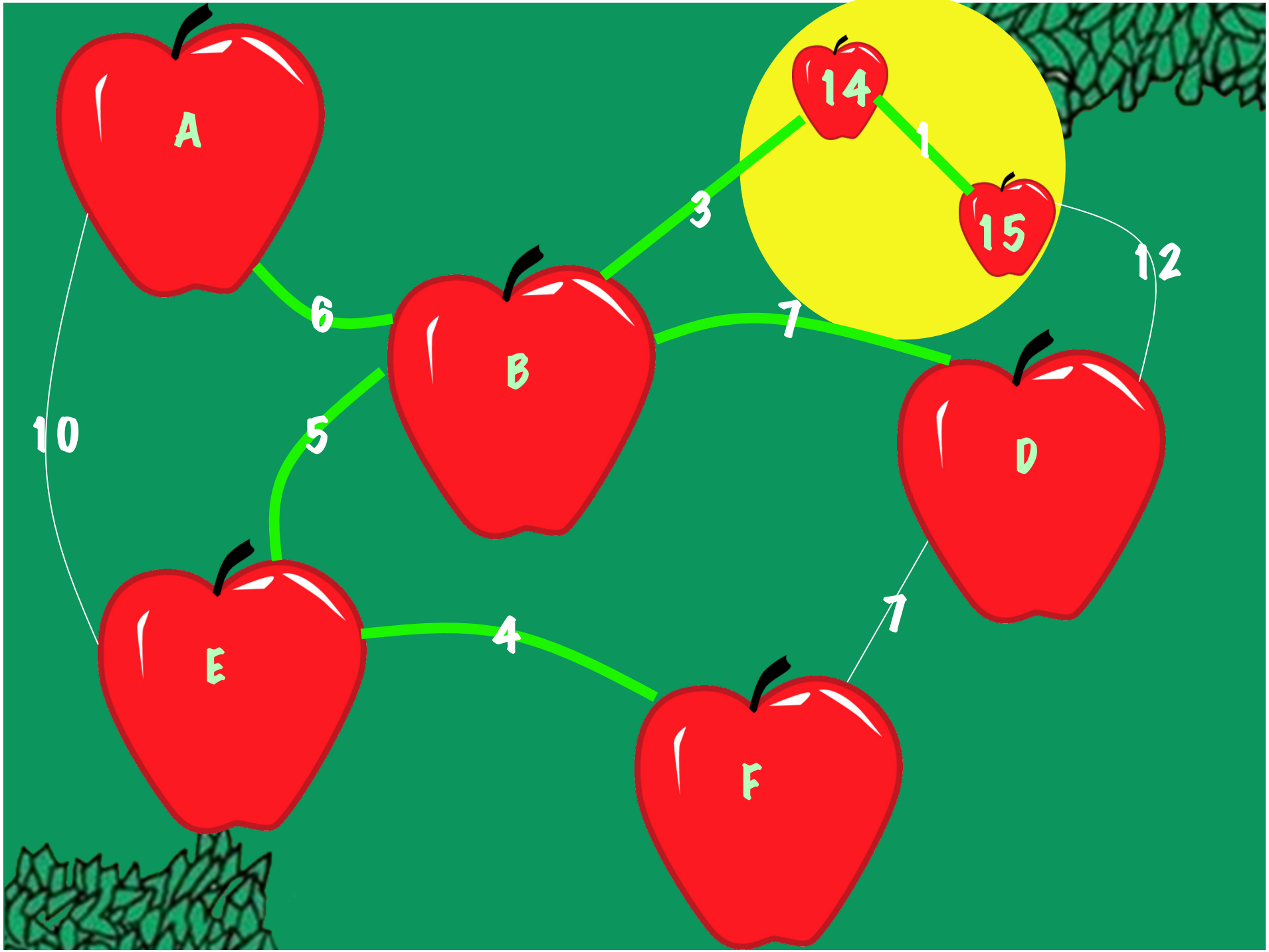

A
C
B
D
E
F
6
3
12
7
10
5
4
7
A
C
B
D
E
F
3
12
7
4
7

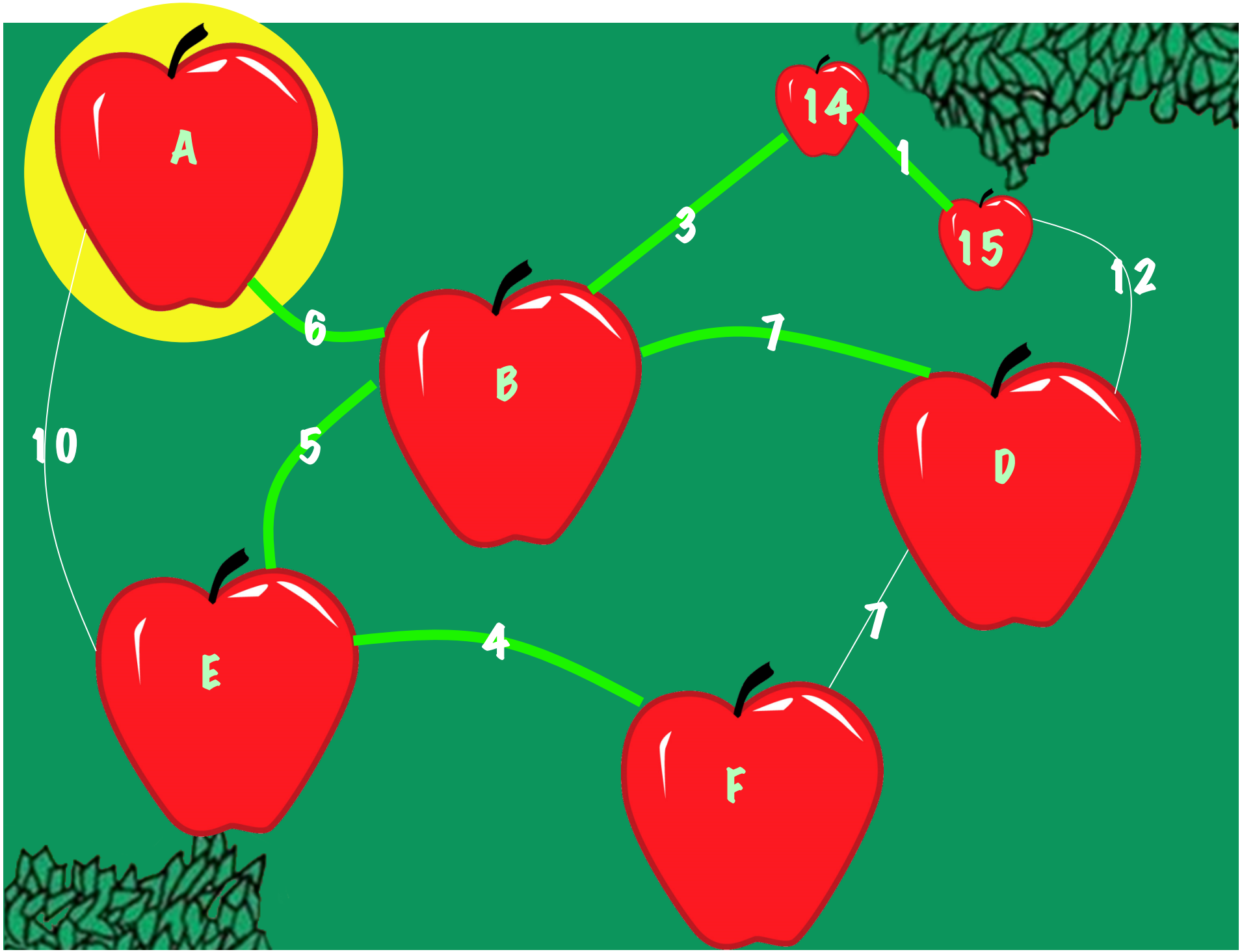

A
C
B
D
E
F
6
3
12
10
5
7
4
7

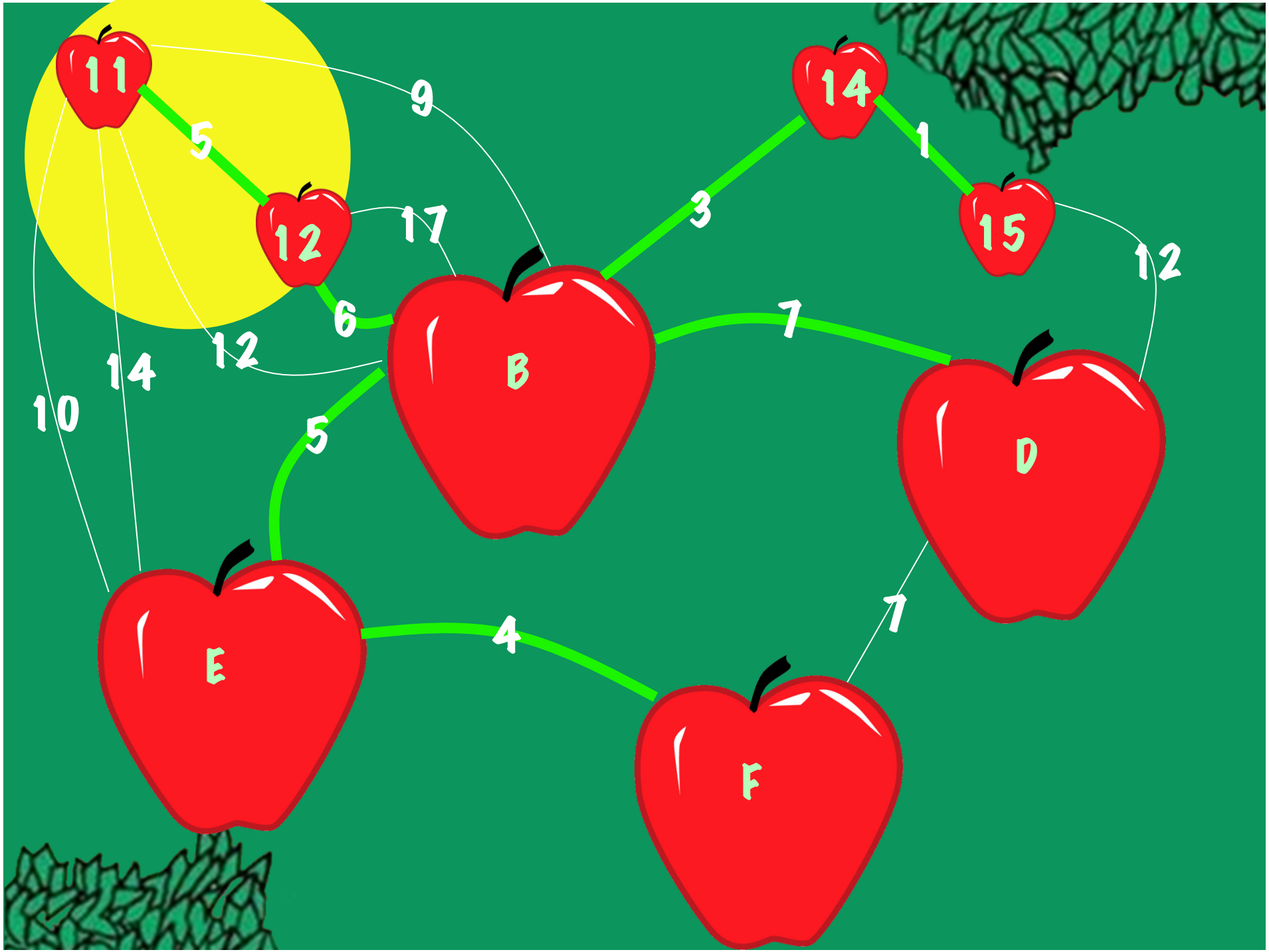

A
C
B
D
E
F
3
12
6
7
10
5
4

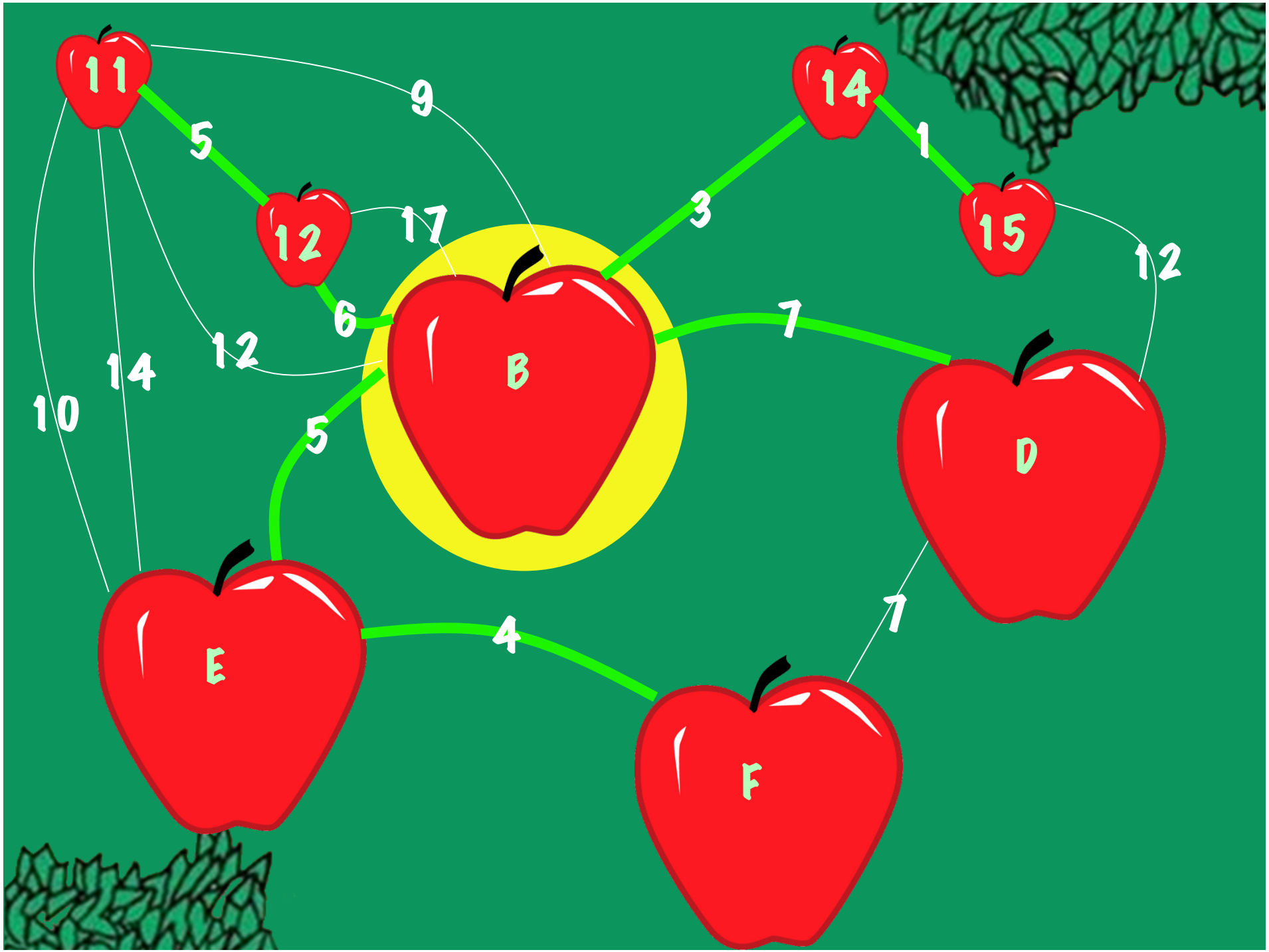

A
C
B
D
E
F
6
3
12
10
5
7
4
7

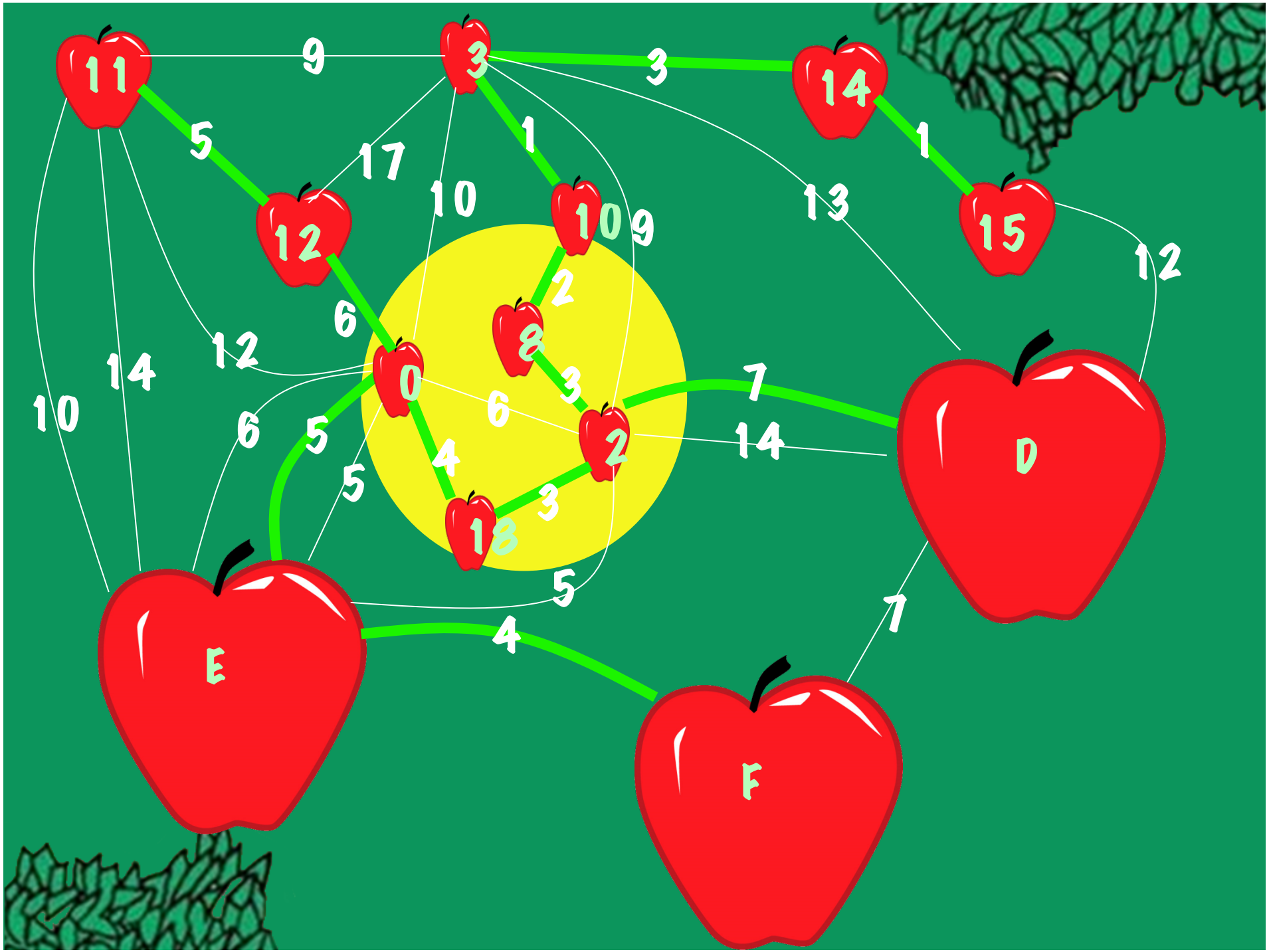

A
C
B
D
E
F
3
12
6
7
10
5
4
7

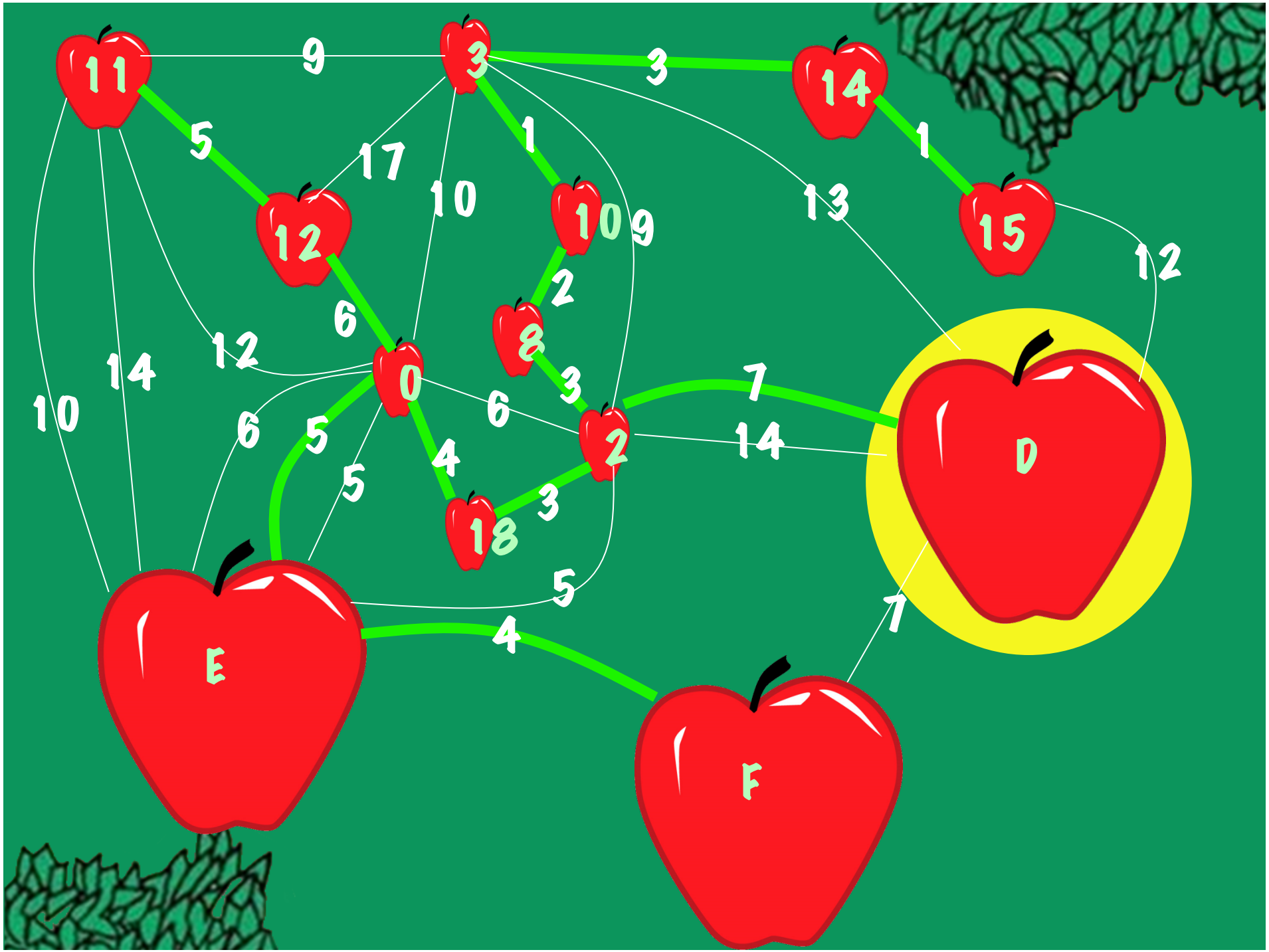

A
B
D
E
F
14
15
6
3
1
12
10
5
7
4
7

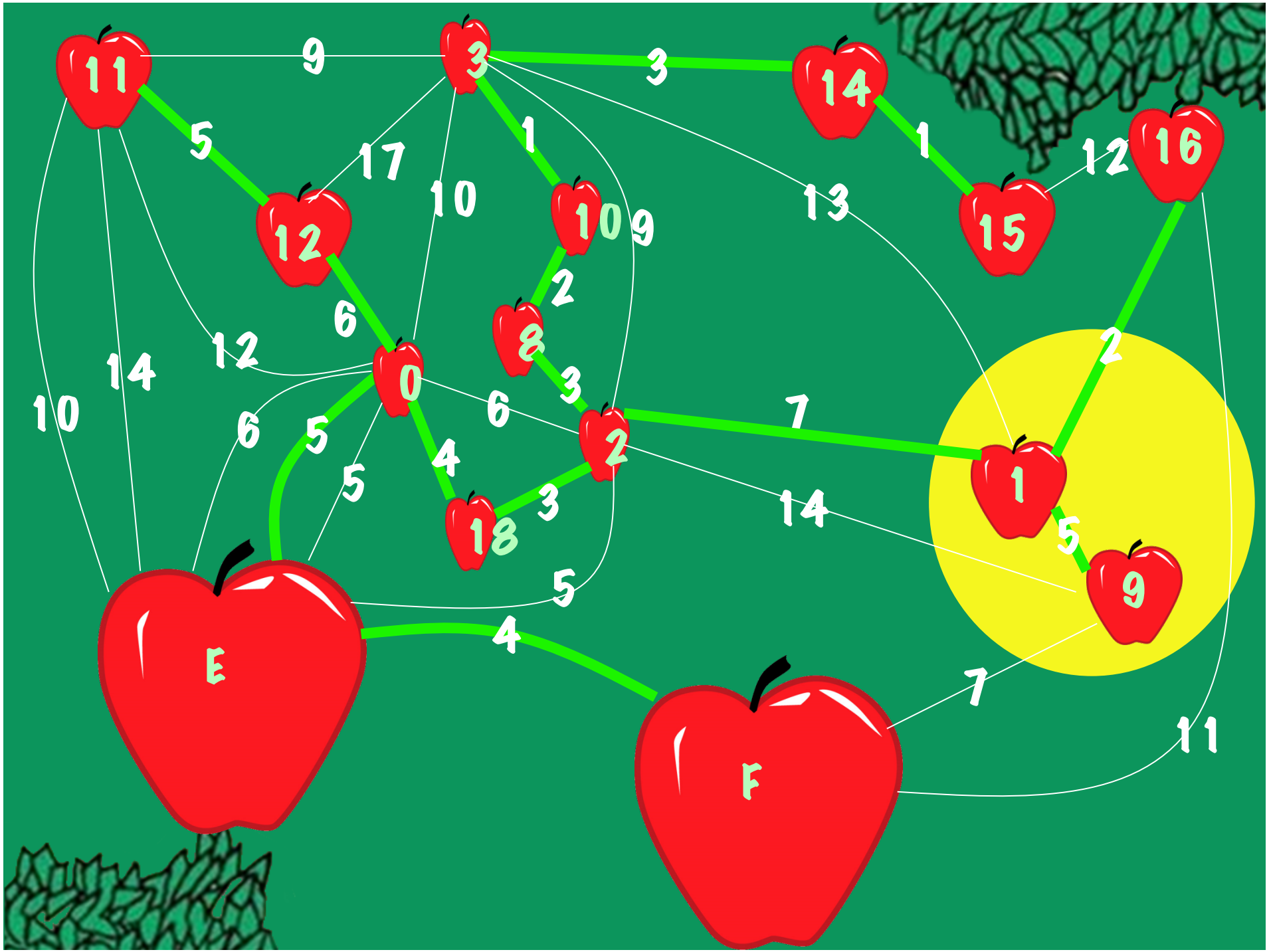

A
14
15
B
D
E
F
6
3
1
12
10
5
7
4
7

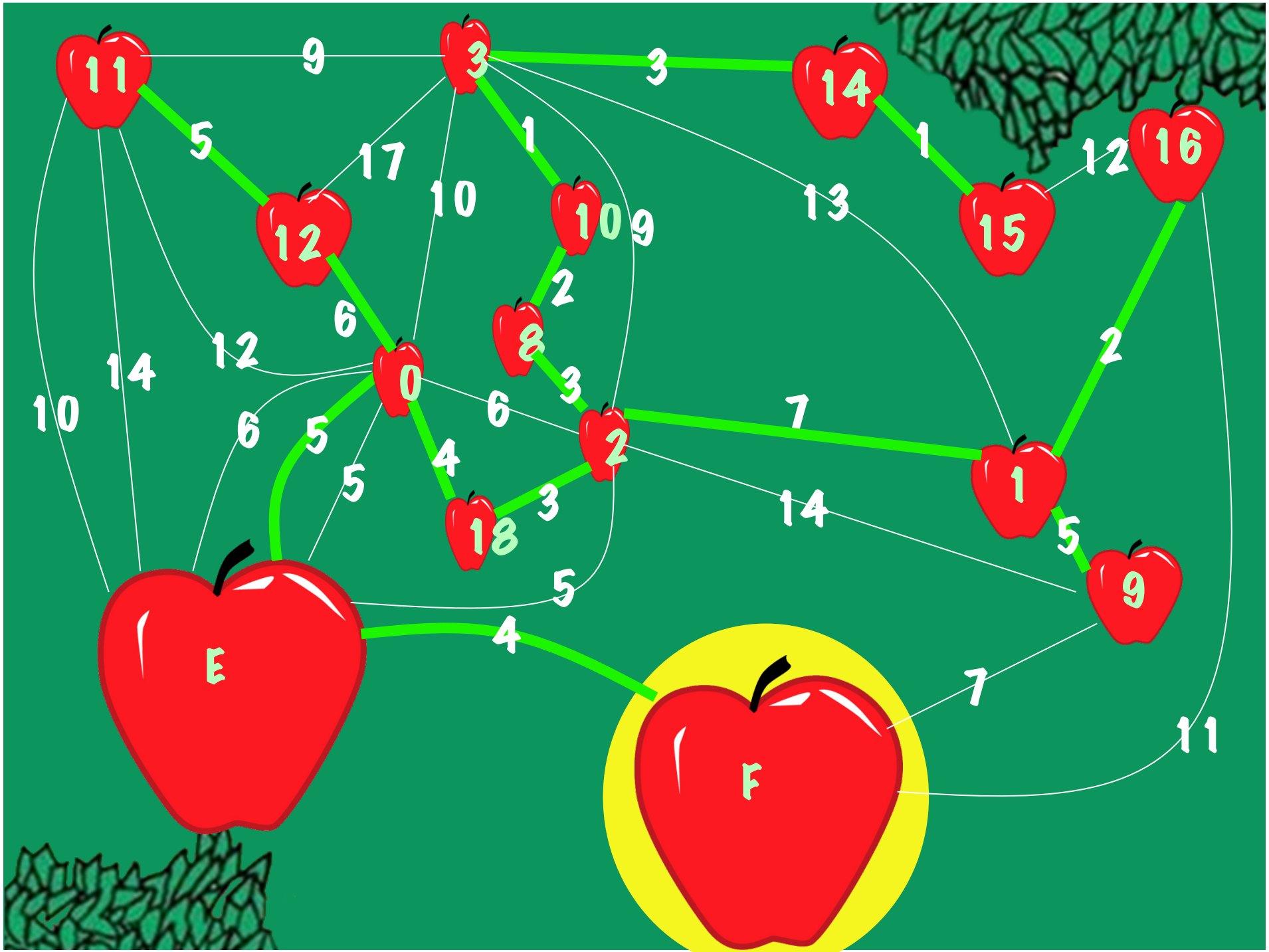

11
12
14
15
B
D
E
F
5
9
17
3
1
12
6
7
14
12
10
5
4
7

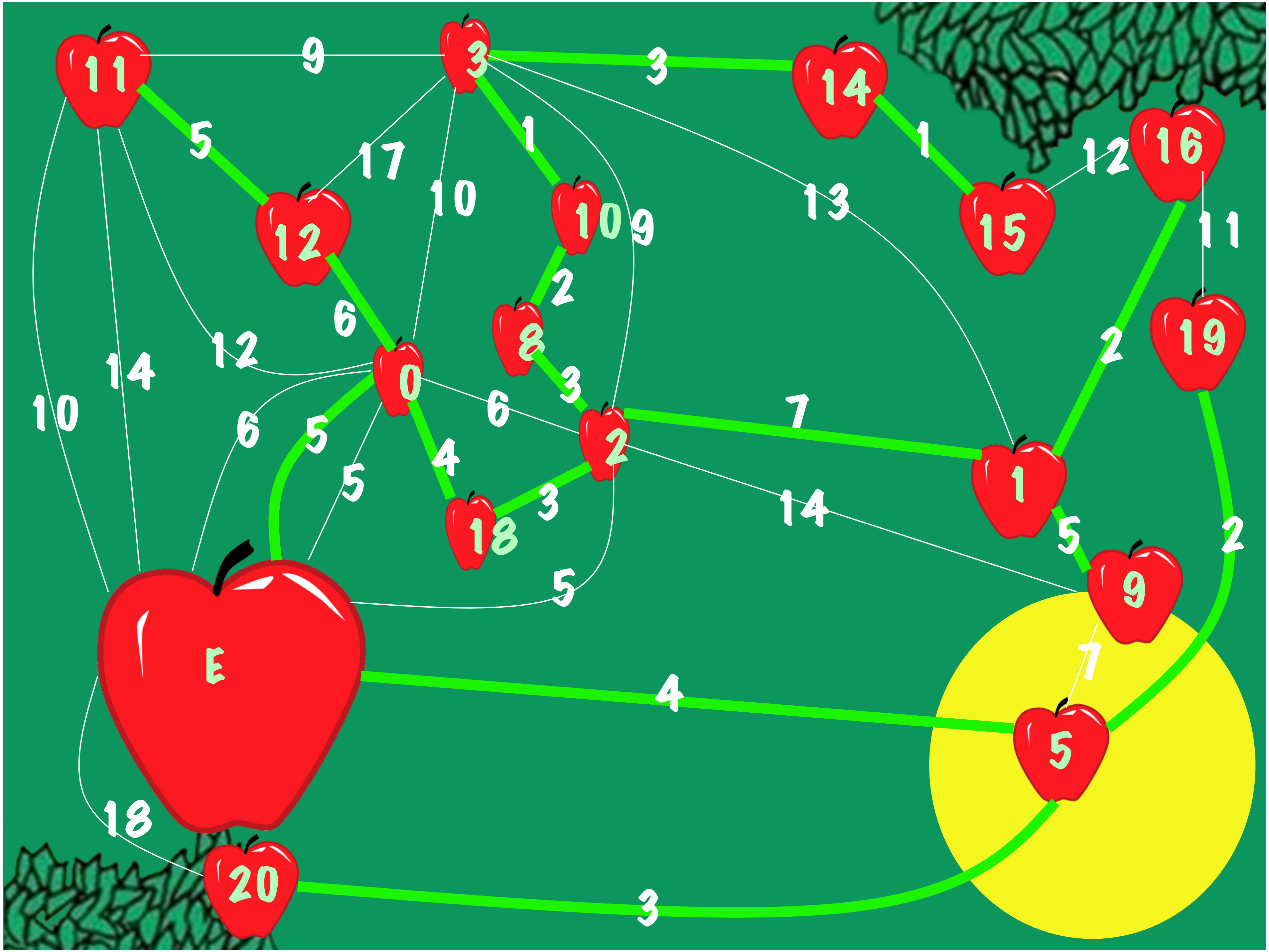

11
14
12
15
B
D
E
F
9
5
17
3
1
12
6
7
12
14
10
5
4
7


11
9
3
3
14
5
17
1
1
10
10
9
13
15
12
12
6
2
8
12
14
10
6
5
0
6
3
7
14
5
4
2
D
5
3
18
5
E
4
7
F

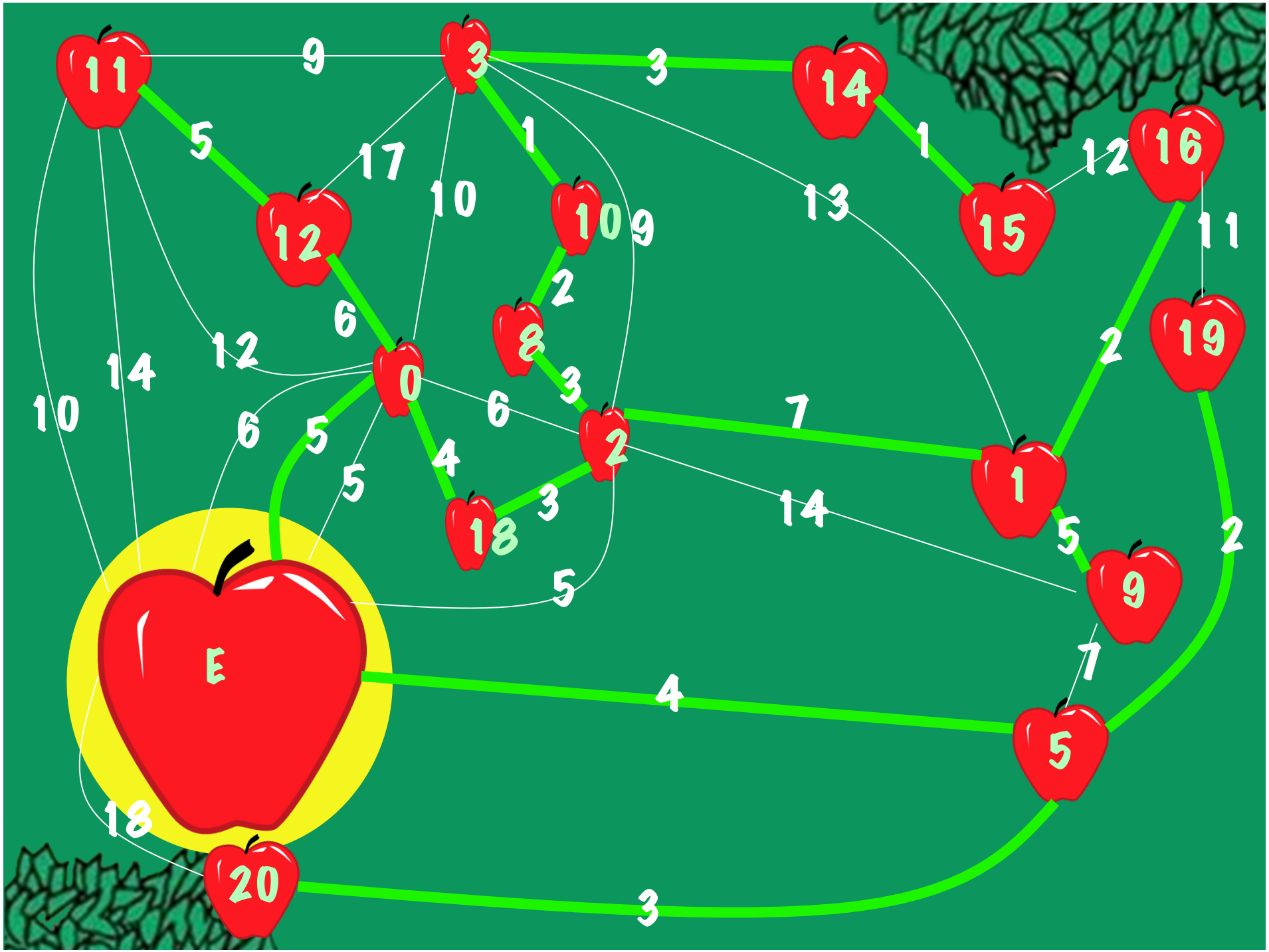

11
9
3
3
14
5
17
1
1
10
10
9
13
15
12
12
6
2
8
14
12
0
3
7
10
6
6
5
2
14
D
4
5
5
3
18
5
E
4
F
7

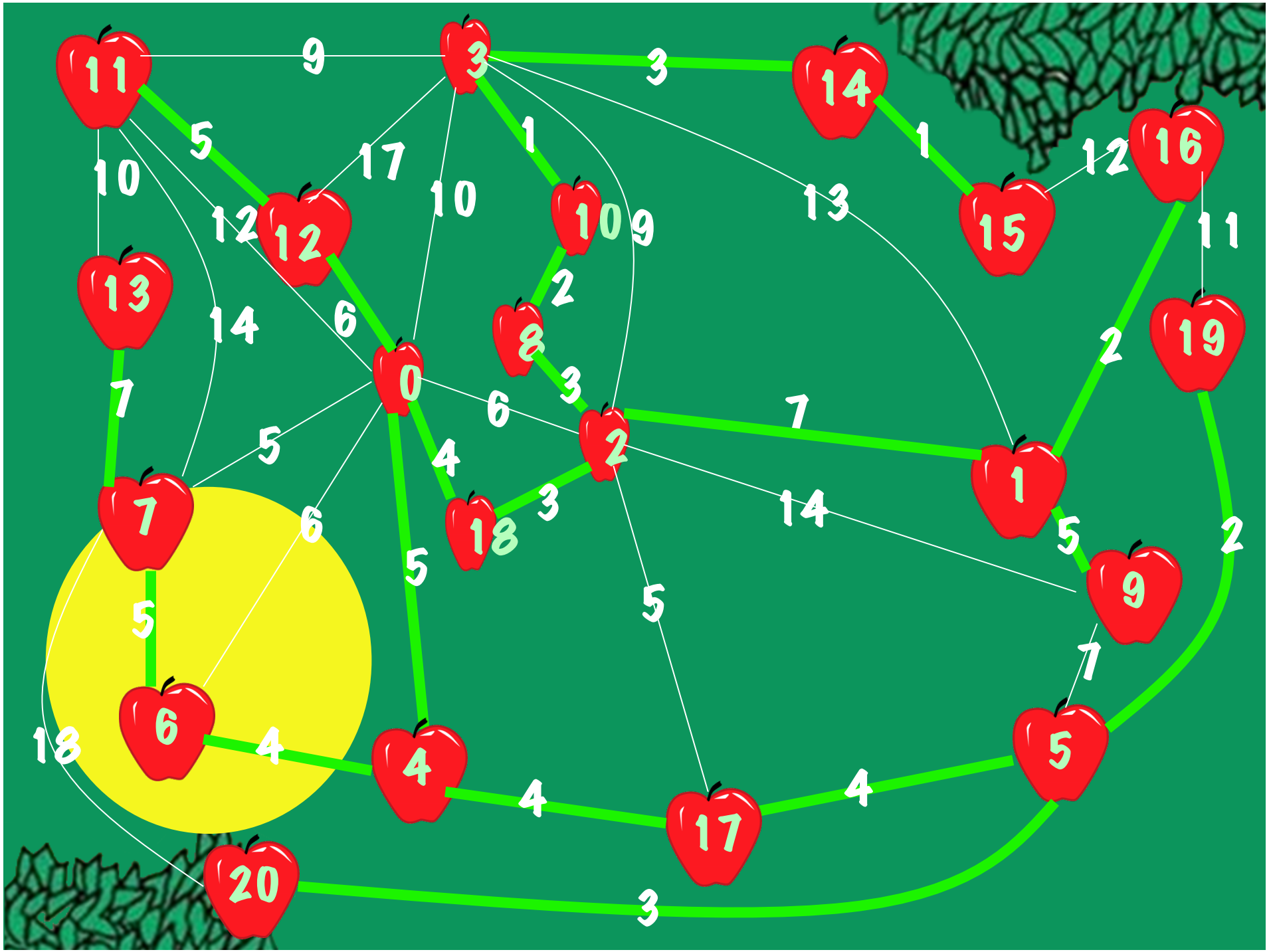

11
9
3
3
14
5
17
1
1
12
16
12
10
10
9
13
15
2
6
8
12
14
2
10
0
6
3
7
6
5
6
4
2
1
5
5
3
14
18
5
9
5
E
4
7
F
11

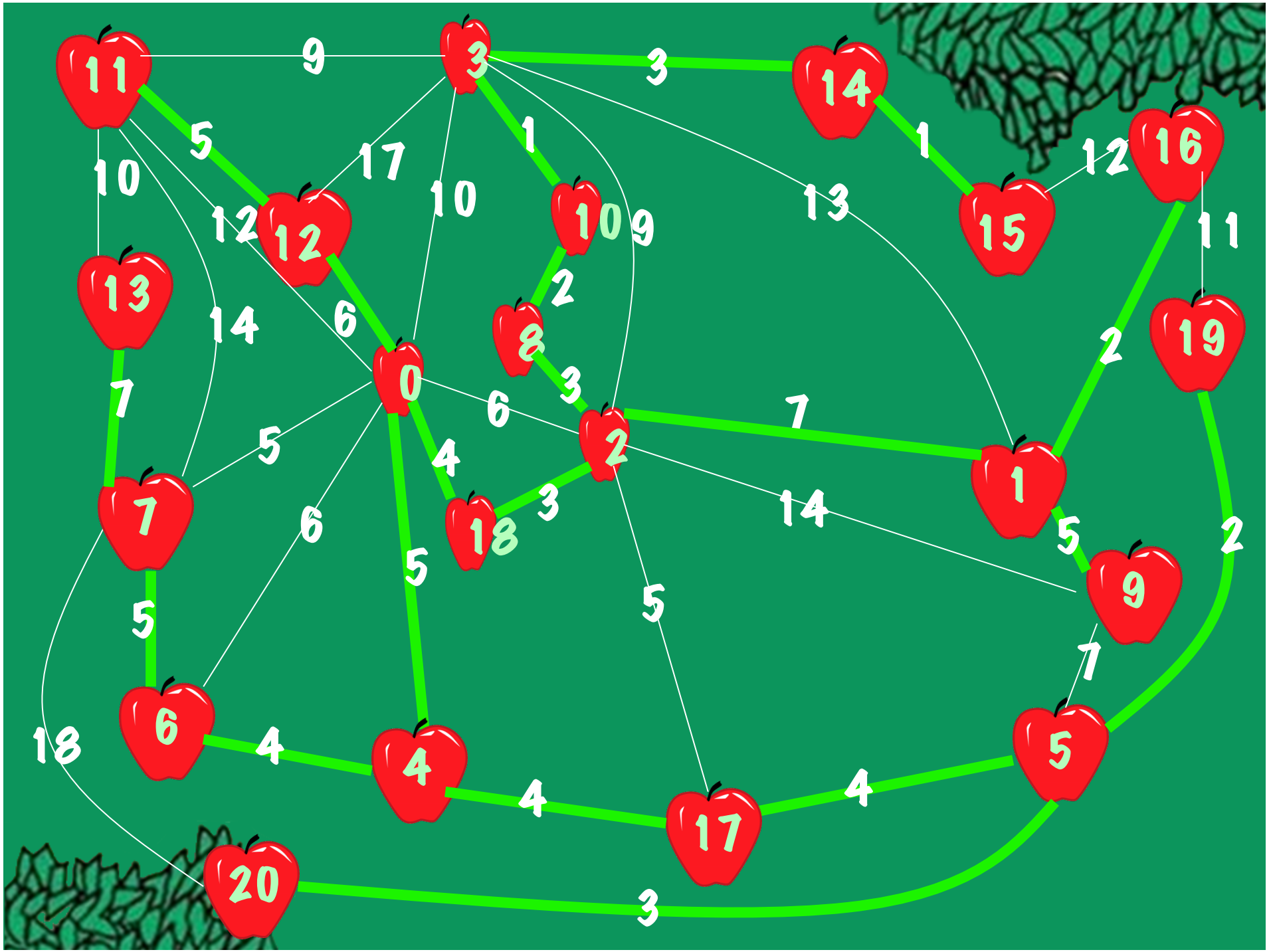

11
9
3
3
14
5
17
1
12
10
10
9
13
15
1
12
16
6
2
12
14
10
8
3
0
6
2
7
1
2
6
5
5
4
6
5
3
18
14
5
9
5
E
4
F
7
11

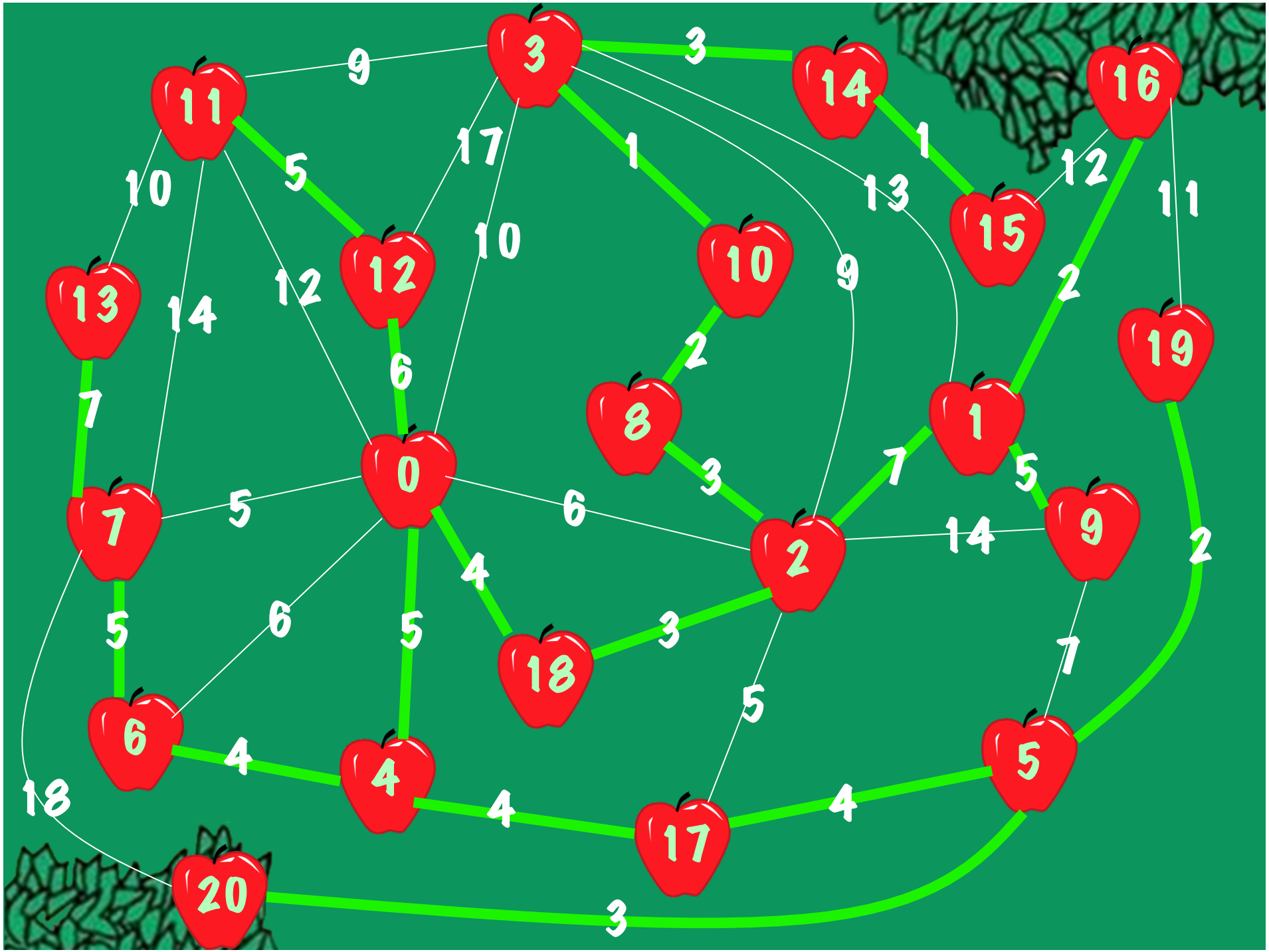


11
9
3
3
14
5
1
1
17
10
12
16
12
10
9
15
13
11
2
8
6
12
19
14
0
2
10
6
7
3
5
2
1
6
5
4
3
14
18
5
2
9
5
E
7
4
5
18
20
3


11
9
3
3
14
5
17
1
12
10
16
10
9
13
15
12
11
2
6
19
8
12
14
0
3
2
10
6
6
5
7
1
4
5
18
3
14
5
9
2
E
5
4
7
5
18
20
3


11
9
3
3
14
5
17
1
16
10
12
10
13
12
10
9
15
12
11
13
14
6
2
8
19
0
3
2
7
7
5
6
4
1
18
3
14
5
2
9
5
5
7
6
18
4
4
4
17
4
5
20
3


11
3
14
16
12
15
13
10
8
0
2
1
19
7
18
9
6
4
17
5
20
9
3
5
12
17
10
1
10
9
13
1
12
11
13
14
6
2
3
7
2
7
5
4
6
3
14
5
2
5
5
7
18
4
4
4
3


3
11
14
16
15
12
10
13
12
10
19
8
1
0
7
9
2
18
6
4
17
5
20
9
3
14
1
10
17
1
13
12
11
10
14
12
6
2
7
5
6
7
5
6
4
5
3
5
7
4
4
4
18
3

Why Does It Terminate?

- The Boruvka step reduces the size of the graph by a factor of 2
- Terminate when all the nodes are coalesced in all the subgraphs, then build back up

Why Does It Work?

- Proof by induction:
  - **Cut property:** edges contracted in STEP 1 are in the MST.
  - **Cycle property:** edges deleted in STEP 2 not in the MST.
  - Remaining edges of the *original graph's* MST form an MST of the *contracted graph*.
  - By the inductive hypothesis, MST of the remaining graph is correctly determined in recursive call of STEP 3.

How Long Does It Take?

- WORST CASE:
  - Total time spent in steps 1-3, ignoring recursion, is linear in the number of edges.
  - Step 1 is two steps of Buruvka, linear
  - Step 2 is linear using the modified Dixon-Rauch-Tarjan verification algorithm
  - Total running time is bounded by the total number of edges and in all recursive subproblems
  - So how many edges are there in all of the problems?

Worst Case, Continued

- Assume a graph with n vertices and m edges. m >= n/2.
- Each invocation generates at most two recursive subproblems.
- Consider the binary tree of recursive subproblems, where the root is the initial problem.
- First subproblem is the left child (step 2), second is right child (step 3).

Worst Case, Continued

- At depth d, the tree of subproblems has at most 2^d nodes, each a problem on a graph of at most $n/4^d$ vertices.
- The the depth of the tree is at most $\log_v n$, and there are at most 2n vertices total in the original problem and all subproblems.
- A subproblem at depth d contains at most $(n/4^d)^2/2$ edges. The total number of edges in all subproblems at an recursive depth d is therefore, at most, m.

Worst Case, Finally!

- Since there are O(logn) depths in the tree, the total number of edges in all recursive subproblems is O(mlogn)
- This is the same as Baruvka's original algorithm.

Best Case

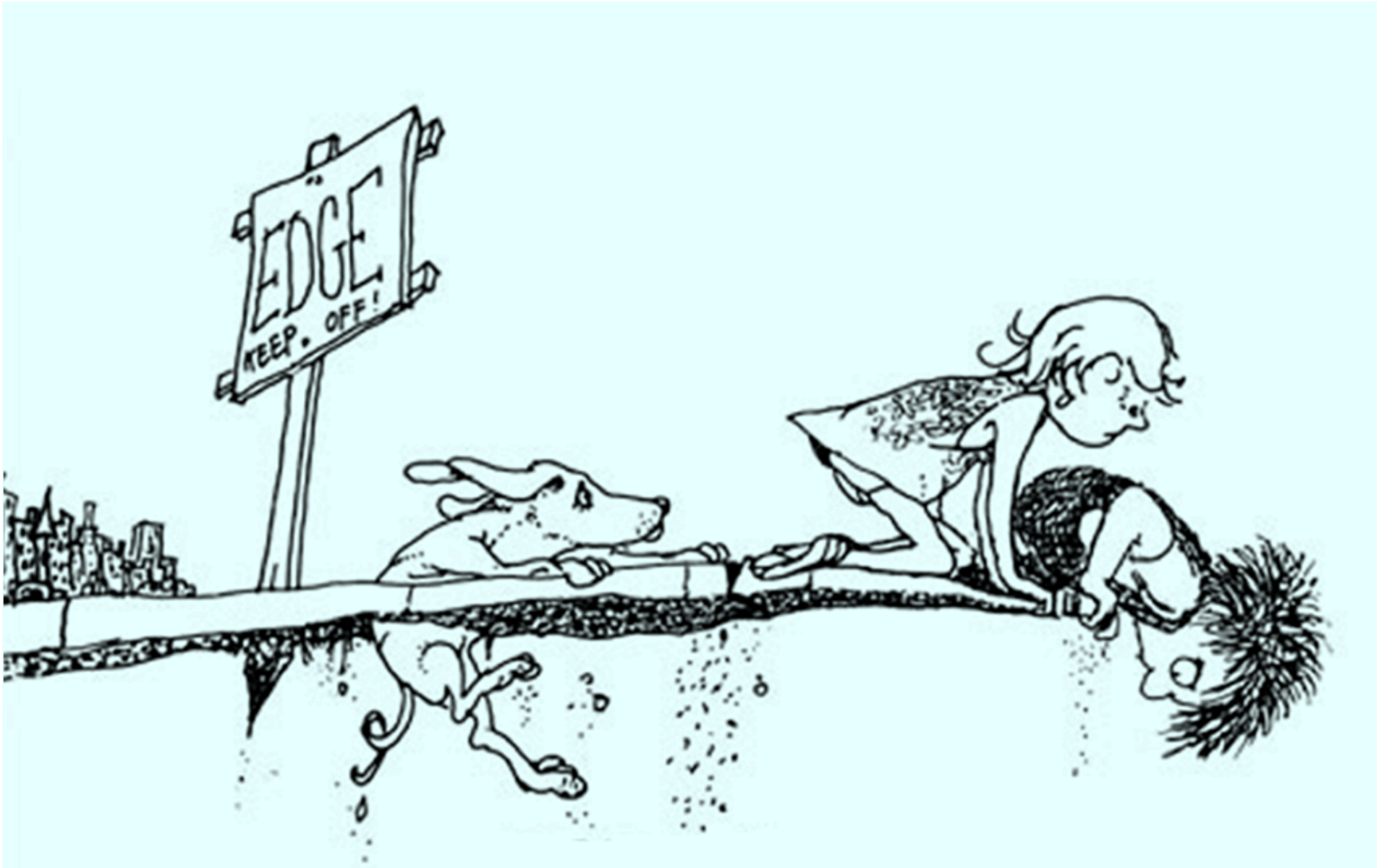
- The expected running time is O(m)
- The expected number of edges in the left subproblem is at most half the expected number of edges in its parent.
- m + (m/2) + (m/4) … = 2m
- The sum of expected number of edges in every subproblem in the left side of the tree is therefore at most 2m.

Best Case, continued

- The total number of edges is bounded by 2m and the expected total number of edges in all right subproblems
- The number of edges in the right subproblem is at most twice the number of vertices in the subproblem.

Best Case, continued

- The total number of vertices in all right subproblems is at most n/2.
- So the expected number of edges in the original problem and all subproblems is at most 2m+n
- The algorithm runs in O(m) time with probability 1 - (exp(-Omega(m))



THE END

…questions?