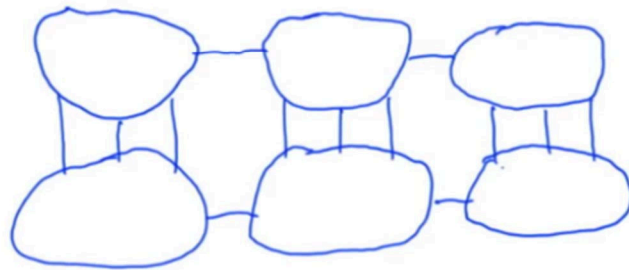


Save the Turtles

Save The Turtles



14

cut the bridges so plastic stays in one piece but with no loops (turtle traps).

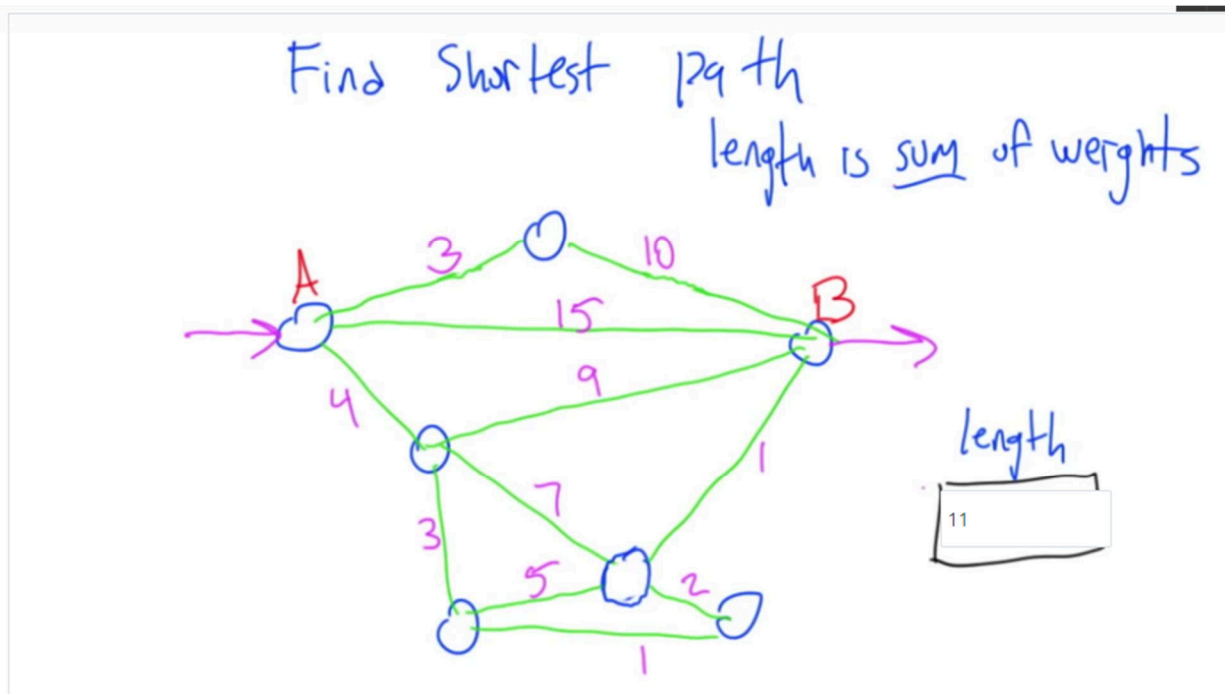
Strength of Connections

Compute Connection Strength

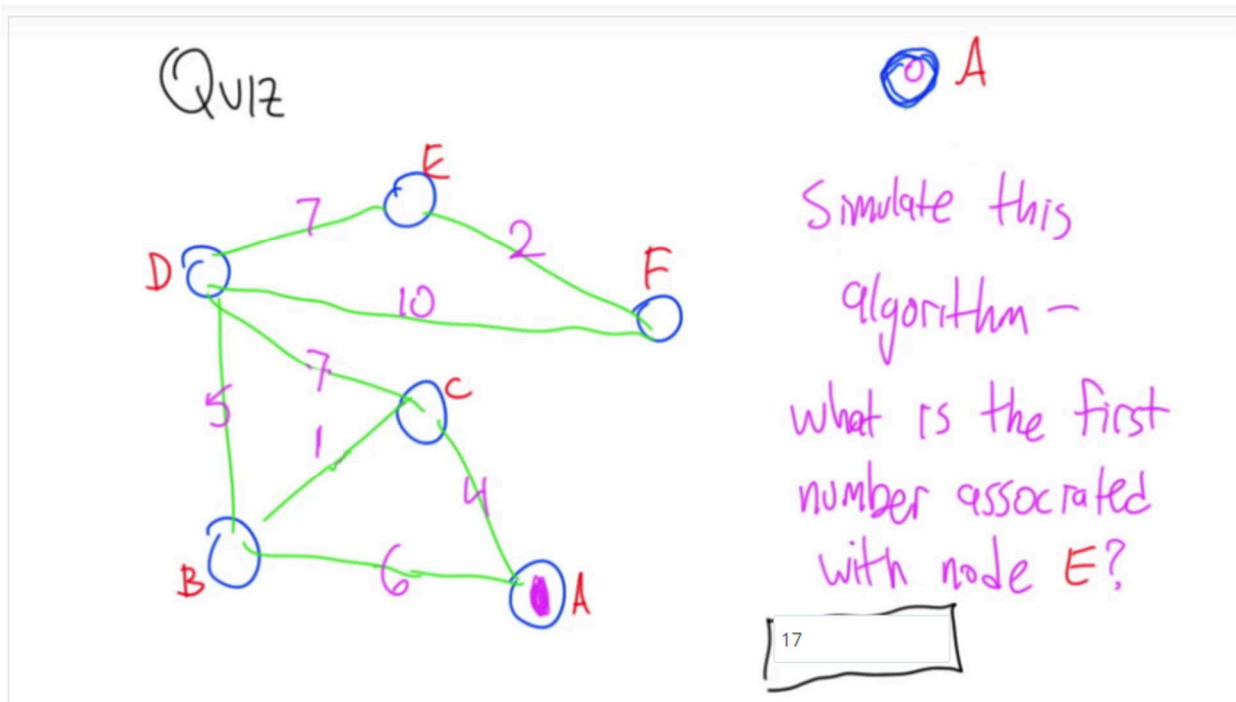
Write a program to read the Marvel graph and put a strength value on each link. Which link has the highest strength value?

- ☒ HUMAN TORCH/JOHNNY S - THING/BENJAMIN J. GR
- ☐ INVISIBLE WOMAN/SUE - MR. FANTASTIC/REED R
- ☐ SPIDER-MAN/PETER PARKER - WATSON-PARKER, MARY
- ☐ CAPTAIN AMERICA - IRON MAN/TONY STARK

Find the Shortest Path



Simulate this Algorithm



What is Missing?

Running Time of Dijkstra n, m

For each node :

- Find the shortest dist-so-far
- remove it
- Check each of its neighbors

- depends on how "find shortest" is implemented
- depends on the degree
- depends if graph is dense or sparse

Implementing Shortest Distance

Implementing shortest-dist-node (dist-so-far)

dist-so-far
nodes, values

Smallest (& delete it) $\Theta(n)$

loop through list - $\Theta(n)$

Running time of dijkstra with this implementation?

- $\Theta(n+m)$
- $\Theta(nm)$
- $\Theta(n^2)$
- $\Theta(n^2+nm)$

Using Heaps

Running Time of Dijkstra n, m with Heaps!

For each node

- Find the shortest dist-so-far
- remove it
- Check each of its neighbors possibly reducing distance

• $\Theta(n^3)$

• $\Theta(n^2)$

• $\Theta(n \log n)$

• $\Theta(m \log n)$

• $\Theta(nm \log n)$

Randomizing Clustering Coefficient

```
9 def expected_C(G, v):  
0     # G[v].keys() is the set of neighbors of v  
1     neighbors = G[v].keys()  
2     degree = len(neighbors)  
3     # x in G[w][x] if x and w are connected in the graph (C[w,x])  
4     return clustering_coefficient(G, v)  
5
```