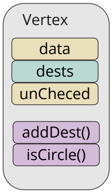
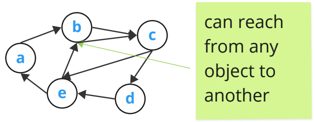
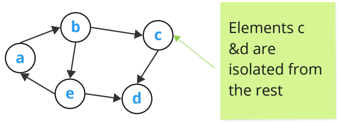
SCC Strongly Connected Component



Template class that can store different types of data, and points to other Vertex objects.

- The '**data**' element set by the constructor.
- **addDest**() function adds another destination to '**dests**' list and increases '**unChecked**' number.
- **isCircle**() function checks the destinations and determines if the connection is strong.



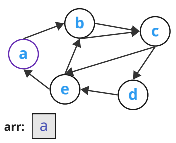


The function **isCircle**, which is part of Vertex class, receives vector of vertex pointers - **arr** as parameter, and operates in the following way:

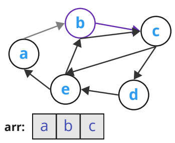
1. check whether container object is last in the graph, or it points to more objects. this is done by **unChecked** variable. if it last:
   when container equal first vector element, circle complete, return 1. else return 0.
2. when there are more unchecked destinations, add next detonation pointer to the array, and call it's **isCircle** function, to complete the circle.
   when returned value is true, it means the called object reached first element in **arr**.
3. At this point check again if there any unchecked destinations. Now call the next destination with temporary vector containing pointer to container object. the circle should reach back to it

Example of operating **isCircle** starting with object a:

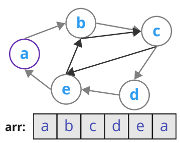
1. start by calling **a.isCircle**



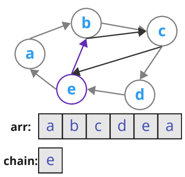
3. Object **b** call **c.isCircle**



5. Object **a** returns **true**

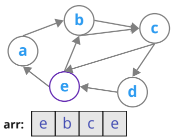


6. Object **e** creates **chain**

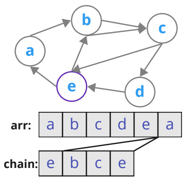


1. Vector **arr** contains pointer to **a**. Container function calls **a.isCircle**.
2. Object **a** contains one destination to **b**. Reduce **unChecked**, adds **b** to **arr** and calls **b.isCircle**.
3. Object **b** repeat the same process with pointer to **c**.
4. Process continues from **c** to **d**, **e** and **a**.
5. Object **a** contains pointer to **b** in its destinations array, but it already checked. the function test that **arr**[0] points to a, so it returns **true**.
6. Object **e** has destination to **b**, so it creates temporary vector - **chain**, started with pointer to itself.
7. Object **b** second pointer points also to **c**, which points back to **e**.
8. Function of object e starts again (recursively), no more destinations, and **arr**[0] points to itself, returns true.
9. Functions of objects c and b also returns true to previous instance of **e**, from step 6.
10. Object **e** inserts the chain to vector **arr**, and returns true.
11. Recursively the functions return true to the container function.

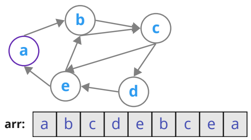
7. Object **b** calls **c** and **e**



10. Object **e** insert the chain and returns **true**



11. Each function in turn returns **true**



a
b
c
d
e
arr:
a
b
c
d
e
b
c
e
a