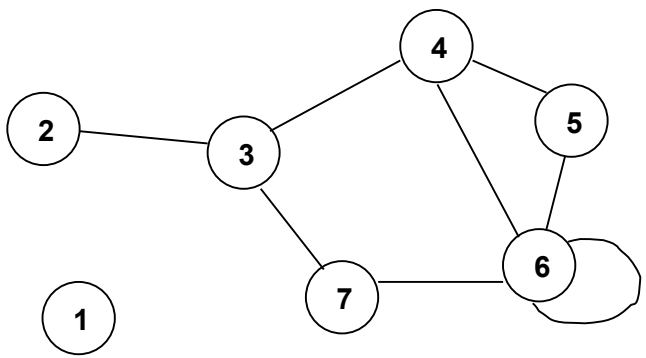
מסלול ומעגל אוילר

מסלול אוילר הוא מסלול בגרף העובר בכל קשת בו בדיוק פעם אחת.

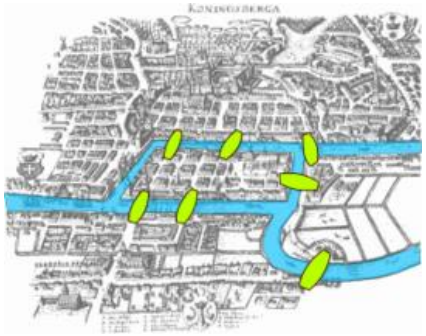
מעגל אוילר הוא מסלול אוילר מעגלי (מתחיל ונגמר באותו צומת).

המסלול ו**המעגל** נקראים על שם **לאונרד אוילר** שעסק בהם לראשונה כאשר פתר את בעיית הגשרים של קניגסברג.

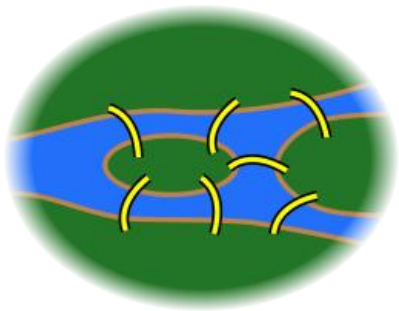
דרגה של צומת מתארת את מספר הקשתות המקושרות לצומת מסוים. זהו המידע הבסיסי ביותר שאפשר למסור על צומת בגרף ,משום שהוא מתאר את תמונת העולם המקומית של הקודקודים שלו. דרגה של צומת v מסומנת כ-$\deg(v)$.



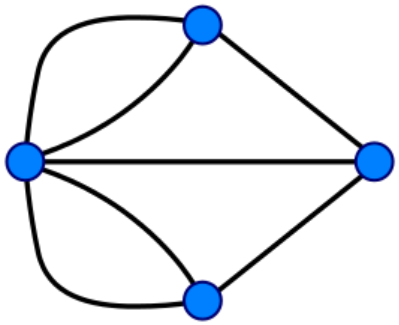
צומת	1	2	3	4	5	6	7
דרגה	0	1	3	3	2	5	2



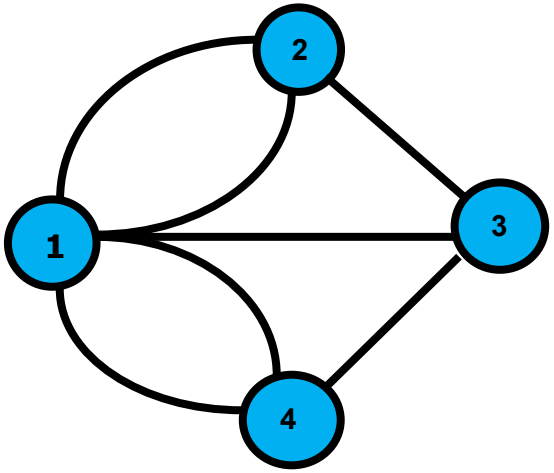
→



→

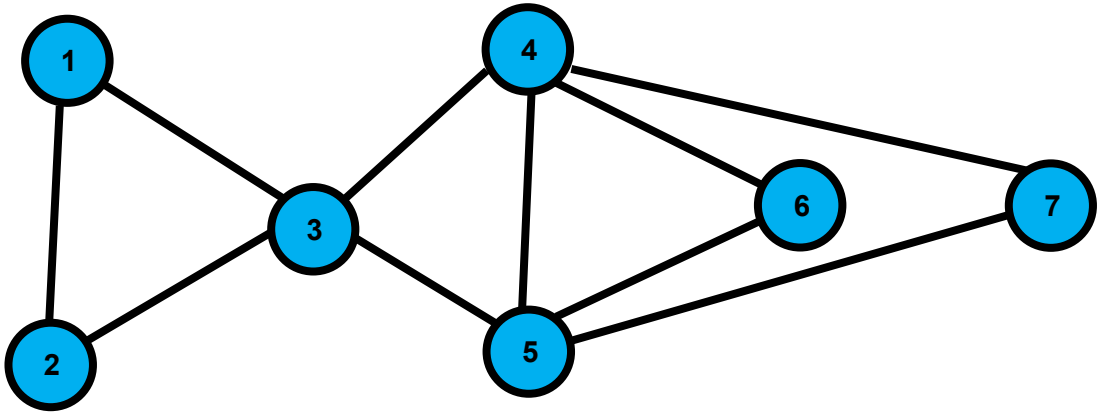


הגשרים של קניגסברג



צומת	1	2	3	4
דרגה	5	3	3	3

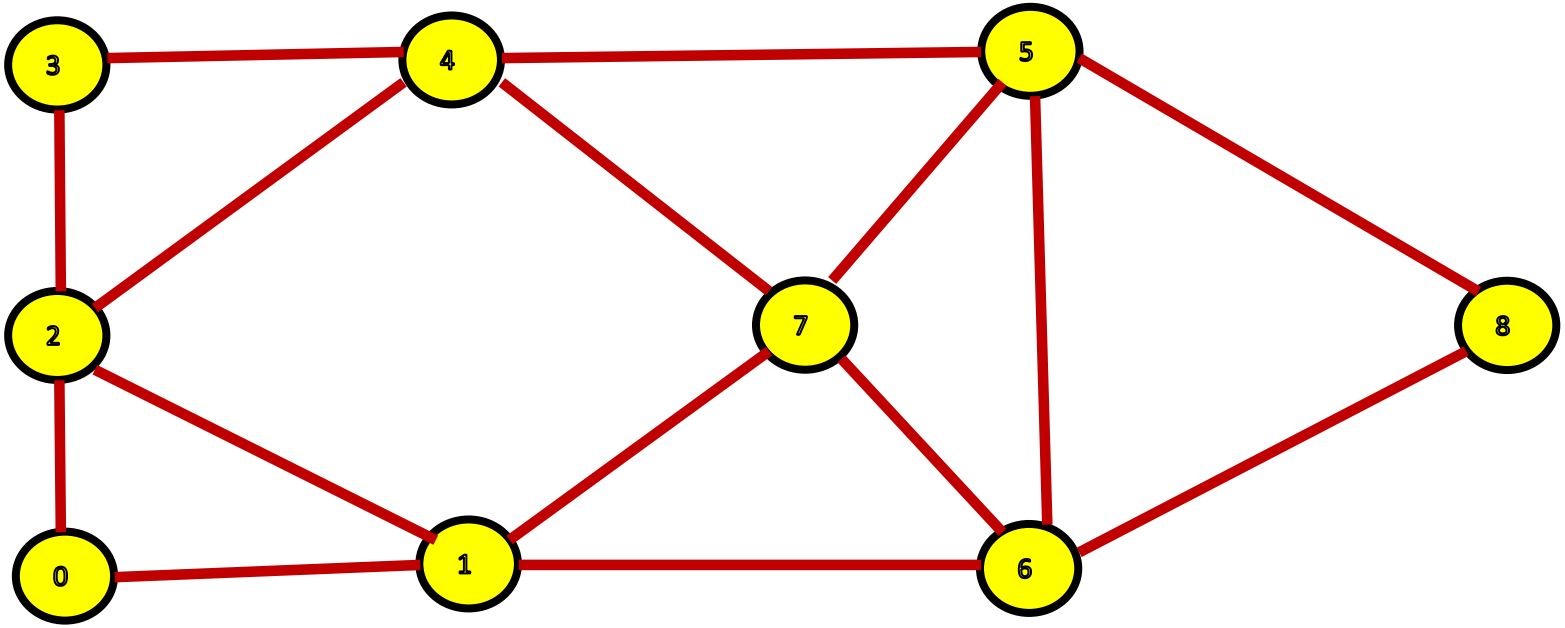
מעגל אוילר: בגרף קשיר קיים מעגל אוילר **אם ורק אם** כל הדרגות זוגיות.



צומת	1	2	3	4	5	6	7
דרגה	2	2	4	4	4	2	2

מעגל אוילר: 3, 4, 7, 5, 6, 4, 5, 3, 1, 2, 3

Example



ArrayList<Integer>[] graph

0 :	1	2		
1 :	0	2	6	7
2 :	0	1	3	4
3 :	2	4		
4 :	2	3	5	7
5 :	4	6	7	8
6 :	1	5	7	8
7 :	1	4	5	6
8 :	5	6		

ArrayList<Integer> ec מעגל

[]

Stack<Integer> st מסלול

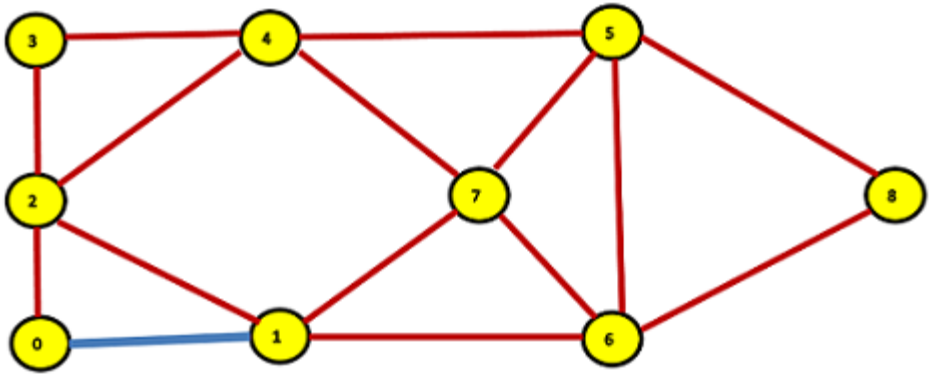
[]

graph : start 0 next 1

0 : ~~1~~ 2
1 : ~~0~~ 2 6 7
2 : 0 1 3 4
3 : 2 4
4 : 2 3 5 7
5 : 4 6 7 8
6 : 1 5 7 8
7 : 1 4 5 6
8 : 5 6

st [0, 1]

ec []

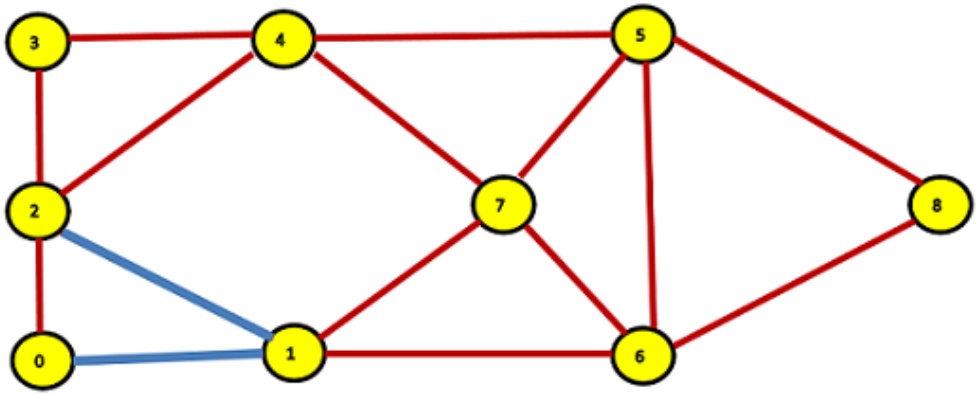


graph : **start** **1** **next** **2**

0 :	~~1~~	2		
1 :	~~0~~	~~2~~	6	7
2 :	0	~~1~~	3	4
3 :	2	4		
4 :	2	3	5	7
5 :	4	6	7	8
6 :	1	5	7	8
7 :	1	4	5	6
8 :	5	6		

st [0, 1, 2]

ec []

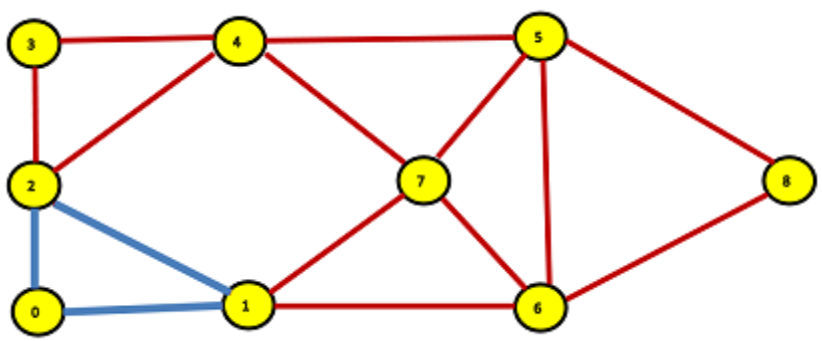


graph : start 2 next 0

0 : ~~1~~ ~~2~~
1 : ~~0~~ ~~2~~ 6 7
2 : ~~0~~ ~~1~~ 3 4
3 : 2 4
4 : 2 3 5 7
5 : 4 6 7 8
6 : 1 5 7 8
7 : 1 4 5 6
8 : 5 6

st [0, 1, 2, 0] → st [0, 1, 2]

ec [0]

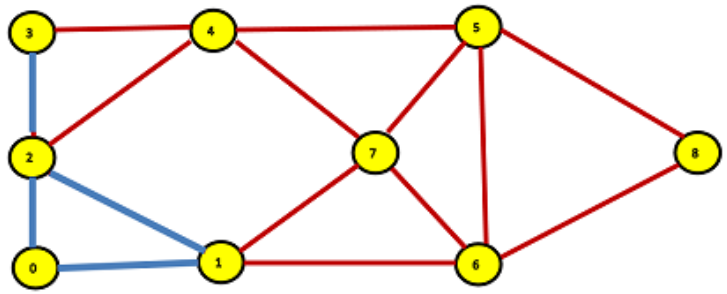


graph : **start** **2** **next** **3**

0 :	~~1~~	~~2~~		
1 :	~~0~~	~~2~~	6	7
2 :	~~0~~	~~1~~	~~3~~	4
3 :	~~2~~	4		
4 :	2	3	5	7
5 :	4	6	7	8
6 :	1	5	7	8
7 :	1	4	5	6
8 :	5	6		

st [0, 1, 2, 3]

ec [0]

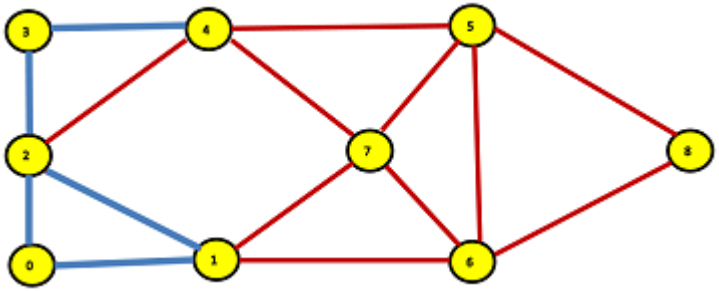


graph : **start** **3** **next** **4**

0 :	~~1~~	~~2~~		
1 :	~~0~~	~~2~~	6	7
2 :	~~0~~	~~1~~	~~3~~	4
3 :	~~2~~	~~4~~		
4 :	2	~~3~~	5	7
5 :	4	6	7	8
6 :	1	5	7	8
7 :	1	4	5	6
8 :	5	6		

st [0, 1, 2, 3, 4]

ec [0]

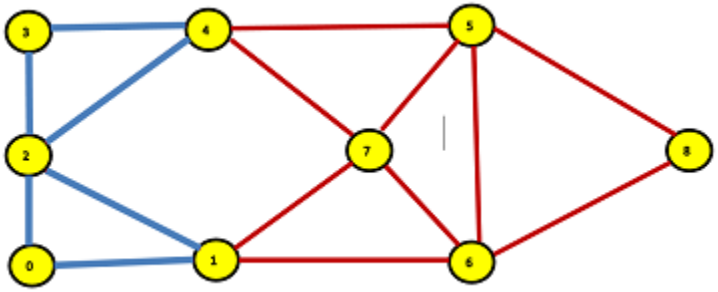


graph : **start** **4** **next** **2**

0 :	~~1~~	~~2~~		
1 :	~~0~~	~~2~~	6	7
2 :	~~0~~	~~1~~	~~3~~	~~4~~
3 :	~~2~~	~~4~~		
4 :	~~2~~	~~3~~	5	7
5 :	4	6	7	8
6 :	1	5	7	8
7 :	1	4	5	6
8 :	5	6		

st [0, 1, 2, 3, 4, 2] → **st [0, 1, 2, 3, 4]**

ec [0, 2]

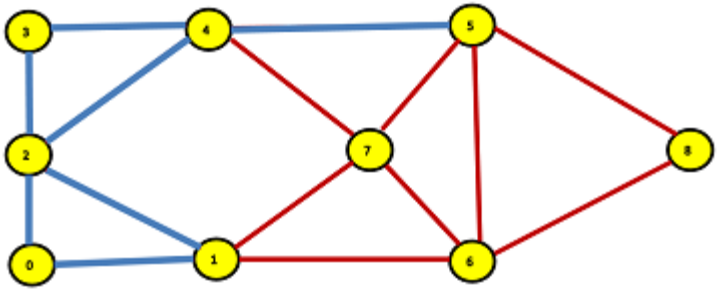


graph : **start** **4** **next** **5**

0 :	~~1~~	~~2~~		
1 :	~~0~~	~~2~~	6	7
2 :	~~0~~	~~1~~	~~3~~	~~4~~
3 :	~~2~~	~~4~~		
4 :	~~2~~	~~3~~	~~5~~	7
5 :	~~4~~	6	7	8
6 :	1	5	7	8
7 :	1	4	5	6
8 :	5	6		

st [0, 1, 2, 3, 4, 5]

ec [0, 2]

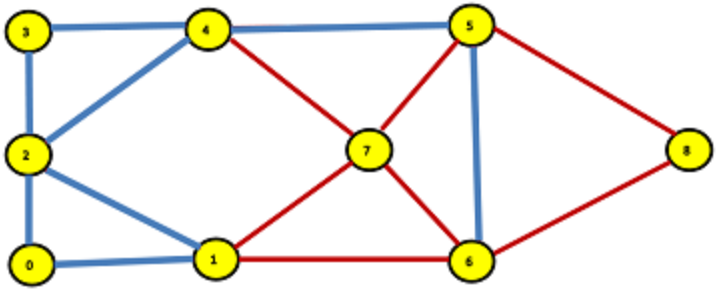


graph : **start** **5** **next** **6**

0 :	~~1~~	~~2~~		
1 :	~~0~~	~~2~~	6	7
2 :	~~0~~	~~1~~	~~3~~	~~4~~
3 :	~~2~~	~~4~~		
4 :	~~2~~	~~3~~	~~5~~	7
5 :	~~4~~	~~6~~	7	8
6 :	1	~~5~~	7	8
7 :	1	4	5	6
8 :	5	6		

st [0, 1, 2, 3, 4, 5, 6]

ec [0, 2]

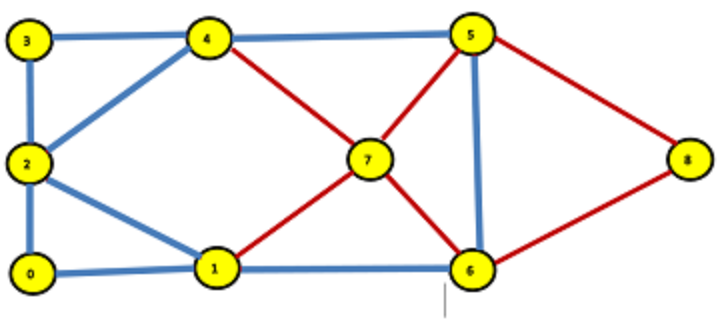


graph : **start** **6** **next** **1**

0 :	~~1~~	~~2~~		
1 :	~~0~~	~~2~~	~~6~~	7
2 :	~~0~~	~~1~~	~~3~~	~~4~~
3 :	~~2~~	~~4~~		
4 :	~~2~~	~~3~~	~~5~~	7
5 :	~~4~~	~~6~~	7	8
6 :	~~1~~	~~5~~	7	8
7 :	1	4	5	6
8 :	5	6		

st [0, 1, 2, 3, 4, 5, 6, 1]

ec [0, 2]

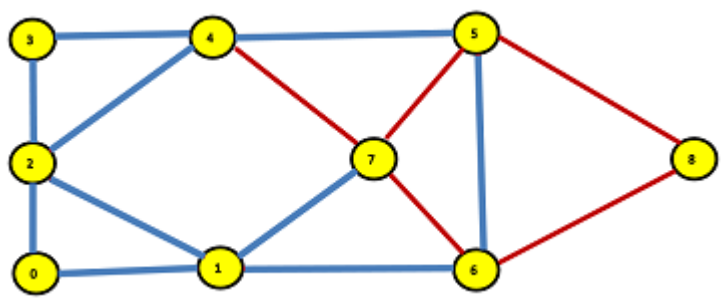


graph : **start** **1** **next** **7**

0 :	~~1~~	~~2~~		
1 :	~~0~~	~~2~~	~~6~~	~~7~~
2 :	~~0~~	~~1~~	~~3~~	~~4~~
3 :	~~2~~	~~4~~		
4 :	~~2~~	~~3~~	~~5~~	7
5 :	~~4~~	~~6~~	7	8
6 :	~~1~~	~~5~~	7	8
7 :	~~1~~	4	5	6
8 :	5	6		

st [0, 1, 2, 3, 4, 5, 6, 1, 7]

ec [0, 2]



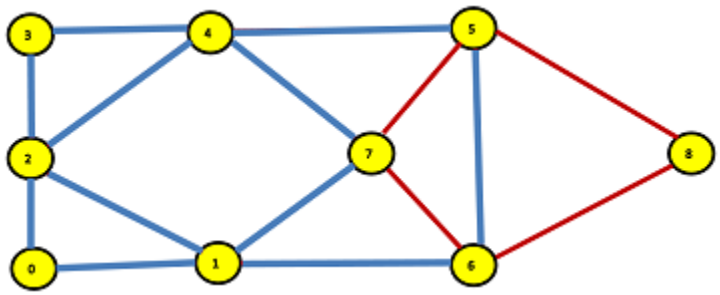
graph : **start** **7** **next** **4**

0 : ~~1~~ ~~2~~
1 : ~~0~~ ~~2~~ ~~6~~ ~~7~~
2 : ~~0~~ ~~1~~ ~~3~~ ~~4~~
3 : ~~2~~ ~~4~~
4 : ~~2~~ ~~3~~ ~~5~~ ~~7~~
5 : ~~4~~ ~~6~~ 7 8
6 : ~~1~~ ~~5~~ 7 8
7 : ~~1~~ ~~4~~ 5 6
8 : 5 6

st [0, 1, 2, 3, 4, 5, 6, 1, 7, 4]

st [0, 1, 2, 3, 4, 5, 6, 1, 7]

ec [0, 2, 4]

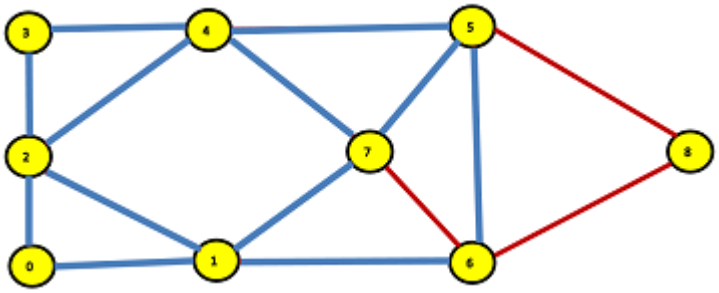


graph : **start** **7** **next** **5**

0 : ~~1~~ ~~2~~
1 : ~~0~~ ~~2~~ ~~6~~ ~~7~~
2 : ~~0~~ ~~1~~ ~~3~~ ~~4~~
3 : ~~2~~ ~~4~~
4 : ~~2~~ ~~3~~ ~~5~~ ~~7~~
5 : ~~4~~ ~~6~~ ~~7~~ 8
6 : ~~1~~ ~~5~~ 7 8
7 : ~~1~~ ~~4~~ ~~5~~ 6
8 : 5 6

st [0, 1, 2, 3, 4, 5, 6, 1, 7, 5]

ec [0, 2, 4]

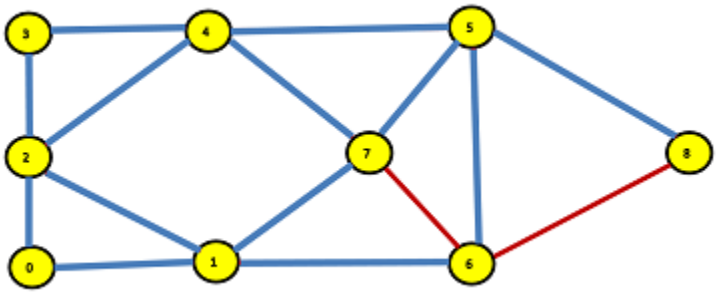


graph : **start** **5** **next** **8**

0 : ~~1~~ ~~2~~
1 : ~~0~~ ~~2~~ ~~6~~ ~~7~~
2 : ~~0~~ ~~1~~ ~~3~~ ~~4~~
3 : ~~2~~ ~~4~~
4 : ~~2~~ ~~3~~ ~~5~~ ~~7~~
5 : ~~4~~ ~~6~~ ~~7~~ ~~8~~
6 : ~~1~~ ~~5~~ 7 8
7 : ~~1~~ ~~4~~ ~~5~~ 6
8 : ~~5~~ 6

st [0, 1, 2, 3, 4, 5, 6, 1, 7, 5, 8]

ec [0, 2, 4]

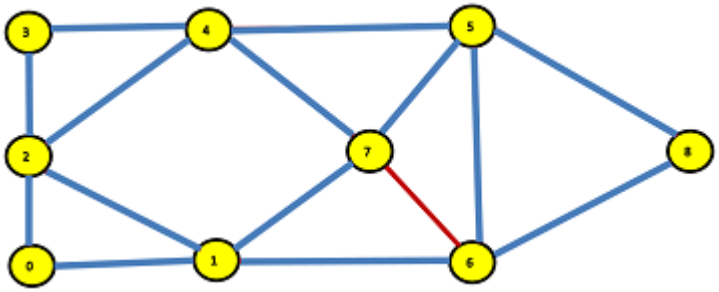


graph : **start** **8** **next** **6**

0 : ~~1~~ ~~2~~
1 : ~~0~~ ~~2~~ ~~6~~ ~~7~~
2 : ~~0~~ ~~1~~ ~~3~~ ~~4~~
3 : ~~2~~ ~~4~~
4 : ~~2~~ ~~3~~ ~~5~~ ~~7~~
5 : ~~4~~ ~~6~~ ~~7~~ ~~8~~
6 : ~~1~~ ~~5~~ 7 ~~8~~
7 : ~~1~~ ~~4~~ ~~5~~ 6
8 : ~~5~~ ~~6~~

st [0, 1, 2, 3, 4, 5, 6, 1, 7, 5, 8, 6]

ec [0, 2, 4]



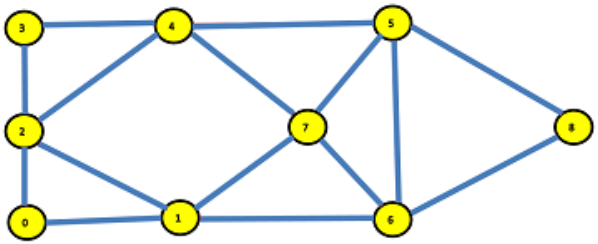
graph : **start** **6** **next** **7**

0 : ~~1~~ ~~2~~
1 : ~~0~~ ~~2~~ ~~6~~ ~~7~~
2 : ~~0~~ ~~1~~ ~~3~~ ~~4~~
3 : ~~2~~ ~~4~~
4 : ~~2~~ ~~3~~ ~~5~~ ~~7~~
5 : ~~4~~ ~~6~~ ~~7~~ ~~8~~
6 : ~~1~~ ~~5~~ ~~7~~ ~~8~~
7 : ~~1~~ ~~4~~ ~~5~~ ~~6~~
8 : ~~5~~ ~~6~~

st [0, 1, 2, 3, 4, 5, 6, 1, 7, 5, 8, 6, 7]

st [0, 1, 2, 3, 4, 5, 6, 1, 7, 5, 8, 6]

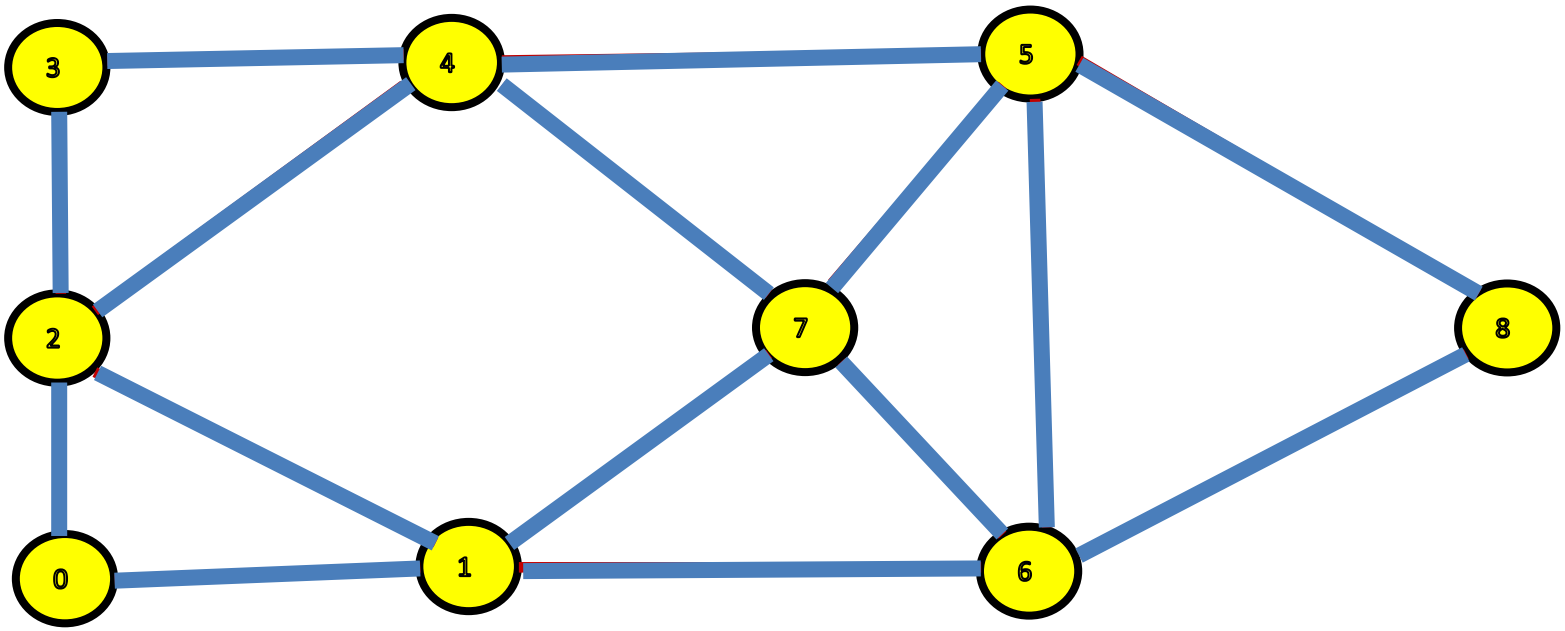
ec [0, 2, 4, 7]



End of process

ec [0, 2, 4, 7]

st [0, 1, 2, 3, 4, 5, 6, 1, 7, 5, 8, 6]

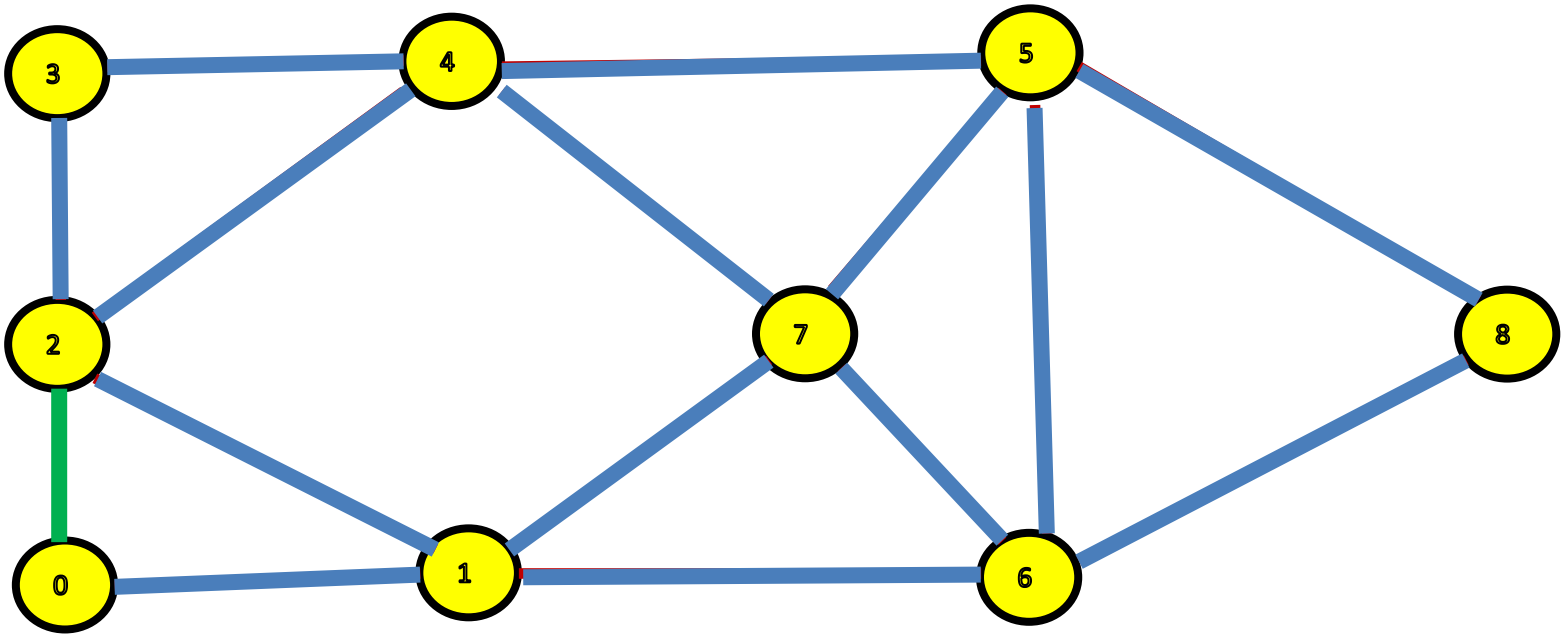


ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]

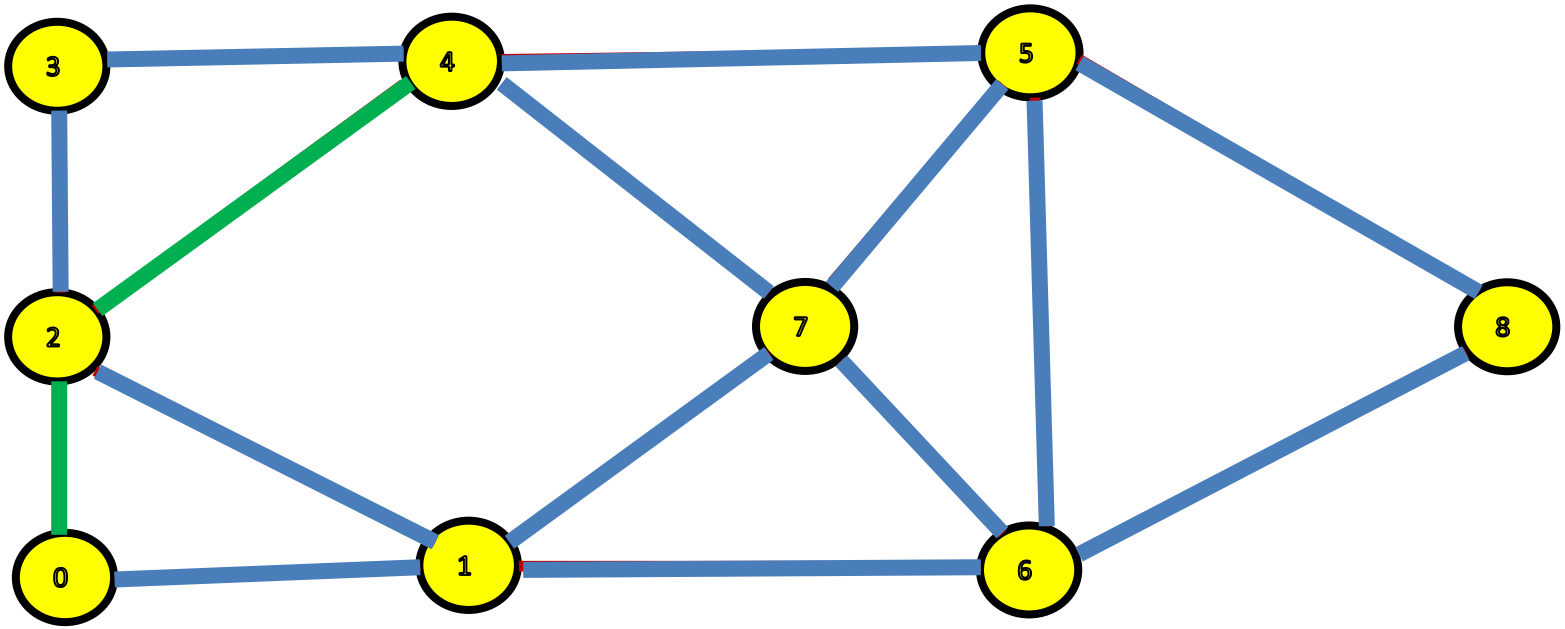
st

ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]

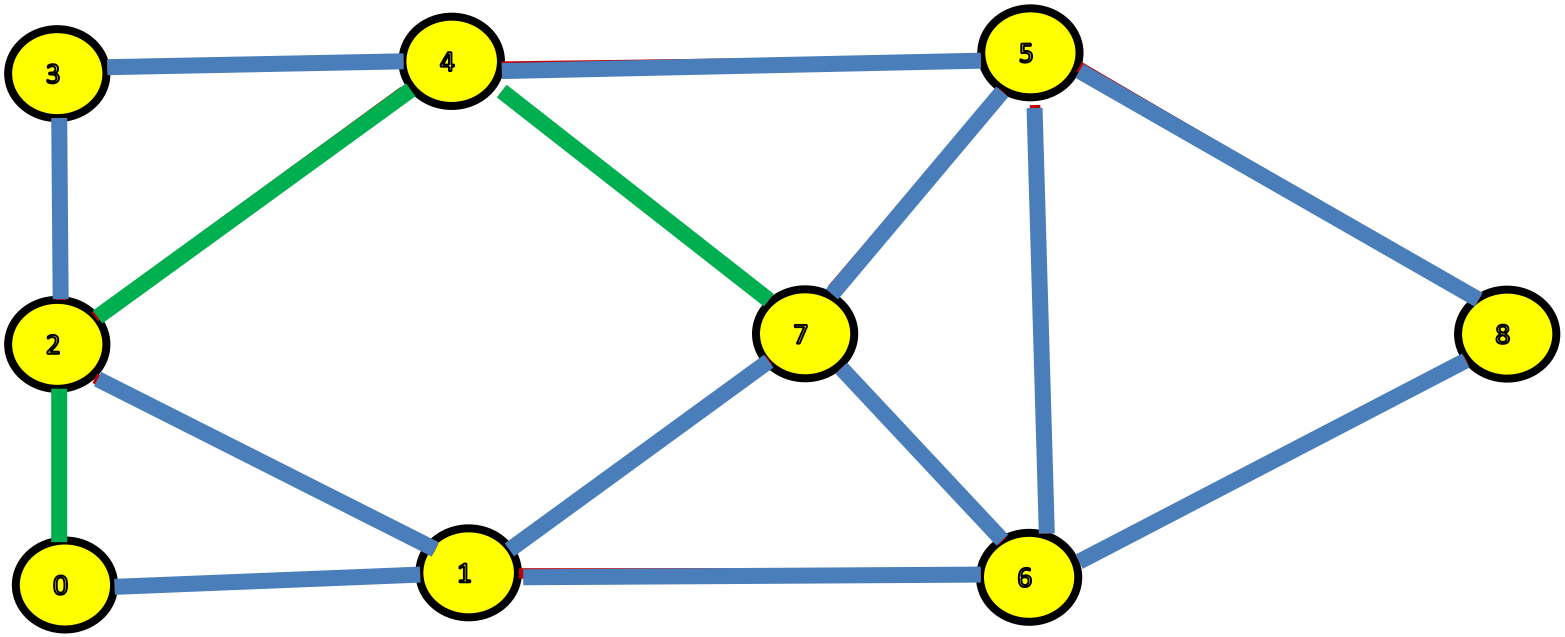
check Eulerian cycle :



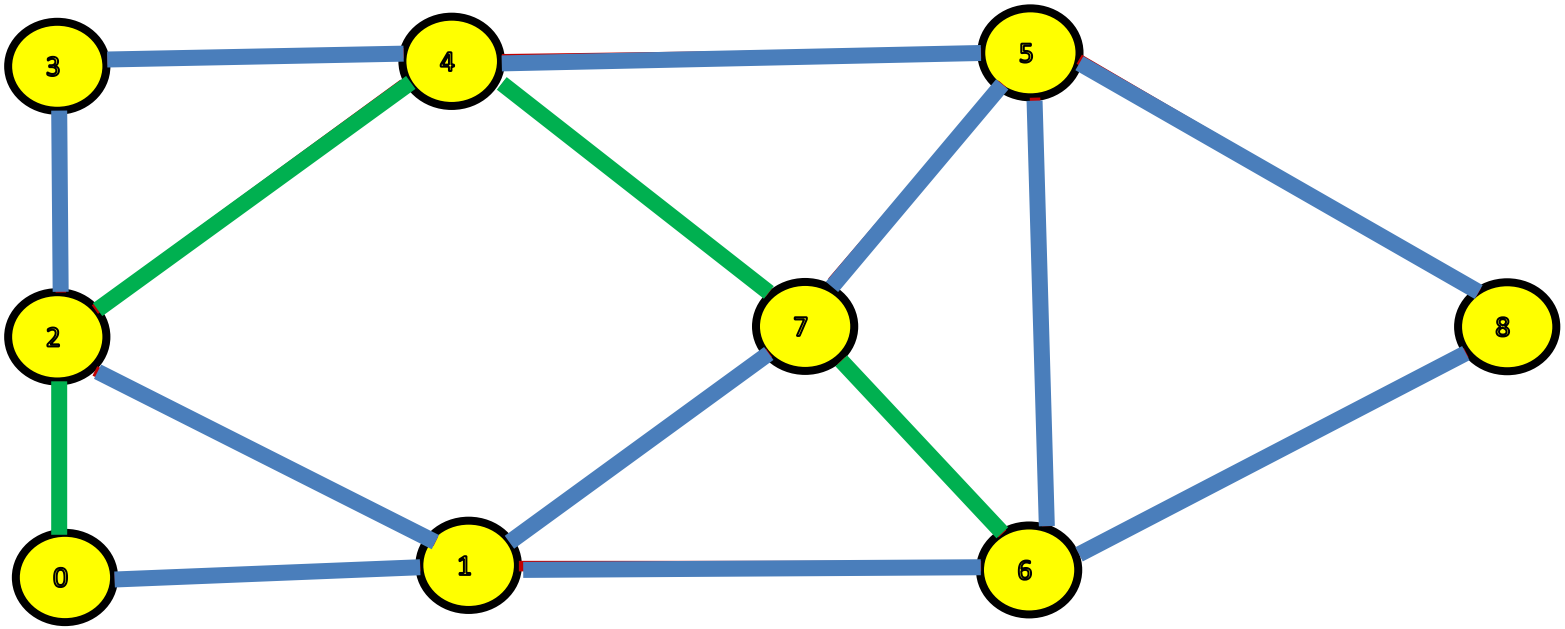
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



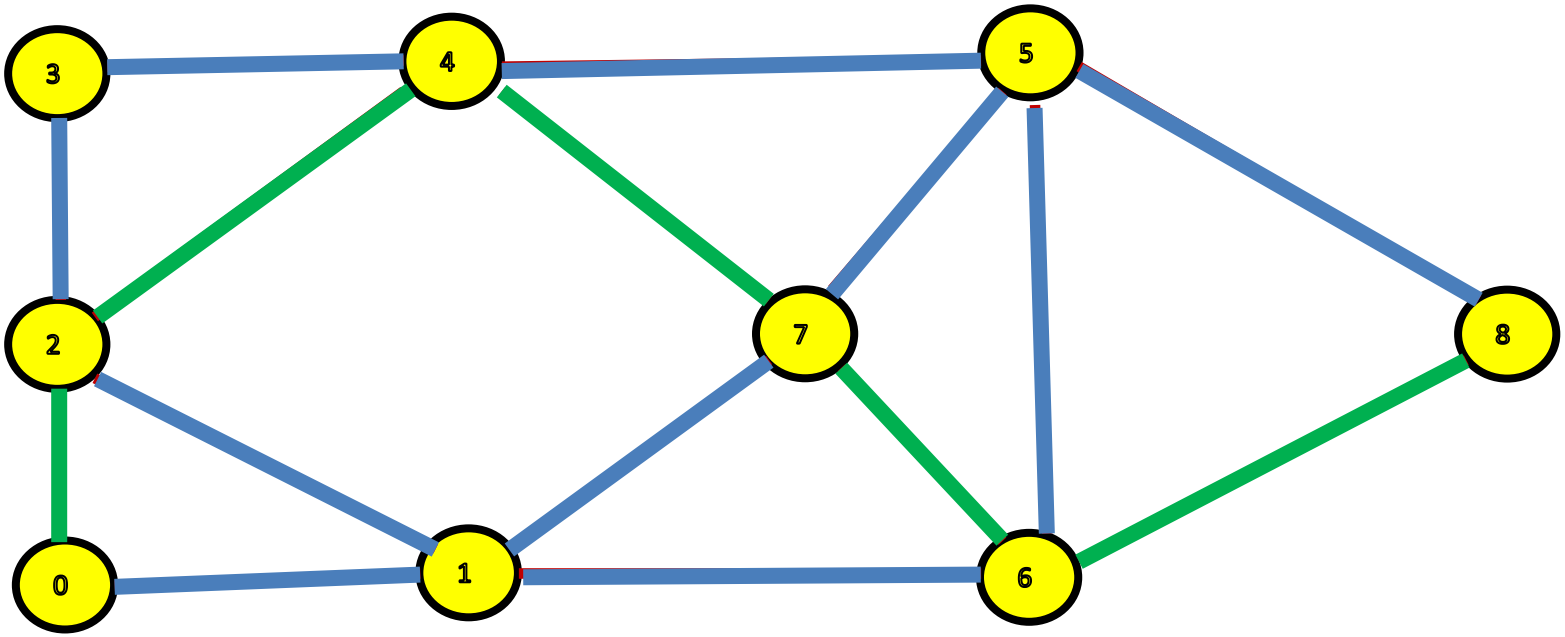
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



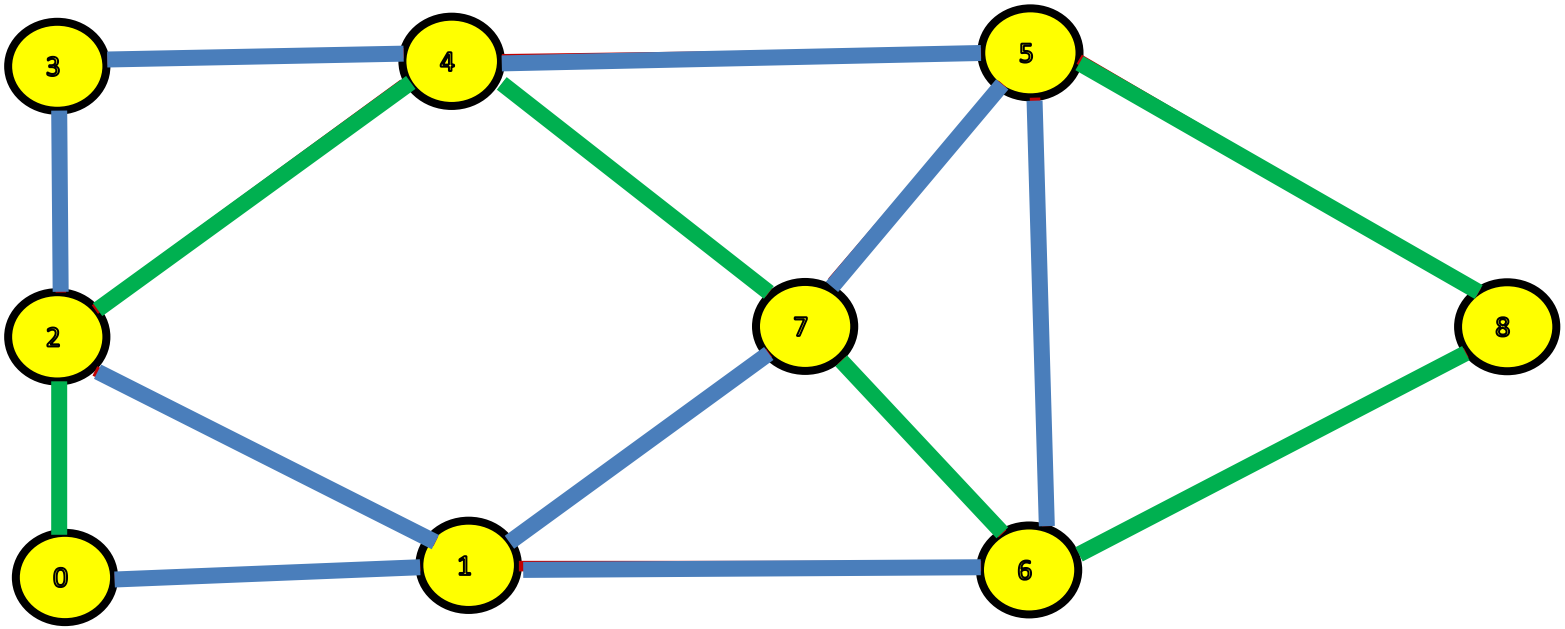
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



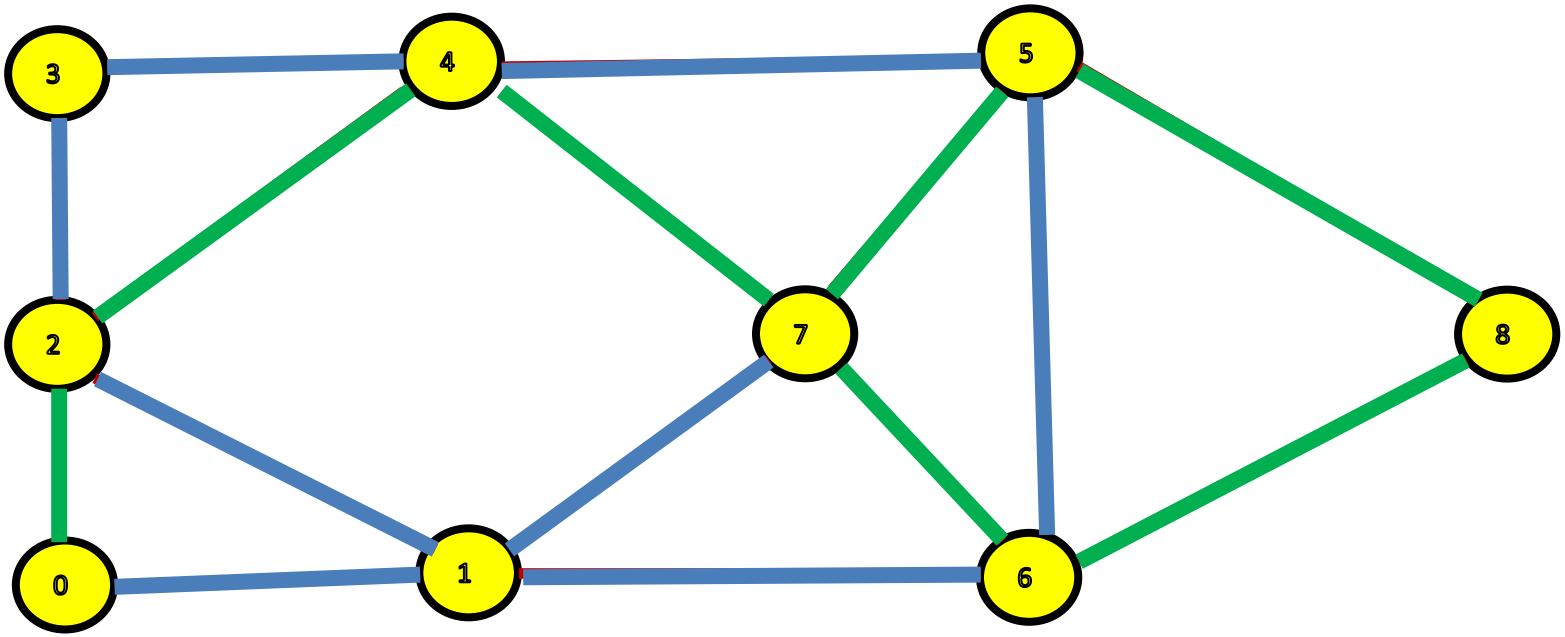
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



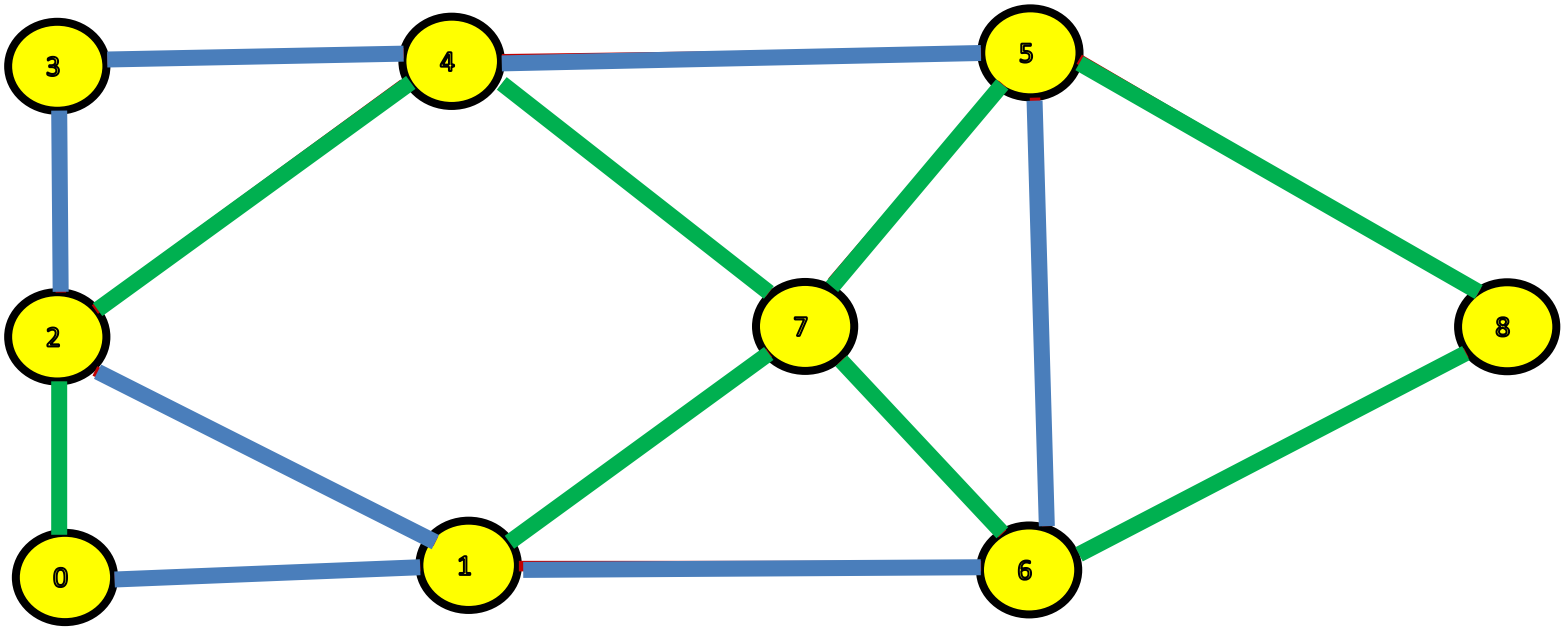
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



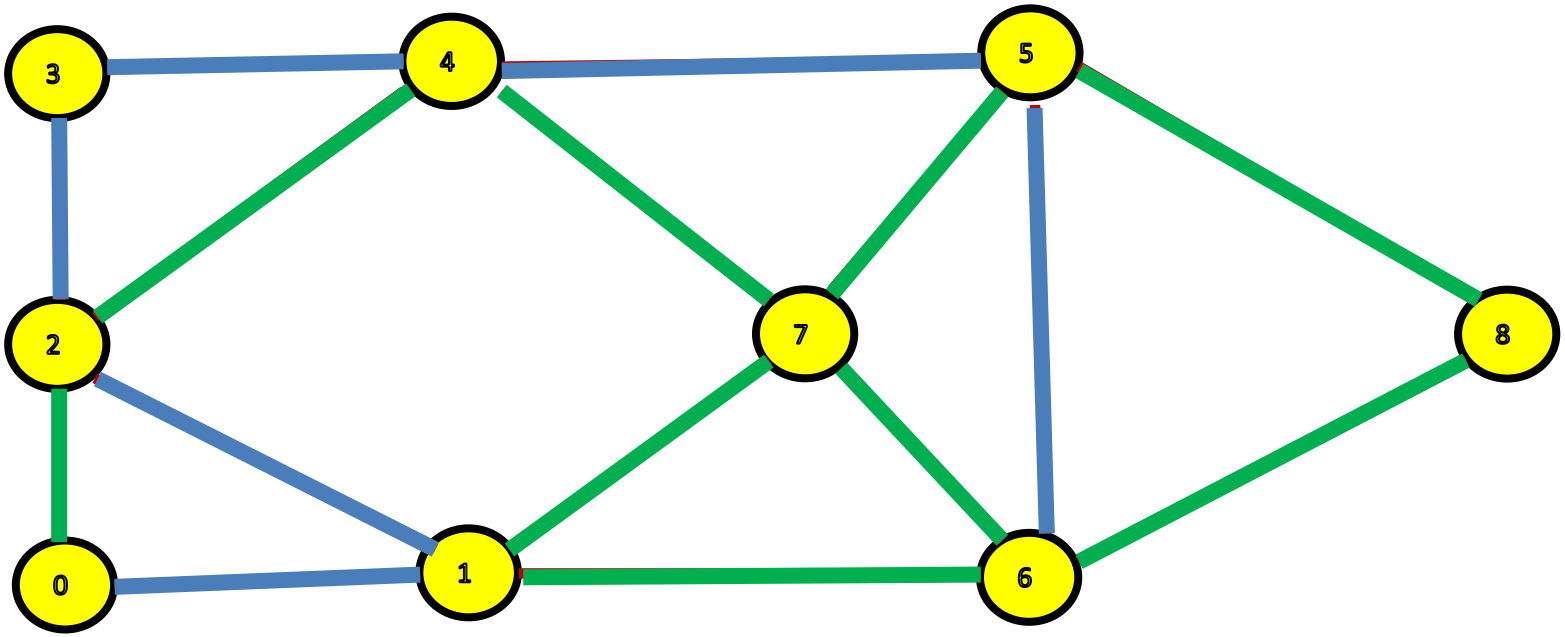
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



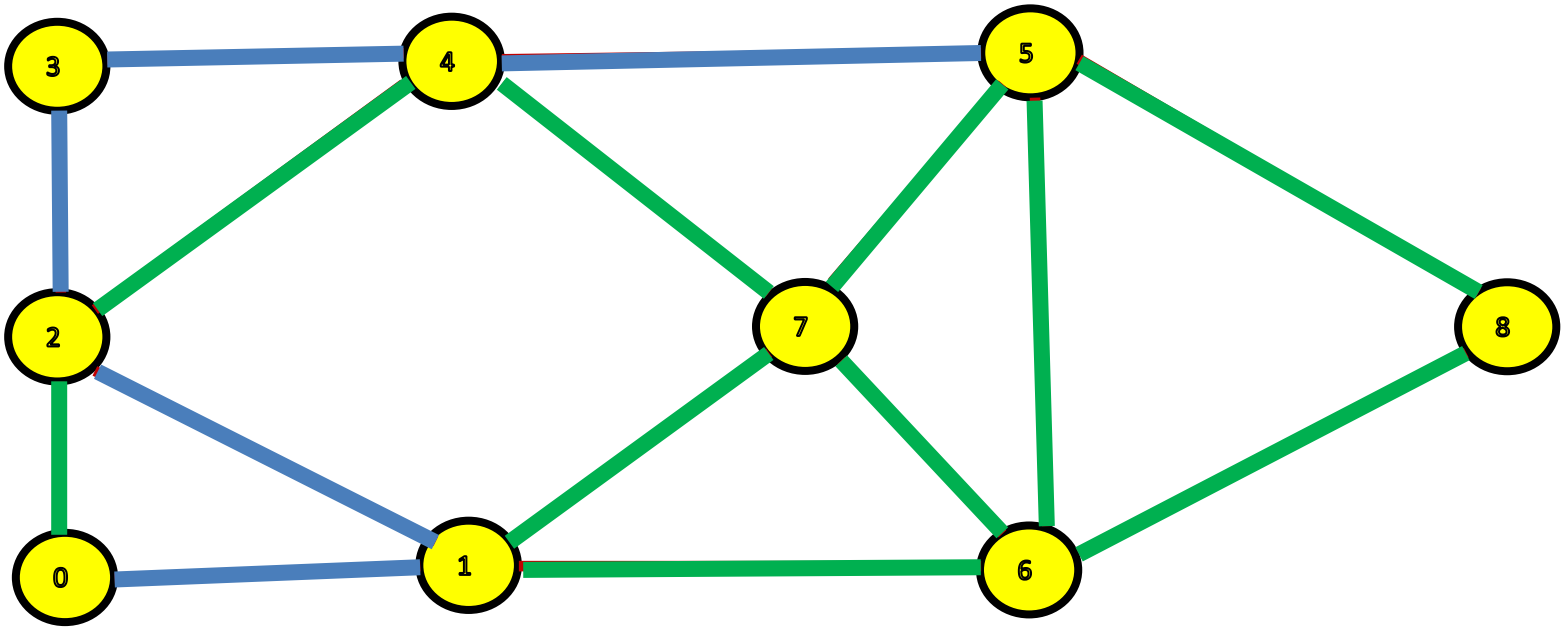
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



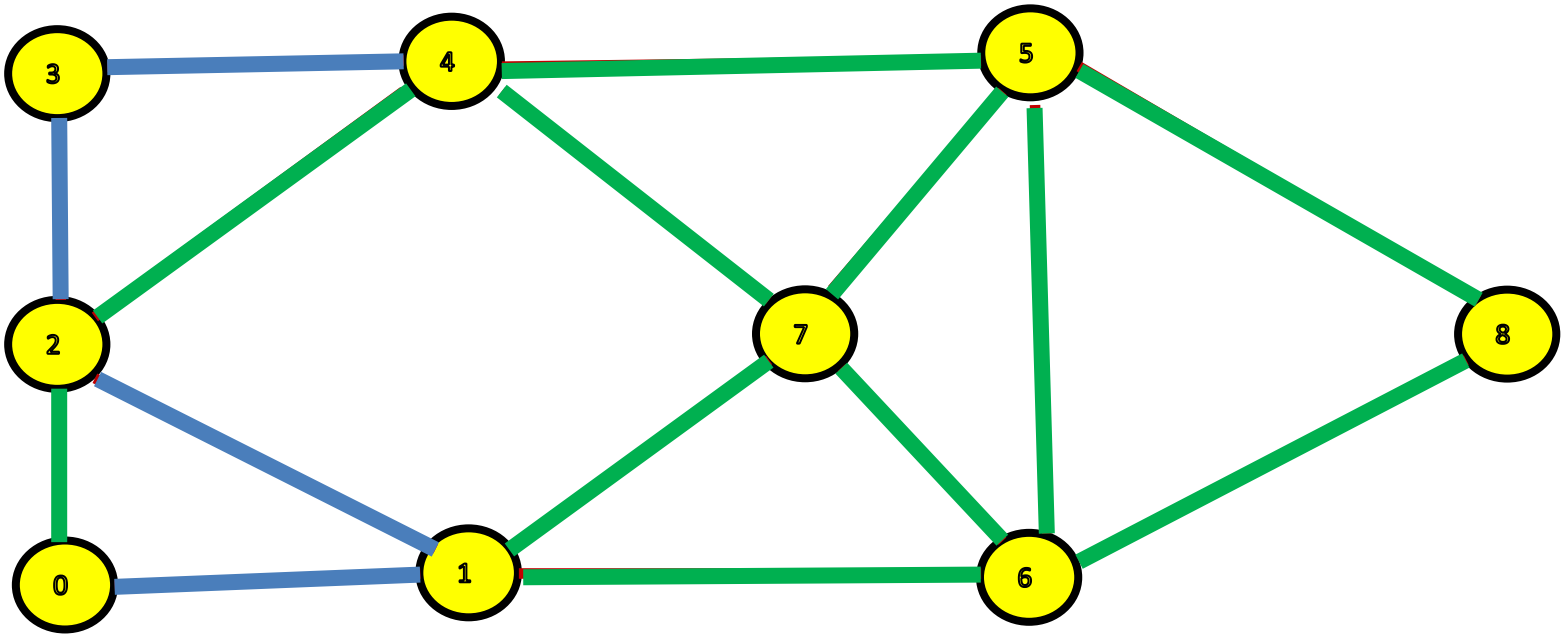
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



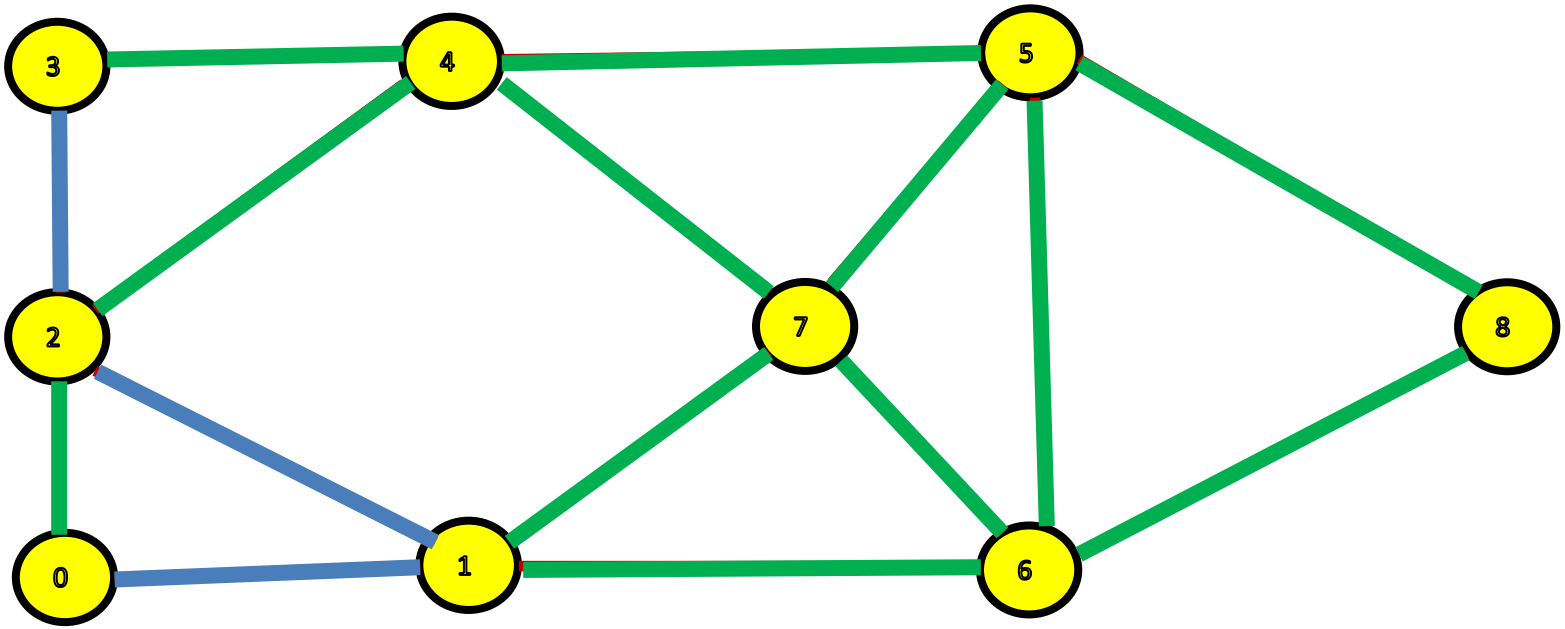
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



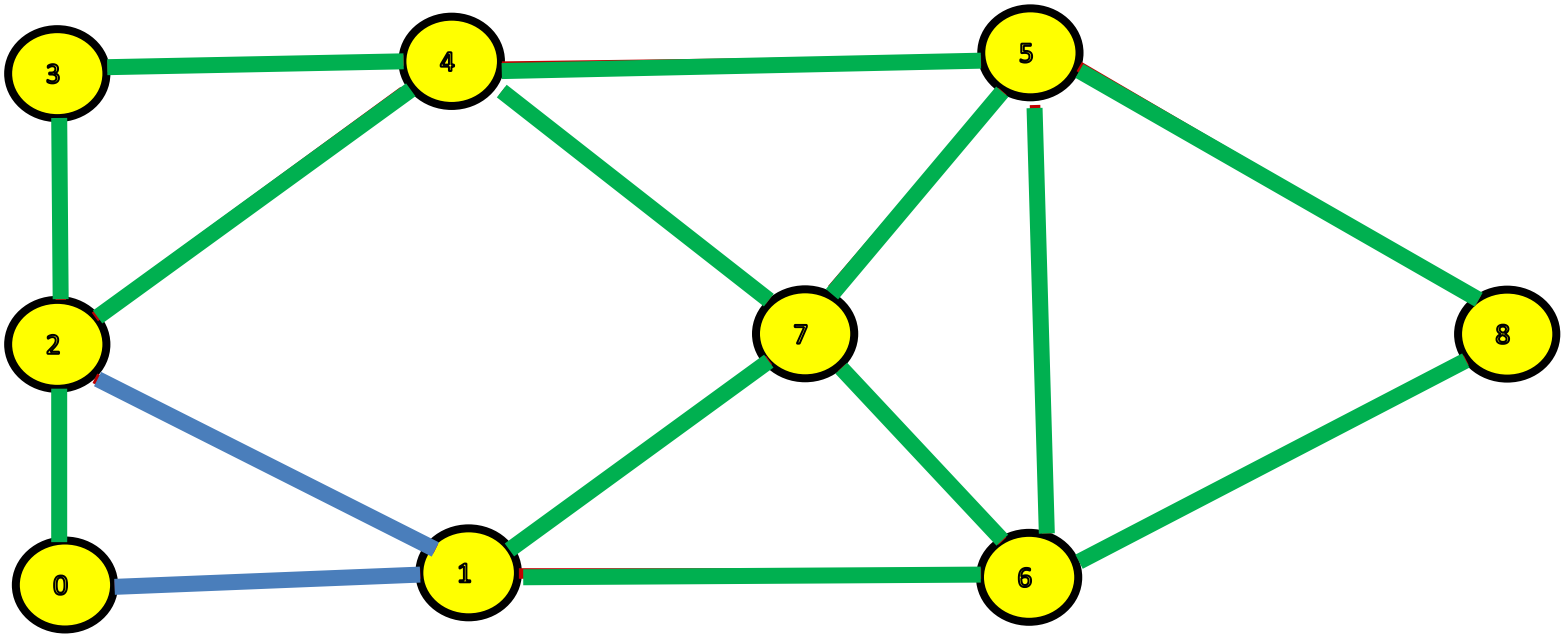
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



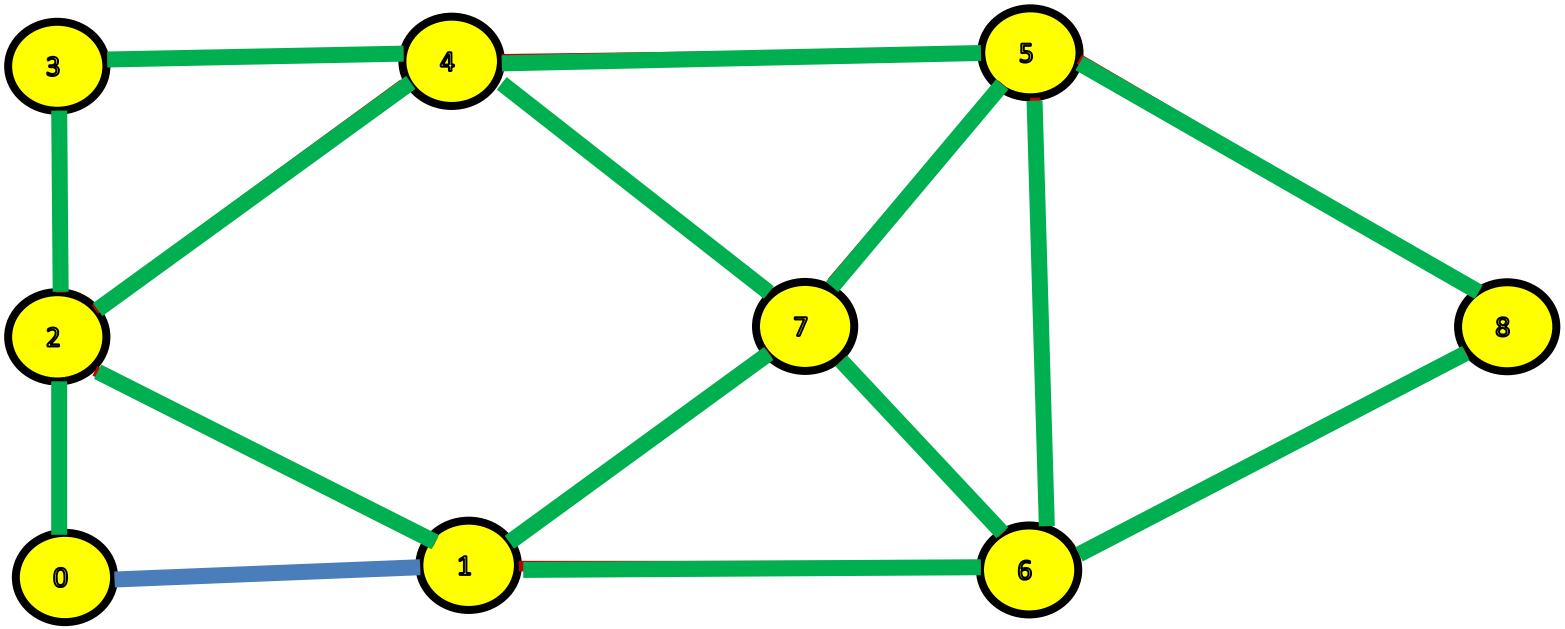
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



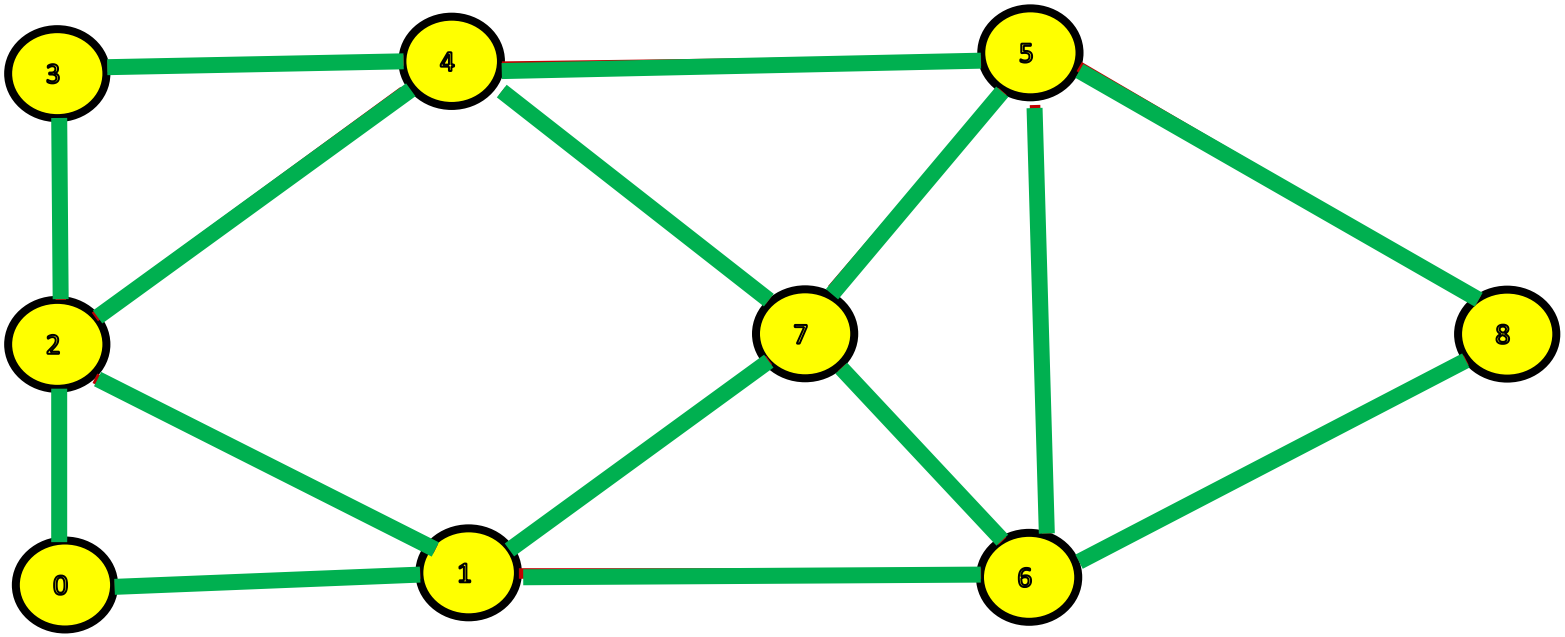
ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]



ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]

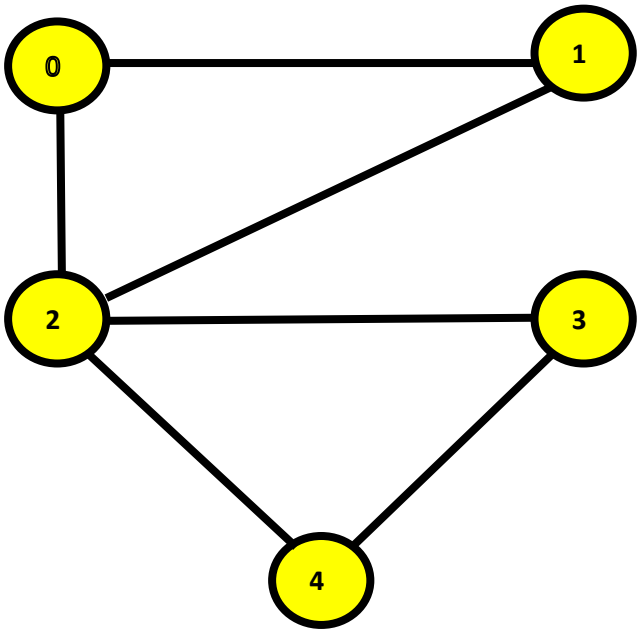


ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]

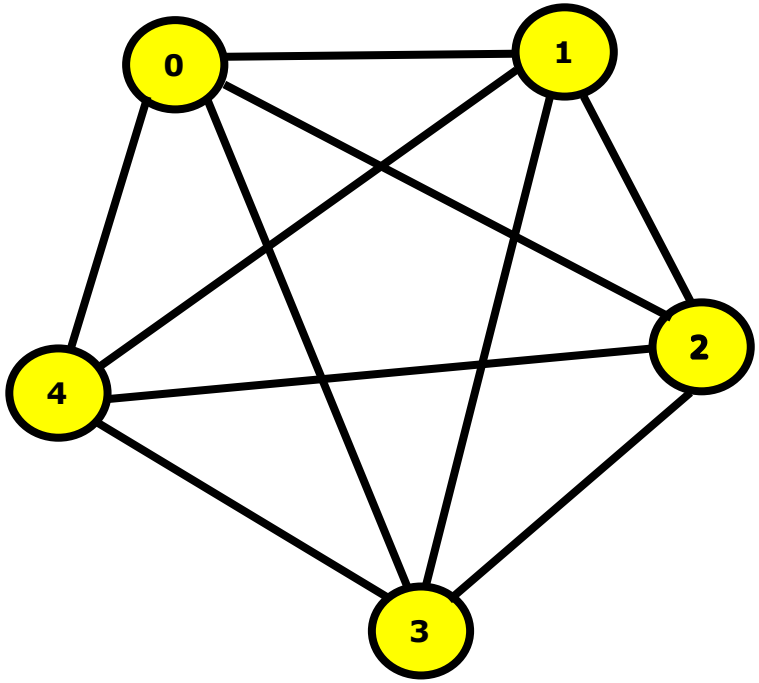


ec [0, 2, 4, 7, 6, 8 ,5, 7, 1, 6, 5, 4, 3 , 2, 1, 0]

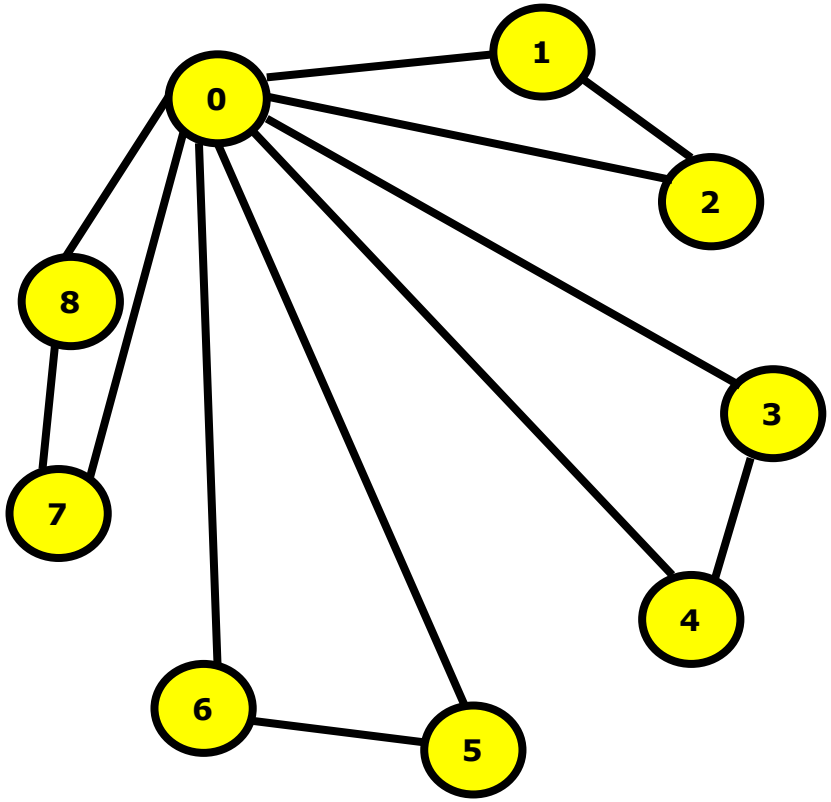
Example 1.



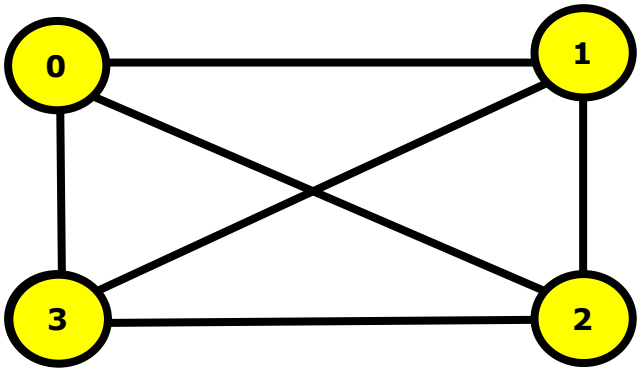
Example 2.



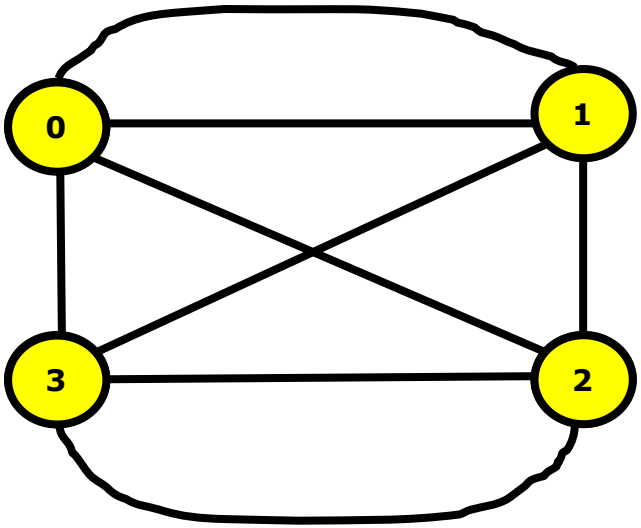
Example 3.



Example 4.



Example 5.



Stack class :

1. **push**

   public E push(E item) – pushes an item onto the top of this stack.
   This has exactly the same effect as : **addElement(item)**.

2. **pop**

   public E pop() – removes the object at the top of this stack and return the object.

3. **peek**

   public E peek() - looks at the object at the top of this stack without removing it from the stack.

4. **add**

   public add(E item) - pushes an item onto the top of this stack.