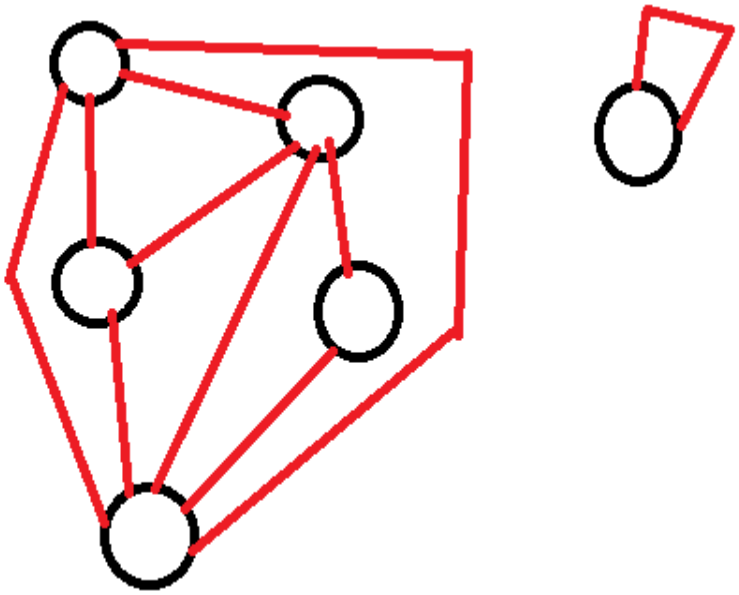
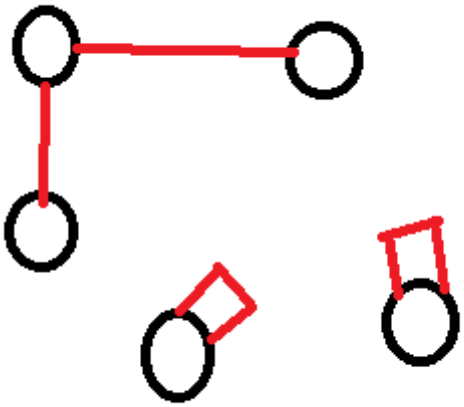
- Rita en oriktad graf med 6 hörn, 10 kanter och 2 sammanhängande komponenter. Går det att rita en graf med 5 hörn, 4 kanter och 3 komponenter?

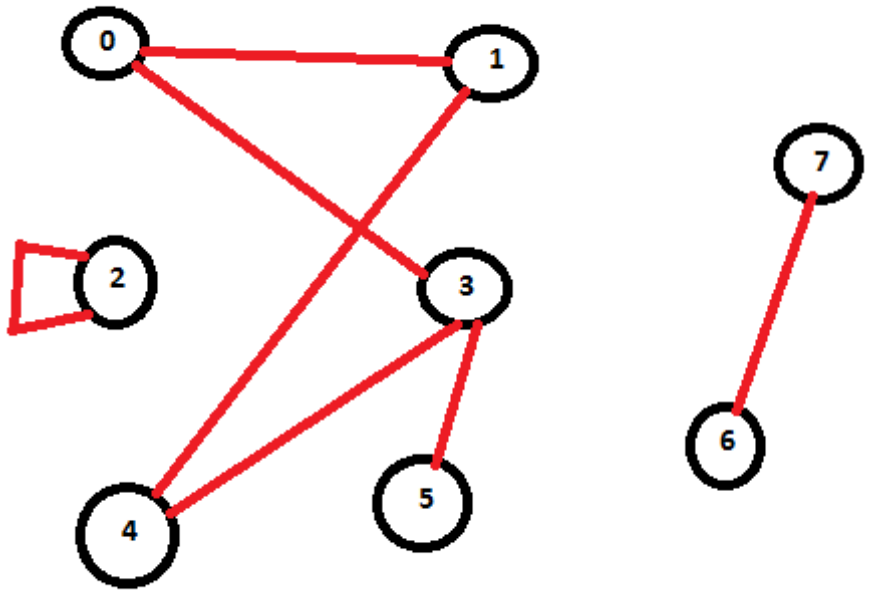


.



- Låt G vara en oriktad graf som består av 8 hörn numrerade från 0 till 7 och kantmängden {(0,1), (0,3), (1,4), (2,2), (3,4), (3,5), (6,7)}.
    - Rita G.
    - 2. Ange ordningen som hörnen besöks vid en djupetförstsökning (DFS) med start i hörn 0.
    - 3. Ange ordningen som hörnen besöks vid en breddenförstsökning (BFS) med start i hörn 0.

.



2. Ordningen: Lägger på 0 på stacken, markerar 0 som visited. Går till 1. Lägger 1 överst i stacken, markerar 1 som visited. Går till 4, lägger 4 överst i stacken, markerar 4 som visited. Går till 3, lägger 3 överst i stacken, markerar 3 som visited. Går till 5, lägger 5 överst i stacken och markerar 5 som visitedd. Sedan kommer den att droppa saker från stacken eftersom att samtliga grannar redan är märkta som visited. Till slut kommer stacken bli tom och sökningen slutar.

Alltså: 0->1->4->3->5.

3. Ordningen: För BFS gäller samma princip, men nu kommer samtliga grannar att besökas istället för ett. Den använder sig även av en kö och inte en stack. Går till 0, grannarna 1 och 3 sätts i kön, 0 stryks från kön och märks som visited. 1 som nu är först i kön märks och stryks. 4 som är granne till 1 ställs i kön efter 3 och 1 stryks. 3 som nu är först i kön besöks och stryks. Grannarna till 3, dvs 4 och 5 ställs i kön och 3 stryks från kön. 4 är först i kön och besöks samt stryks. Alla 4's grannar har blivit besökta så därför ställs inget nytt i kön. 5 som nu är först (och sist) i kön besöks och stryks. Även 5 har inga 'lediga grannar kvar'.

Alltså: 0->1->3->4->5

Du kan anta att grannarna till ett hörn besöks i nummerordning.

- Skulle du representera en graf med hjälp av en närhetsmatris eller med hjälp av närhetslistor i följande fall? Motivera dina svar.
    - 1. Grafen har 1000 hörn och 2000 kanter och det är viktigt att vara sparsam med minnet.
    - 2. Grafen har 1000 hörn och 50000 kanter och det är viktigt att vara sparsam med minnet.
    - Det är viktigt att snabbt (på konstant tid) kunna avgöra om två hörn är grannar. Om möjligt vill du också vara sparsam med minnet.
- 3. Förklara varför DFS tar $\Theta(n^2)$ tid för en sammanhängande graf med n hörn om grafen representeras med en närhetsmatris.

1. Med en närhetslista sker de flesta operationerna snabbare än om man jämför med en närhetsmatris. Detta är pga en närliggande lista är effektivare när graferna är glesare – dvs inte så många kanter (och grannar), då blir listorna små och effektiva att utföra operationer på.

2. Då skulle jag även där använda mig av en närhetslista, då har DFS O(n+c) i tidskomplexitet, där c är antal kanter. Vilket skulle resultera i ~50100. En närhetsmatris har tidskomplexiteten O(n^2) vilket skulle bli mer.

3. DFS har tildskomplexiteten O(n^2) för en närhetsmatris. Om en graf med n hörn representeras i en närhetsmatris består matrisen av totalt n^2 punkter. Alla måste då gås igenom, vilket ger O(n^2) tid.

Prova hur snabbt ditt program blir för olika värden på n, n <= 5000, när man använder närhetsmatris respektive närhetslistor.

- 1. Ange antalet komponenter och den största komponentents storlek när n = 1000.
- 2. Vilken datastruktur är bäst i det här fallet? Varför? Förklara genom att beräkna tidskomplexiteten för DFS med närhetsmatris samt för DFS med närhetslistor.

Antal element (n)	Närhetslista (nanosec)	Närhetsmatris (nanosec)
50	1536396	1162561
100	1999051	1736538
500	2961471	5844779
1000	3739537	20035916
2500	5435416	28632943
5000	8474654	106792157

1. Antal komponenter = 490, Största komponent = 25.

2. DFS har tidskomplexiteten O(n^2) för närhetsmatriser. DFS för närhetslista har tidskomplexiteten O(n + c) där c är antalet kanter och n är antalet hörn. Därför är närhetslistor bättre i detta fall, vilket visas i tabellen ovan.