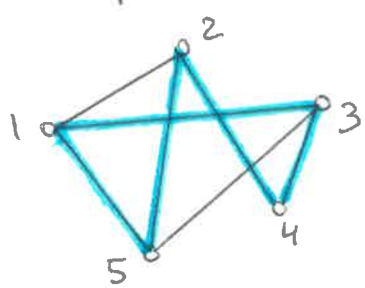
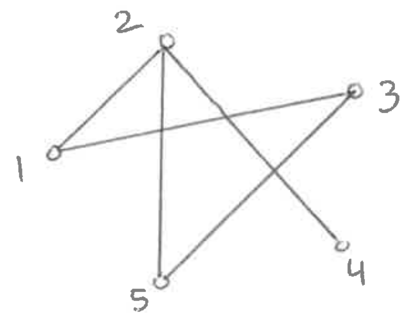
The Hamiltonian-Cycle (HC) Problem

Problem definition : given a graph $G=(V,E)$ undirected, find whether G has a HC (a cycle that contains each vertex exactly once).

example



G has a HC
$HC = (1,3,4,2,5)$



G does not have a HC

- HC problem is NP-complete
- we can represent the solution using an array

	1	2	3	4	5
X	1	3	4	2	5

therefore the solution has the form $X[1], X[2], X[3], \ldots, X[n]$

- we can assume without loss of generality that $X[1]=1$
- the algorithm returns
  - true if G has a HC; stop as soon as the algorithm finds a HC
  - false if G has no HC
- assume that the graph G is represented using the adjacency-matrix "adj"
- the algorithm follows the general framework for a backtracking algorithm and uses the functions:

  - hamilton (adj, x)
  - rhamilton (adj, k, x)
  - path_ok (adj, k, x)

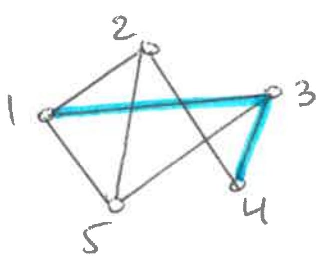
rhamilton (adj, k, x)

- select the k^{th} vertex in the HC
- assume that x[1], x[2], ..., x[k-1] forms a feasible partial solution

path_OK(adj, k, x)

- assume that x[1], x[2], .., x[k-1] is a feasible partial solution and that x[k] has been assigned some value
- return
  - true if x[1], x[2], .., x[k] is a feasible partial solution
  - false otherwise

- How to determine whether x[1], x[2], .., x[k] is a feasible partial solution?



check if
- x[k] is different than x[1], x[2], ..., x[k-1]
- if $k<n$, check if (x[k-1], x[k]) forms an edge
- if $k=n$, check if (x[n-1], x[n]) and (x[n], x[1]) are edges

k=4

	1	2	3	4	5
x	1	3	4	?	

- How can we determine which vertices are already used?

array used [1..n]

used [v]
- true if v has been already included in the path
- false otherwise

example

k=4

	1	2	3	4	5
x	1	3	4	?	

	1	2	3	4	5
used	T	F	T	T	F

input param — output param

```
hamilton(adj, x)
  n = adj.last    // n is the number of vertices
  x[1] = 1
  used[1] = true
  for i = 2 to n
      used[i] = false
  rhamilton(adj, 2, x)
```

```
rhamilton(adj, k, x)
  n = adj.last    // n is the number of vertices
  for x[k] = 2 to n
      if path_OK(adj, k, x) == true
          used[x[k]] = true
          if k == n
              print solution x[1], x[2], ..., x[n]
              return true
          else // k < n
              if rhamilton(adj, k+1, x) == true
                  return true
          used[x[k]] = false
  return false
```

x[1], ..., x[k], ..., x[n]

```
path_OK(adj, k, x)
  n = adj.last    // n is the number of vertices
  if used[x[k]] == true
      return false
  if k < n
      return adj[x[k-1], x[k]]
  else // k = n
      return adj[x[n-1], x[n]] && adj[x[n], x[1]]
```

RT analysis

- How many times is rhamilton(adj, k, x) called?

	$k=1$	0 times
	$k=2$	1 time
$1 \times ?$	$k=3$	$n-1$ times
$1 \times \times ?$	$k=4$	$\leq (n-1)(n-2)$ times
	$\dots$	
	$k=n$	$\leq (n-1)(n-2)\cdot\ldots\cdot 2$ times

- rhamilton() takes another $\Theta(n)$ besides the recursive calls

$$RT \leq n\cdot\left(1+(n-1)+(n-1)(n-2)+\ldots+(n-1)(n-2)\cdots 2\right) =$$

$$= n\cdot(n-1)!\left(\frac{1}{(n-1)!}+\frac{n-1}{(n-1)!}+\frac{(n-1)(n-2)}{(n-1)!}+\ldots+\frac{(n-1)(n-2)\cdots 2}{(n-1)!}\right) =$$

$$= n\cdot(n-1)!\underbrace{\left(\frac{1}{(n-1)!}+\frac{1}{(n-2)!}+\frac{1}{(n-3)!}+\ldots+\frac{1}{1!}\right)}_{e-1}$$

$$\boxed{\sum_{i=0}^{\infty}\frac{1}{i!}=e}$$

$e = 2.718$

$$\frac{1}{1}+\underbrace{\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\ldots}=e$$

$$RT \leq n\cdot(n-1)!\cdot(e-1)$$

$$RT \leq (e-1)\cdot n!$$

$$\boxed{RT = O(n!)}$$