Chapter 8 Graph Theory 图论

Lu Han

hl@bupt.edu.cn

8.2 Paths and Cycles 路径和回路

Definition 8.2.1 Let v_0 and v_n be vertices in a graph. A **path (路径)** from v_0 to v_n of length n is an alternating sequence of $n + 1$ vertices and n edges beginning with vertex v_0 and ending with vertex v_n,

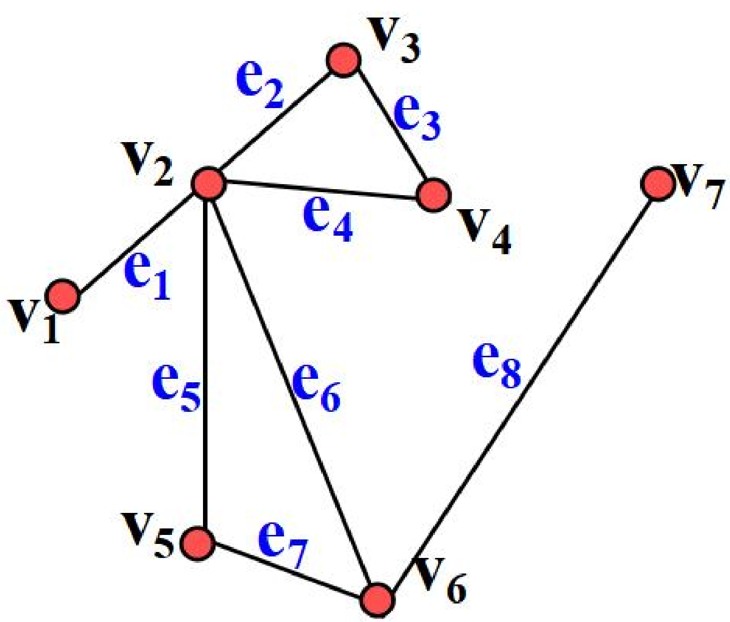
$$(v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n),$$

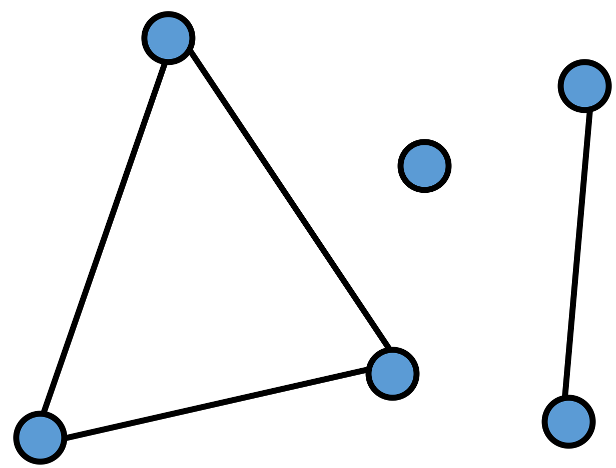
in which edge e_i is incident on vertices v_{i-1} and v_i for $i = 1, \ldots, n$.

8.2 Paths and Cycles 路径和回路

connected graph (连通图)

Definition 8.2.4 A graph G is **connected (连通的)** if given any vertices v and w in G, there is a path from v to w.





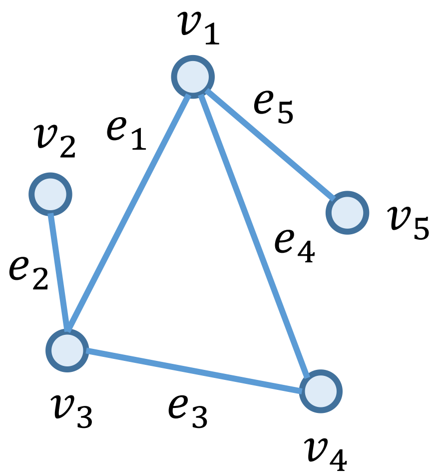
8.2 Paths and Cycles 路径和回路

Definition 8.2.8 Let $G = (V, E)$ be a graph, we call (V', E') a **subgraph (子图)** G if

(a) $V' \subseteq V$ and $E' \subseteq E$.

(b) For every edge $e' \in E'$, if e' is incident on v' and w', then $v', w' \in V'$.

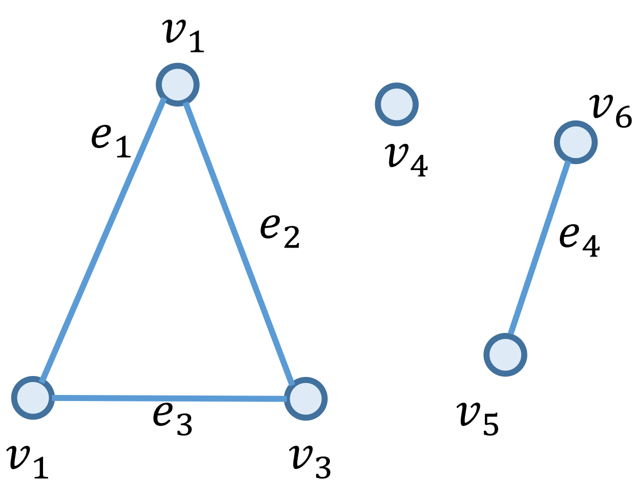
Example



8.2 Paths and Cycles 路径和回路

Definition 8.2.11 Let G be a graph and let v be a vertex in G. The subgraph G' of G consisting of all edges and vertices in G that are contained in some path beginning at v is called the **component (分支)** of G containing v.

Example 8.2.13 The components of G

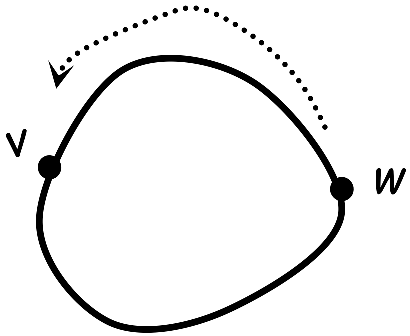


A graph is **connected** if and only if it has only **1 connected component**.

8.2 Paths and Cycles 路径和回路

Definition 8.2.14 Let v and w be vertices in a graph G.

- A **simple path（简单路径）** from v to w is a path from v to w with no repeated vertices.
- A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.
- A **simple cycle（简单回路）** is a cycle from v to v in which, except for the beginning and ending vertices that are both equal to v, there are no repeated vertices.



cycle: $v \cdots w \cdots v$

In a simple cycle, every vertex is of degree exactly 2.

8.2 Paths and Cycles 路径和回路

Eulerian Path (欧拉路径)

Euler Cycle (欧拉回路)： a cycle in a graph G that includes all of the edges and all of the vertices of G.

The **degree of a vertex (顶点度)** v, denoted by $\delta(v)$, is the number of edges incident on v.

Each loop on v contributes 2 to the degree of v.

8.2 Paths and Cycles 路径和回路

A graph G is an **Euler graph (欧拉图)** if it has an Euler cycle.

Theorems 8.2.17 and 8.2.18: G is an Euler graph if and only if G is connected and all its vertices have even degree.

Theorem 8.2.17 If a graph G has an Euler cycle, then G is connected and every vertex has even degree.

Theorem 8.2.18 If G is a connected graph and every vertex has even degree, then G has an Euler cycle.

8.2 Paths and Cycles 路径和回路

Theorem 8.2.21 The handshaking theorem

If G is a graph with m edges and n vertices $v_1, v_2, \ldots, v_n$, then

$$\sum_{i=1}^{n} \delta(v_i) = 2m$$

In particular, the sum of the degrees of all the vertices of a graph is even.

Proof When we sum over the degrees of all the vertices, we count each edge (*vi, vj*) twice—once when we count it as (*vi, vj*) in the degree of *vi* and again when we count it as *(vj, vi)* in the degree of *vj*. The conclusion follows.

If several people shake hands, the total number of hands shake must be even.

8.2 Paths and Cycles 路径和回路

Corollary 8.2.22 In any graph, the number of vertices of odd degree is even.

Proof Let us divide the vertices into two groups: those with even degree $x_1, \ldots, x_m$ and those with odd degree $y_1, \ldots, y_n$.
Let $S = \delta(x_1) + \delta(x_2) + \ldots + \delta(x_m)$, $T = \delta(y_1) + \delta(y_2) + \ldots + \delta(y_n)$.
By **Theorem 8.2.21 (The handshaking theorem)**, $S + T$ is even. Since S is the sum of even numbers, S is even. Thus T is even. But T is the sum of n odd numbers, and therefore n is even.

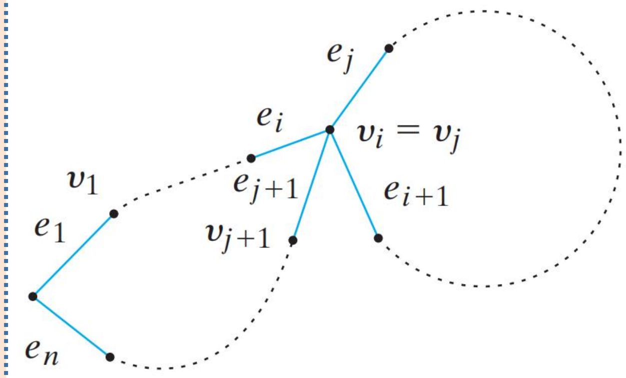
8.2 Paths and Cycles 路径和回路

Theorem 8.2.23 A graph has a path with no repeated edges from v to w ($v \neq w$) containing all the edges and vertices if and only if it is connected and v and w are the only vertices having odd degree.

8.2 Paths and Cycles 路径和回路

Theorem 8.2.24 If a graph G contains a cycle from v to v, G contains a simple cycle from v to v.

Proof Let
$C = (v_0, e_1, v_1, \ldots, e_i, v_i, e_{i+1}, \ldots, e_j, v_j, e_{j+1}, v_{j+1}, \ldots, e_n, v_n)$
be a cycle from v to v where $v = v_0 = v_n$. If C is not a simple cycle, then $v_i = v_j$, for some $i < j < n$. We can replace C by the cycle
$C' = (v_0, e_1, v_1, \ldots, e_i, v_i, e_{j+1}, v_{j+1}, \ldots, e_n, v_n)$
If C' is not a simple cycle from v to v, we repeat the previous procedure. Eventually we obtain a simple cycle from v to v.

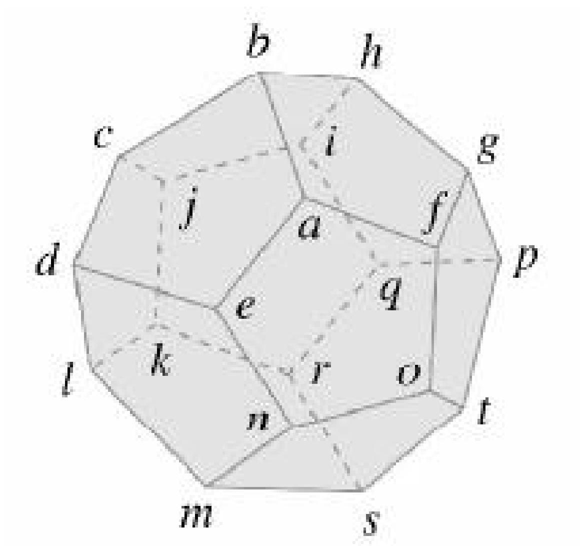


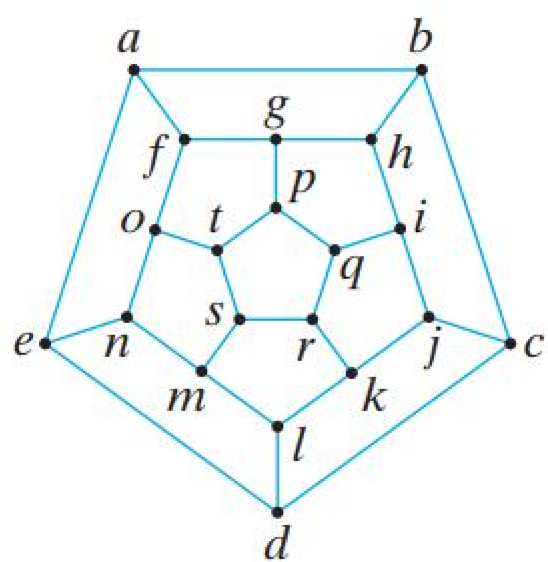
A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.

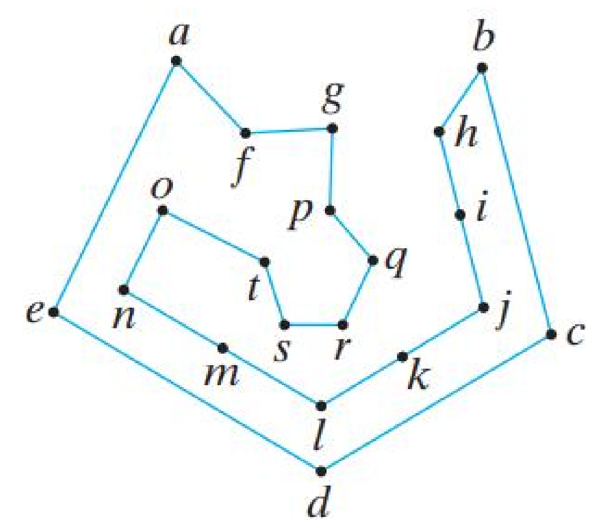
8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Sir William Rowan Hamilton marketed a puzzle in the mid-1800s in the form of a dodecahedron.

Hamilton’s puzzle (哈密顿难题): Can we find a cycle in the graph of the dedecahedron that contains each vertex exactly once？







8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

(Unlike the situation for Euler cycles, no easily verified necessary and sufficient conditions are known for the existence of a Hamiltonian cycle in a graph.)

Theorems 8.2.17 and 8.2.18: G is an Euler graph if and only if G is connected and all its vertices have even degree.

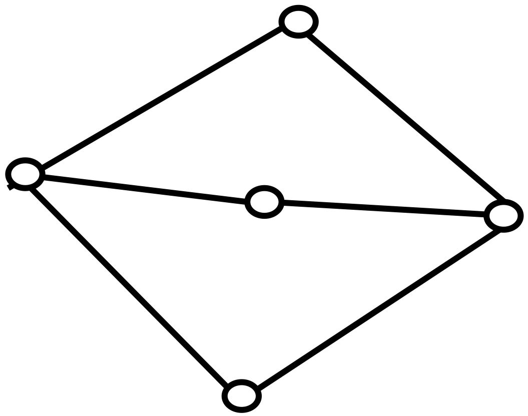
- A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.
- **Euler Cycle (欧拉回路):** a cycle in a graph G that includes all of the edges and all of the vertices of G.

8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

(Unlike the situation for Euler cycles, no easily verified necessary and sufficient conditions are known for the existence of a Hamiltonian cycle in a graph.)

Example 8.3.2 Does the following graph contain a Hamiltonian cycle?

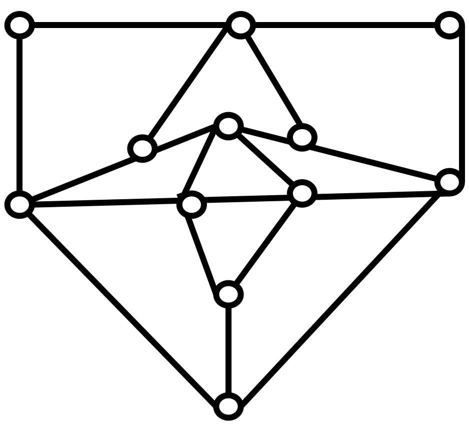


8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

(Unlike the situation for Euler cycles, no easily verified necessary and sufficient conditions are known for the existence of a Hamiltonian cycle in a graph.)

Example 8.3.3 Does the following graph contain a Hamiltonian cycle?

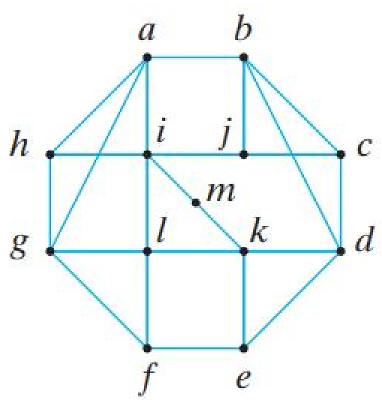


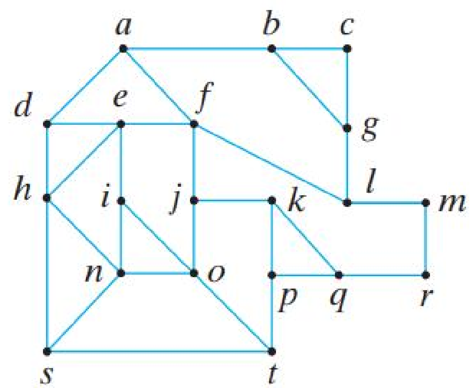
8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

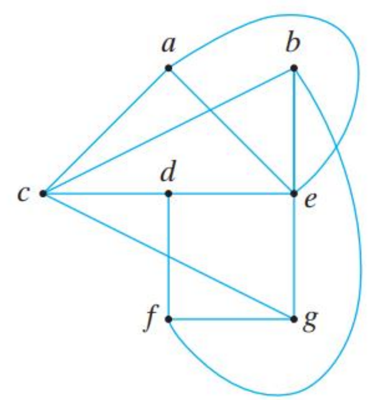
Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

(Unlike the situation for Euler cycles, no easily verified necessary and sufficient conditions are known for the existence of a Hamiltonian cycle in a graph.)

Exercise Does the following graphs contain a Hamiltonian cycle?







8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Euler Cycle (欧拉回路)： a cycle in a graph G that includes all of the edges and all of the vertices of G.

Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

1. Given an example of a graph that has an Euler cycle and a Hamiltonian cycle.

2. Given an example of a graph that has an Euler cycle but not a Hamiltonian cycle.

3. Given an example of a graph that has a Hamiltonian cycle but not an Euler cycle.

4. Given an example of a graph that has neither an Euler cycle nor a Hamiltonian cycle.

Randomized Hamiltonian Cycle 随机哈密顿回路算法

This algorithm takes as an input **a simple graph $\boldsymbol{G = (V, E)}$** and **searches for a Hamiltonian cycle**.

- If the algorithm returns true, it has found a Hamiltonian cycle.
- If the algorithm returns false, there is no Hamiltonian cycle (in fact, there is a vertex of degree 1).
- If the algorithm does not terminate, there may or may not be a Hamiltonian cycle.

If v is a vertex, $N(v)$ is the set of vertices adjacent to v.

Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

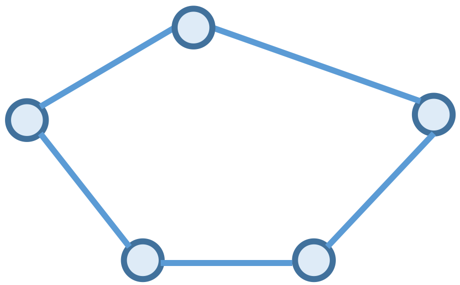
```
randomized_hamiltonian_cycle (G, n) {
  if (n == 1 ∨ n == 2) // trivial cases
    return false
  v_1 = random vertex in G
  i = 1
  while (i ¬= n ∨ v_1 ∉ N(v_i)) {
    N = N(v_i) − {v_1, . . . , v_{i−1}} // if i is 1, {v_1, . . . , v_{i−1}} is ∅
    // N contains the vertices adjacent to v_i (the current last vertex
    // of the path) that are not already on the path
    if (N ≠ ∅) {
      i = i + 1
      v_i = random vertex in N
    }
    else if (v_j ∈ N(v_i) for some j, 1 ≤ j < i − 1)
      (v_1, . . . , v_i) = (v_1, . . . , v_j, v_i, . . . , v_{j+1})
    else
      return false
  }
  return true
}
```

Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

Find a cycle $(v_1, v_2, \ldots, v_n, v_1)$ in which for any $i \neq j$, $v_i \neq v_j$

- Arbitrarily choose a vertex as v_1.

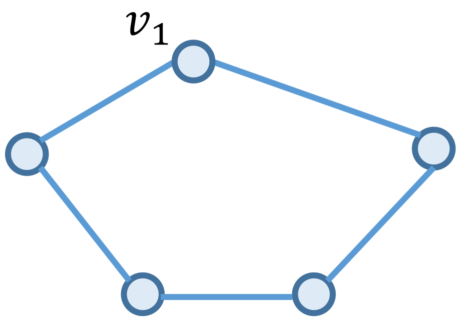


Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

Find a cycle $(v_1, v_2, \ldots, v_n, v_1)$ in which for any $i \neq j$, $v_i \neq v_j$

- Arbitrarily choose a vertex as v_1.

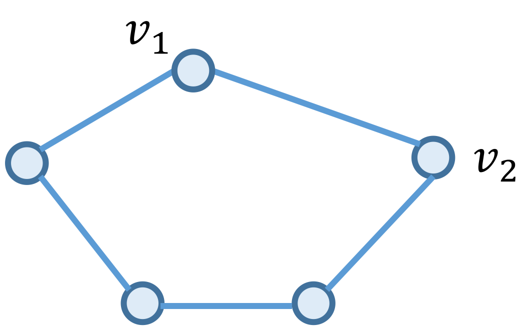


Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

Find a cycle $(v_1, v_2, \ldots, v_n, v_1)$ in which for any $i \neq j, v_i \neq v_j$

- Arbitrarily choose a vertex as v_1.
- Select a random adjancent vertex of v_1 as v_2.
  Now the path is (v_1, v_2).

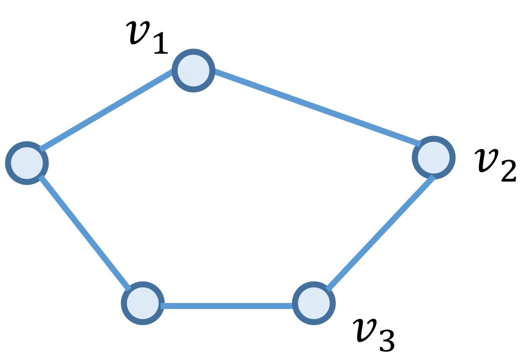


Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

Find a cycle $(v_1, v_2, \ldots, v_n, v_1)$ in which for any $i \neq j, v_i \neq v_j$

- Arbitrarily choose a vertex as v_1.
- Select a random adjancent vertex of v_1 as v_2.
  Now the path is (v_1, v_2).
- Select a random vertex in $N(v_2) - \{v_1, v_2\}$ as v_3.
  Now the path becomes (v_1, v_2, v_3).



Randomized Hamiltonian Cycle 随机哈密顿回路算法

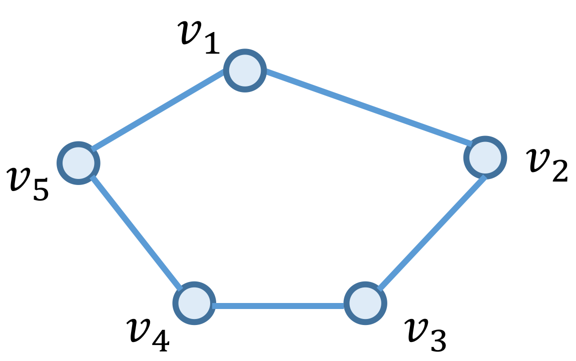
Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

Find a cycle $(v_1, v_2, \ldots, v_n, v_1)$ in which for any $i \neq j, v_i \neq v_j$

- Arbitrarily choose a vertex as v_1.
- Select a random adjancent vertex of v_1 as v_2.
  Now the path is (v_1, v_2).
- Select a random vertex in $N(v_2) - \{v_1, v_2\}$ as v_3.
  Now the path becomes (v_1, v_2, v_3).

$\vdots$

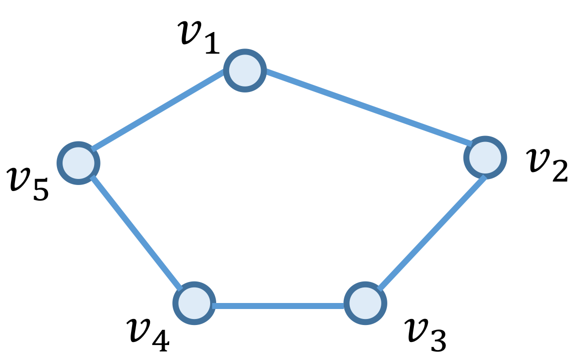
- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle.
  Returns true.



Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

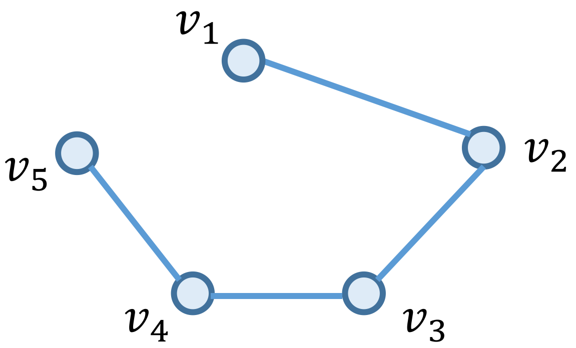
- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle.
  If $v_1 \notin N(v_n)$, what does the graph G look like?

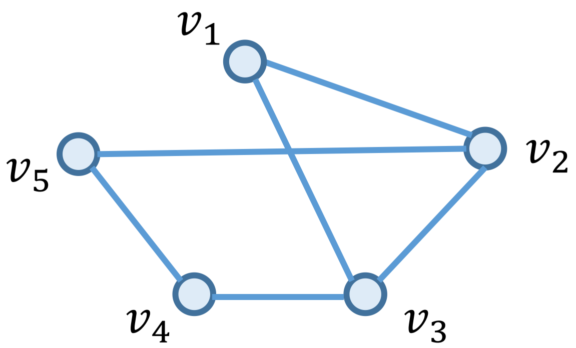


Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle.
  If $v_1 \notin N(v_n)$, what does the graph G look like?

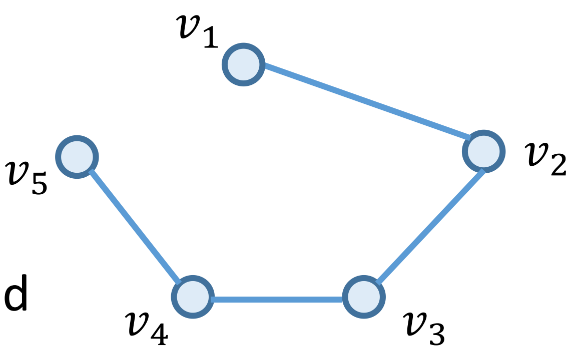


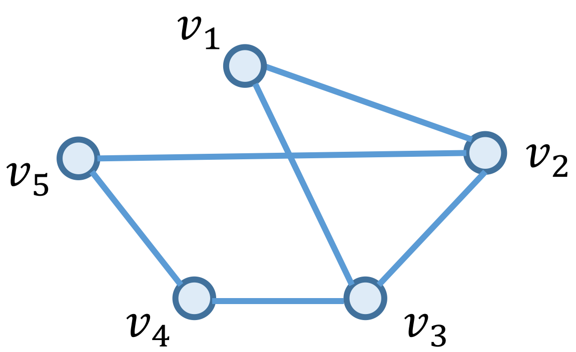


Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle.
  If $v_1 \notin N(v_n)$, what does the graph G look like?
  (i) If $\delta(v_n) = 1$, we have found a vertex of degree 1 and then returns false.

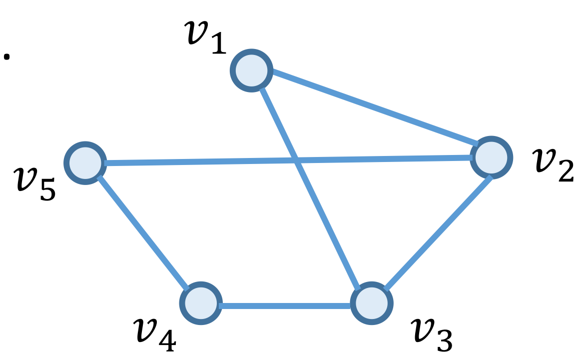




Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- The path becomes $(v_1, v_2, v_3, \dots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle.
  If $v_1 \notin N(v_n)$, what does the graph *G* look like?
  (i) If $\delta(v_n) = 1$, we have found a vertex of degree 1 and then returns false.
  (ii) If $\delta(v_n) > 1$, randomly select one of the vertices adjacent to v_n different from v_{n-1}, which we call v_j.



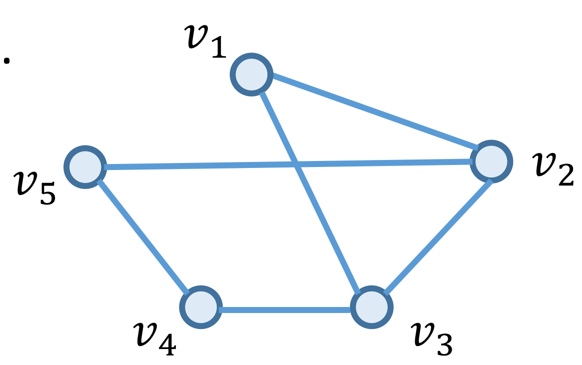
Update the path $(v_1, v_2, \dots, v_j, v_{j+1}, \dots, v_{n-1}, v_n)$ to
$(v_1, v_2, \dots, v_j, v_n, v_{n-1}, \dots, v_{j+1})$.

Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle.
  If $v_1 \notin N(v_n)$, what does the graph G look like?
  (i) If $\delta(v_n) = 1$, we have found a vertex of degree 1 and then returns false.
  (ii) If $\delta(v_n) > 1$, randomly select one of the vertices adjacent to v_n different from v_{n-1}, which we call v_j.

Update the path $(v_1, v_2, \ldots, v_j, v_{j+1}, \ldots, v_{n-1}, v_n)$ to $(v_1, v_2, \ldots, v_j, v_n, v_{n-1}, \ldots, v_{j+1})$.



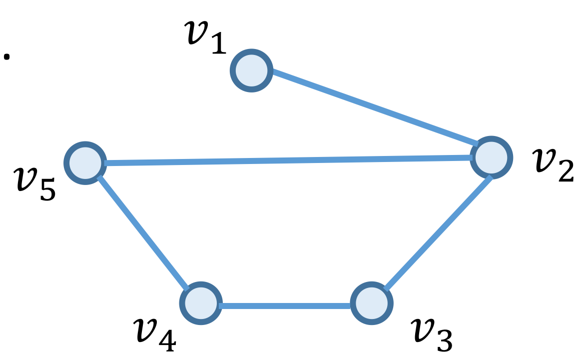
$(v_1, v_2, v_3, v_4, v_5)$ becomes $(v_1, v_2, v_5, v_4, v_3)$

Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle.
  If $v_1 \notin N(v_n)$, what does the graph G look like?
  (i) If $\delta(v_n) = 1$, we have found a vertex of degree 1 and then returns false.
  (ii) If $\delta(v_n) > 1$, randomly select one of the vertices adjacent to v_n different from v_{n-1}, which we call v_j.

Update the path $(v_1, v_2, \ldots, v_j, v_{j+1}, \ldots, v_{n-1}, v_n)$ to $(v_1, v_2, \ldots, v_j, v_n, v_{n-1}, \ldots, v_{j+1})$.



$(v_1, v_2, v_3, v_4, v_5)$ becomes $(v_1, v_2, v_5, v_4, v_3)$

Randomized Hamiltonian Cycle 随机哈密顿回路算法

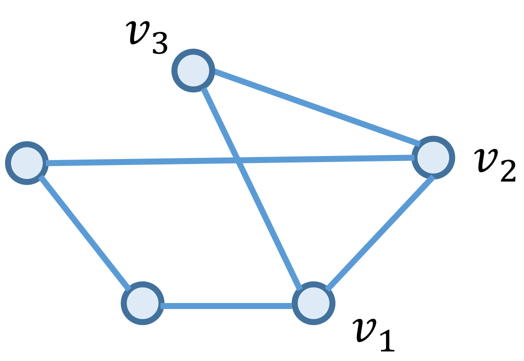
Input: A simple graph $G = (V, E)$ with n vertices.

Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- Arbitrarily choose a vertex as v_1.
- Select a random adjancent vertex of v_1 as v_2.
- Select a random vertex in $N(v_2) - \{v_1, v_2\}$ as v_3.

  ⋮
- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle. Returns true.
  If $v_1 \notin N(v_n)$, …

The path is $(v_1, v_2, v_3, \ldots, v_i)$ where $i \neq n$.



Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.

Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- Arbitrarily choose a vertex as v_1.
- Select a random adjancent vertex of v_1 as v_2.
- Select a random vertex in $N(v_2) - \{v_1, v_2\}$ as v_3.

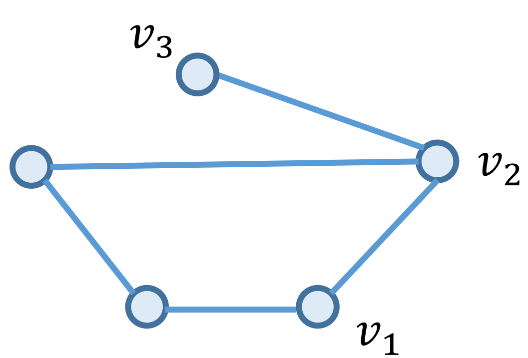
  $\vdots$

- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle. Returns true.
  If $v_1 \notin N(v_n)$, …

The path is $(v_1, v_2, v_3, \ldots, v_i)$ where $i \neq n$.

(i) If $\delta(v_i) = 1$, we have found a vertex of degree 1 and then returns false.

(ii) If $\delta(v_i) > 1$, randomly select one of the vertices adjacent to v_i different from v_{i-1}, which we call v_j.
Update the path $(v_1, v_2, \ldots, v_j, v_{j+1}, \ldots, v_{i-1}, v_i)$ to
$(v_1, v_2, \ldots, v_j, v_i, v_{i-1}, \ldots, v_{j+1})$.



Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.

Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- Arbitrarily choose a vertex as v_1.
- Select a random adjancent vertex of v_1 as v_2.
- Select a random vertex in $N(v_2) - \{v_1, v_2\}$ as v_3.

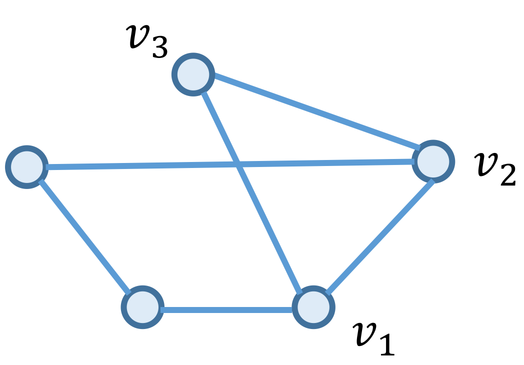
  ⋮
- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle. Returns true.
  If $v_1 \notin N(v_n)$, …

The path is $(v_1, v_2, v_3, \ldots, v_i)$ where $i \neq n$.

(i) If $\delta(v_i) = 1$, we have found a vertex of degree 1 and then returns false.

(ii) If $\delta(v_i) > 1$, randomly select one of the vertices adjacent to v_i different from v_{i-1}, which we call v_j. Update the path $(v_1, v_2, \ldots, v_j, v_{j+1}, \ldots, v_{i-1}, v_i)$ to

$$(v_1, v_2, \ldots, v_j, v_i, v_{i-1}, \ldots, v_{j+1}).$$



Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

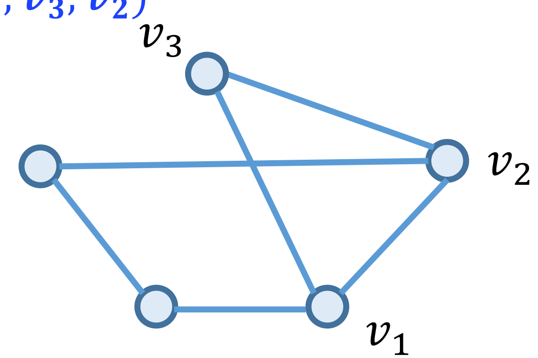
- Arbitrarily choose a vertex as v_1.
- Select a random adjancent vertex of v_1 as v_2.
- Select a random vertex in $N(v_2) - \{v_1, v_2\}$ as v_3.

  ⋮
- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle. Returns true.
  If $v_1 \notin N(v_n)$, …

The path is $(v_1, v_2, v_3, \ldots, v_i)$ where $i \neq n$.
(i) If $\delta(v_i) = 1$, we have found a vertex of degree 1 and then returns false.
(ii) If $\delta(v_i) > 1$, randomly select one of the vertices adjacent to v_i different from v_{i-1}, which we call v_j.
Update the path $(v_1, v_2, \ldots, v_j, v_{j+1}, \ldots, v_{i-1}, v_i)$ to

$$(v_1, v_2, \ldots, v_j, v_i, v_{i-1}, \ldots, v_{j+1}).$$

(v_1, v_2, v_3) becomes (v_1, v_3, v_2)



Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.

Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- Arbitrarily choose a vertex as v_1.
- Select a random adjancent vertex of v_1 as v_2.
- Select a random vertex in $N(v_2) - \{v_1, v_2\}$ as v_3.

  $\vdots$

- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle. Returns true.
  If $v_1 \notin N(v_n)$, …

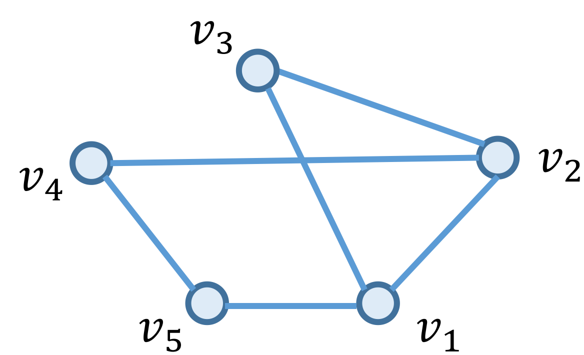
The path is $(v_1, v_2, v_3, \ldots, v_i)$ where $i \neq n$.

(i) If $\delta(v_i) = 1$, we have found a vertex of degree 1 and then returns false.

(ii) If $\delta(v_i) > 1$, randomly select one of the vertices adjacent to v_i different from v_{i-1}, which we call v_j.
Update the path $(v_1, v_2, \ldots, v_j, v_{j+1}, \ldots, v_{i-1}, v_i)$ to

$$(v_1, v_2, \ldots, v_j, v_i, v_{i-1}, \ldots, v_{j+1}).$$

(v_1, v_2, v_3) **becomes** (v_1, v_3, v_2)



$(v_1, v_3, v_2, v_4, v_5)$

Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.

Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- Arbitrarily choose a vertex as v_1.
- Select a random adjancent vertex of v_1 as v_2.
- Select a random vertex in $N(v_2) - \{v_1, v_2\}$ as v_3.

  ⋮
- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle. Returns true.
  If $v_1 \notin N(v_n)$, …

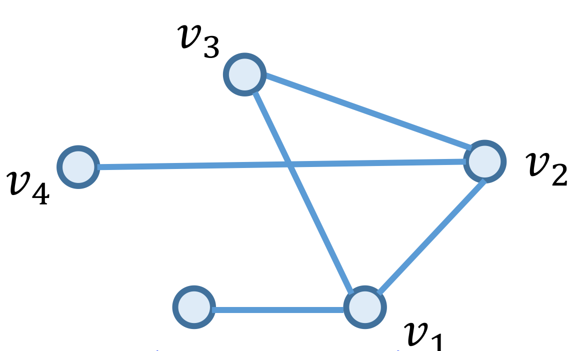
The path is $(v_1, v_2, v_3, \ldots, v_i)$ where $i \neq n$.

(i) If $\delta(v_i) = 1$, we have found a vertex of degree 1 and then returns false.

(ii) If $\delta(v_i) > 1$, randomly select one of the vertices adjacent to v_i different from v_{i-1}, which we call v_j.
Update the path $(v_1, v_2, \ldots, v_j, v_{j+1}, \ldots, v_{i-1}, v_i)$ to

$$(v_1, v_2, \ldots, v_j, v_i, v_{i-1}, \ldots, v_{j+1}).$$

(v_1, v_2, v_3) **becomes** (v_1, v_3, v_2)



(v_1, v_3, v_2, v_4)

Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.

Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

- Arbitrarily choose a vertex as v_1.
- Select a random adjancent vertex of v_1 as v_2.
- Select a random vertex in $N(v_2) - \{v_1, v_2\}$ as v_3.

  ⋮
- The path becomes $(v_1, v_2, v_3, \ldots, v_n)$.
  If $v_1 \in N(v_n)$, we have found a Hamiltonian cycle. Returns true.
  If $v_1 \notin N(v_n)$, …

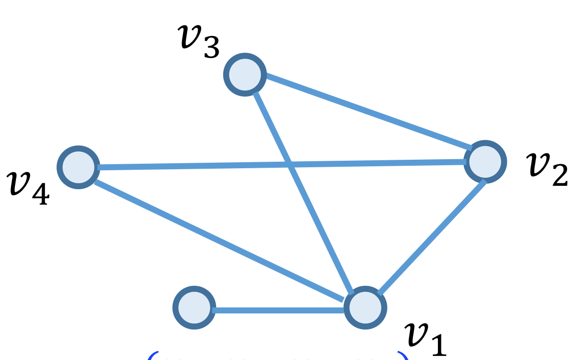
The path is $(v_1, v_2, v_3, \ldots, v_i)$ where $i \neq n$.

(i) If $\delta(v_i) = 1$, we have found a vertex of degree 1 and then returns false.

(ii) If $\delta(v_i) > 1$, randomly select one of the vertices adjacent to v_i different from v_{i-1}, which we call v_j.
Update the path $(v_1, v_2, \ldots, v_j, v_{j+1}, \ldots, v_{i-1}, v_i)$ to

$$(v_1, v_2, \ldots, v_j, v_i, v_{i-1}, \ldots, v_{j+1}).$$

(v_1, v_2, v_3) **becomes** (v_1, v_3, v_2)



(v_1, v_3, v_2, v_4)

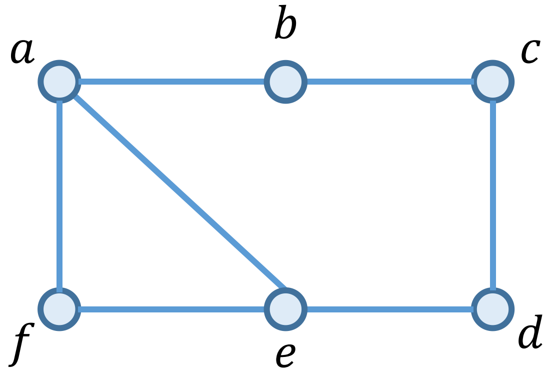
Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.
Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

```
randomized_hamiltonian_cycle (G, n) {
  if (n == 1 ∨ n == 2) // trivial cases
    return false
  v₁ = random vertex in G
  i = 1
  while (i ¬= n ∨ v₁ ∉ N(vᵢ)) {
    N = N(vᵢ) − {v₁, . . . , vᵢ₋₁} // if i is 1, {v₁, . . . , vᵢ₋₁} is ∅
    // N contains the vertices adjacent to vᵢ (the current last vertex
    // of the path) that are not already on the path
    if (N ≠ ∅) {
      i = i + 1
      vᵢ = random vertex in N
    }
    else if (vⱼ ∈ N(vᵢ) for some j, 1 ≤ j < i − 1)
      (v₁, . . . , vᵢ) = (v₁, . . . , vⱼ, vᵢ, . . . , vⱼ₊₁)
    else
      return false
  }
  return true
}
```

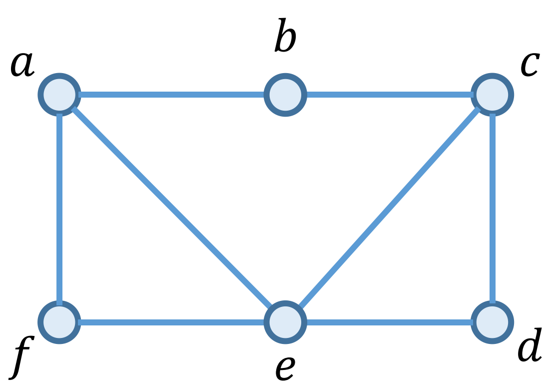
Randomized Hamiltonian Cycle 随机哈密顿回路算法

Show that in the following graph, the Randomized Hamiltonian Cycle Algorithm will terminate.



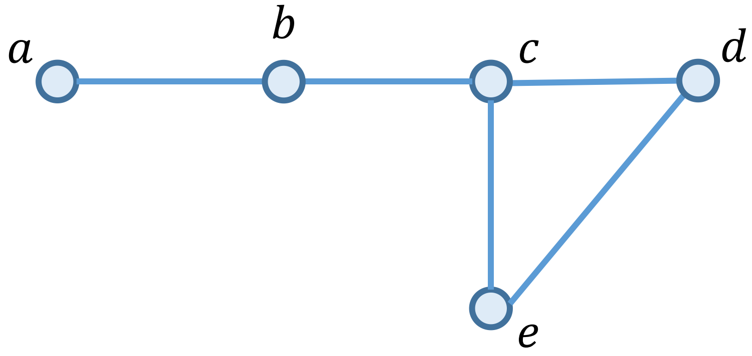
Randomized Hamiltonian Cycle 随机哈密顿回路算法

Show that in the following graph, it is possible for the Randomized Hamiltonian Cycle Algorithm to fail to find a Hamiltonian cycle even though there is one and, in this case, it does not terminate.



Randomized Hamiltonian Cycle 随机哈密顿回路算法

Show that in the following graph, it is possible that the Randomized Hamiltonian Cycle Algorithm may not find the vertex of degree 1, and, in this case, it will not terminate.



Randomized Hamiltonian Cycle 随机哈密顿回路算法

True or False?

If a graph with no Hamiltonian cycle and all vertices of degree at least 2 is input to the Randomized Hamiltonian Cycle Algorithm, the algorithm will not terminate.

Randomized Hamiltonian Cycle 随机哈密顿回路算法

True or False?

If a graph with no Hamiltonian cycle and all vertices of degree at least 2 is input to the Randomized Hamiltonian Cycle Algorithm, the algorithm will not terminate.

```
randomized_hamiltonian_cycle (G, n) {
    if (n == 1 ∨ n == 2) // trivial cases
        return false
    v₁ = random vertex in G
    i = 1
    while (i ¬= n ∨ v₁ ∉ N(vᵢ)) {
        N = N(vᵢ) − {v₁, . . . , vᵢ₋₁} // if i is 1, {v₁, . . . , vᵢ₋₁} is ∅
        // N contains the vertices adjacent to vᵢ (the current last vertex
        // of the path) that are not already on the path
        if (N ≠ ∅) {
            i = i + 1
            vᵢ = random vertex in N
        }
        else if (vⱼ ∈ N(vᵢ) for some j, 1 ≤ j < i − 1)
            (v₁, . . . , vᵢ) = (v₁, . . . , vⱼ, vᵢ, . . . , vⱼ₊₁)
        else
            return false
    }
    return true
}
```

Randomized Hamiltonian Cycle 随机哈密顿回路算法

Truth or False?

The algorithm always terminate when the input is K_n.

The algorithm always terminate when the input is $K_{m,n}$.

Randomized Hamiltonian Cycle 随机哈密顿回路算法

Suggest ways to improve the algorithm.