

CS5314

Randomized Algorithms

Lecture 16: Balls, Bins, Random Graphs
(Random Graphs, Hamiltonian Cycles)

Objectives

- Introduce **Random Graph Model**
 - used to define a probability space for all graphs with **n** vertices
- Introduce a **randomized** algorithm for finding **Hamiltonian cycle**

Random Graph Model

- Random Graph Model is used to define a probability space for generating:
undirected, simple (no loops, no multiple edges)
graphs with n (labeled) vertices
- There are two common models:
 1. The $G_{n,p}$ model
 2. The $G_{n,N}$ model

The $G_{n,p}$ model

- Let $M = n(n-1)/2$
 - M = maximum #edges in an undirected simple graph with n vertices
- The $G_{n,p}$ model generates a graph with n vertices randomly
- In particular, for a graph G with a particular set of m edges,

$$\Pr(G \text{ is generated}) = p^m(1-p)^{M-m}$$

The $G_{n,p}$ model

- One way to generate a random graph in $G_{n,p}$ is as follows:
 1. Start with an empty graph with n vertices
 2. Add each edge independently with probability p

Question: How many graphs in $G_{n,p}$ model?

The $G_{n,N}$ model

- The $G_{n,N}$ model generates a graph with n vertices, N edges uniformly at random
- One way to generate a random graph in $G_{n,N}$ is as follows:
 1. Start with an empty graph with n vertices
 2. Add N edges, each time choose an edge uniformly from remaining edges

Question: How many graphs in $G_{n,N}$ model?

$G_{n,p}$ versus $G_{n,N}$

- When $p = N/M = N/(n(n-1)/2)$
 - #edges in a graph in $G_{n,p} = \text{Bin}(M, p)$
 - Its expected value = N , and it is also concentrated around N
- Also, when #edges of a graph in $G_{n,p}$ is conditioned to be exactly N , it has the same (uniform) distribution as $G_{n,N}$
- similar to *exact case* and *Poisson case* in balls-and-bins model

$G_{n,p}$ versus $G_{n,N}$ (2)

In fact, there are further similarities ...

E.g., Throwing edges into the graph in $G_{n,N}$ is like throwing balls into bins: Each vertex is a bin, each time we throw two balls

E.g., in the coupon collector's problem, we found that when we throw $n \ln n + cn$ balls, the probability of having no empty bins converges to $e^{-e^{-c}}$ as $n \rightarrow \infty$, and ...

$G_{n,p}$ versus $G_{n,N}$ (3)

... and similarly, we have the following:

Theorem: Let $N = (n \ln n + cn) / 2$

Then the probability that a random graph chosen in $G_{n,N}$ has no isolated vertices converges to $e^{-e^{-c}}$ as $n \rightarrow \infty$

Proof: Left as an exercise (Ex. 5.19)

[Idea: Relate this with the balls-and-bins model]

Why study random graphs?

- Many graph problems are **NP-hard**, which are believed to be very difficult
- E.g., Max Clique, Max Independent Set, Hamilton cycle, Min Vertex Cover, ...

Question: Are these problems difficult for most inputs? Or, only a small fraction of inputs are causing the difficulty?

Hamiltonian Cycles

Definition: A **Hamiltonian cycle** is a cycle that traverses each vertex exactly once

- In general, finding a Hamiltonian cycle is very difficult...
- However, we shall show that when input is suitably chosen, we have a randomized algorithm such that :
 - ➔ for most input, our algorithm can find a Hamiltonian cycle quickly **w.h.p.**

Hamiltonian Cycles

Precisely, we shall show that :

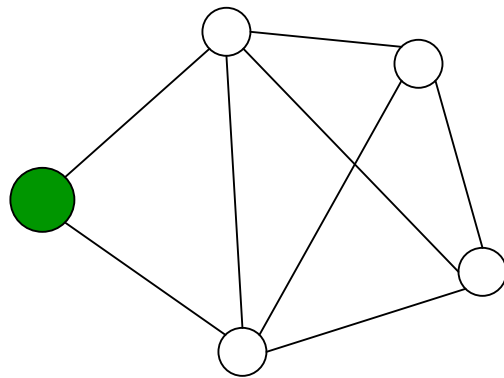
Theorem:

For sufficiently large n , we can design a randomized algorithm which can find a Hamiltonian cycle in a random graph from $G_{n,p}$ w.h.p., whenever $p \geq (40 \ln n)/n$

This immediately implies that when p is at least $(40 \ln n)/n$, a random graph from $G_{n,p}$ will contain a Hamilton cycle w.h.p.

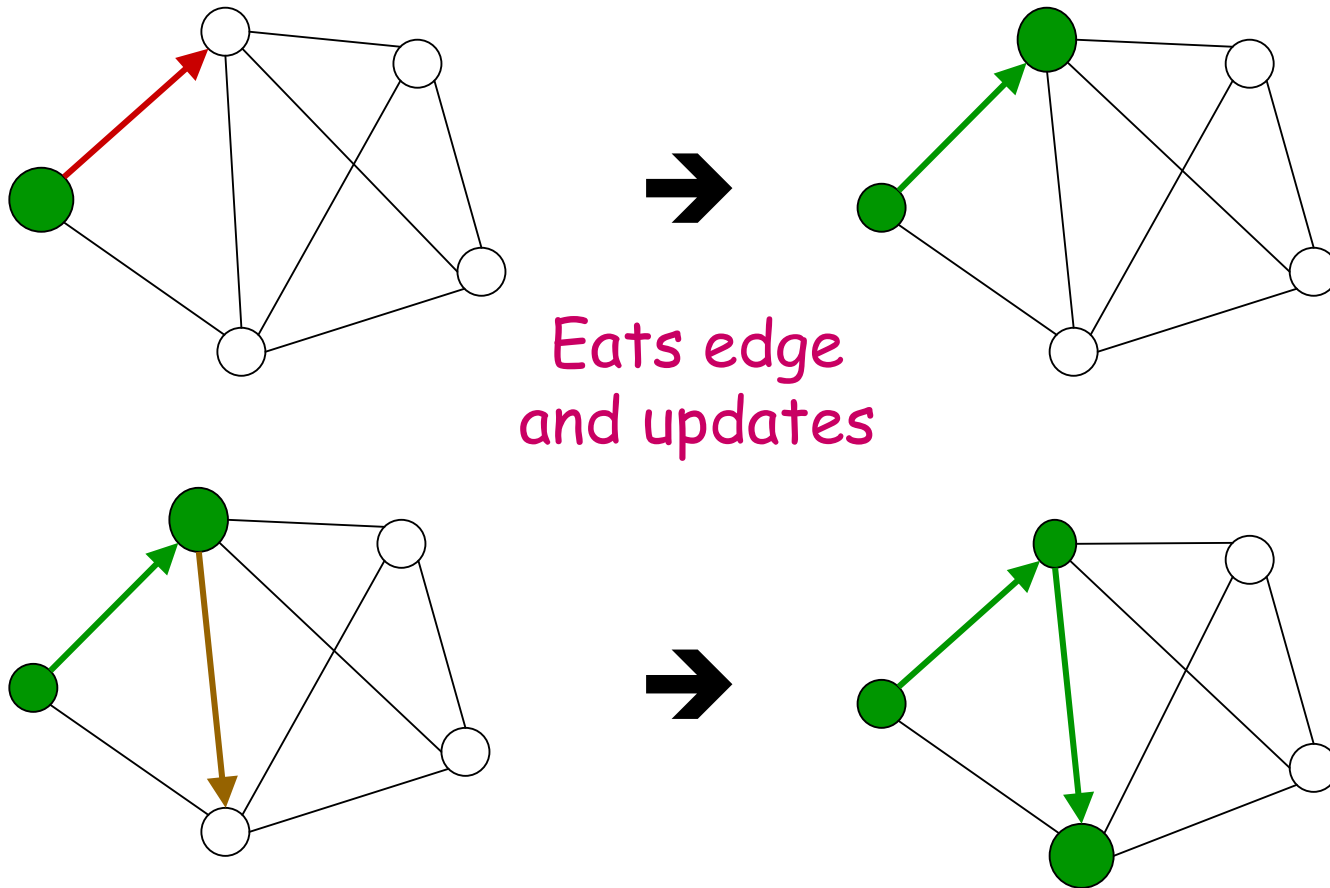
The Algorithm (version 1)

- The algorithm is like the game **snake/nibbles** (貪吃蛇) and make use of a simple operation called **rotation**
- We start by picking a random vertex in the graph as the **head** (of the snake)

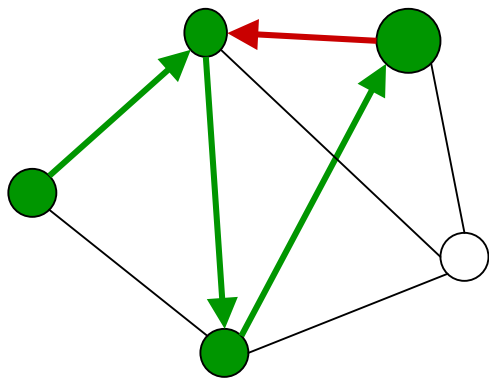
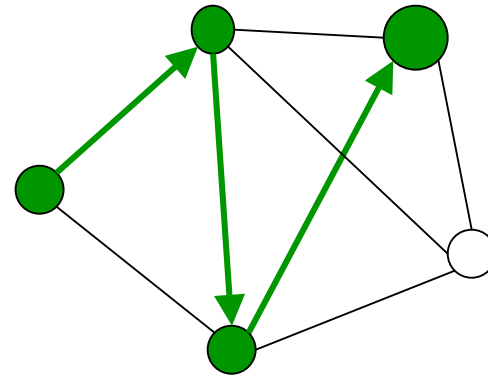
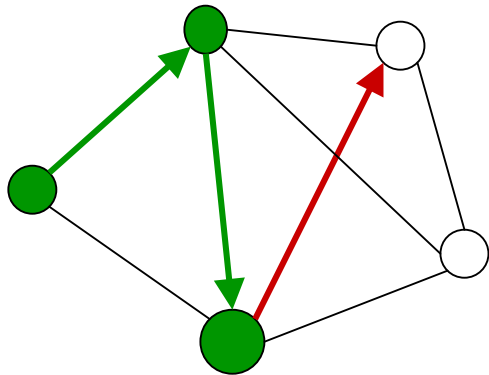


● head

Then, repeatedly, we choose an adjacent edge of the head. The snake "eats" this edge and updates the position of the head



The Algorithm (version 1)



What to do when the snake tries to eat itself?

The Algorithm (version 1)

Let $P = v_1, v_2, \dots, v_k$ be the current snake,

$v_k = \text{head of snake}$

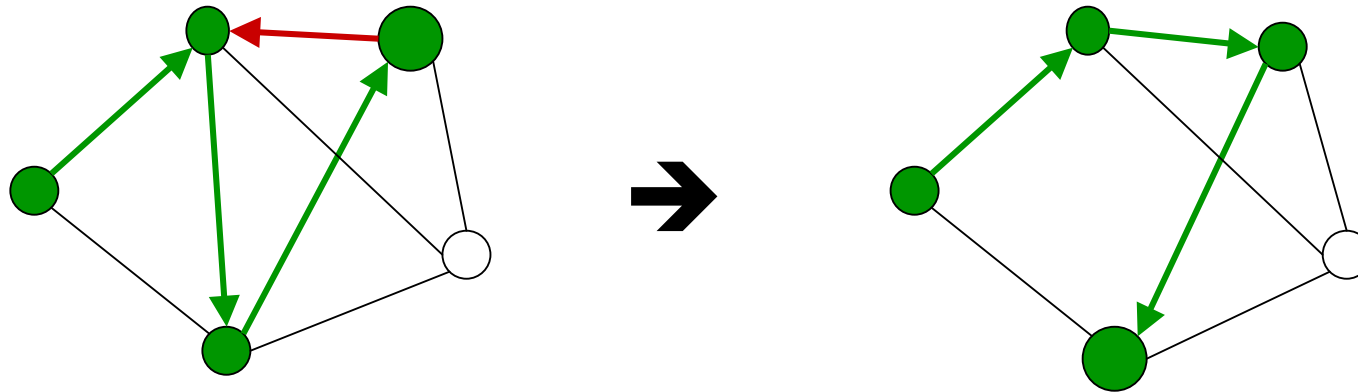
Suppose the snake now eats the edge (v_k, v_j)

That is, the snake eats itself

- If $v_j = \text{tail} = v_1$ and if all vertices are explored, we obtain an Hamiltonian cycle
- Otherwise, we change perform **rotation** to change the snake into:

$$P' = v_1, v_2, \dots, v_{j-1}, v_j, v_k, v_{k-1}, v_{k-2}, \dots, v_{j+2}, v_{j+1}$$

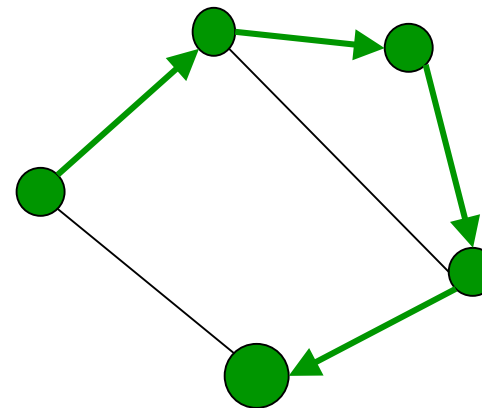
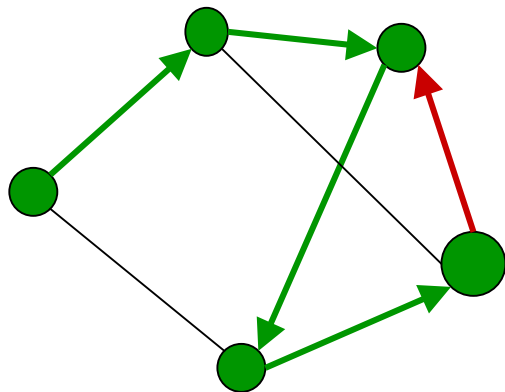
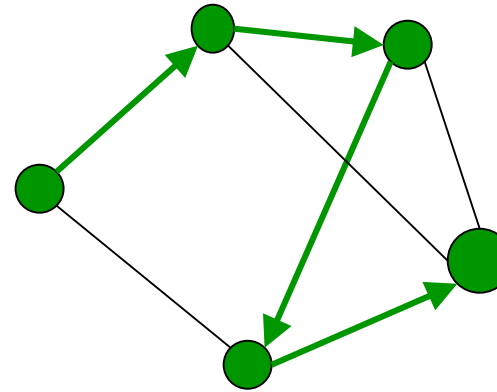
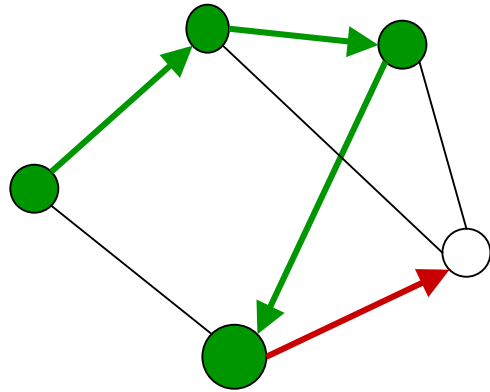
The Algorithm (version 1)



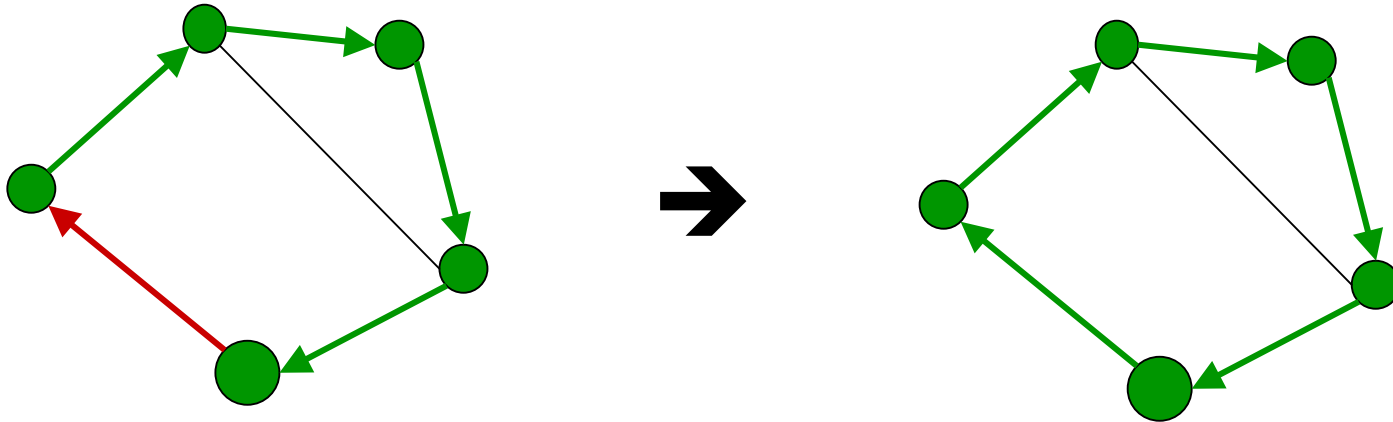
What to do when the
snake tries to eat itself?

perform rotation

The Algorithm (version 1)



The Algorithm (version 1)



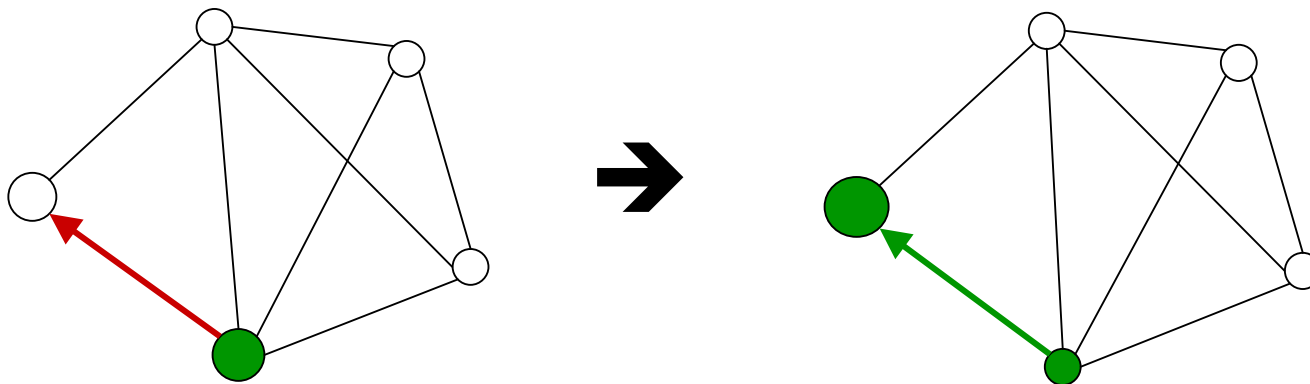
When the snake eats the tail
and all vertices are visited

We get Hamiltonian cycle

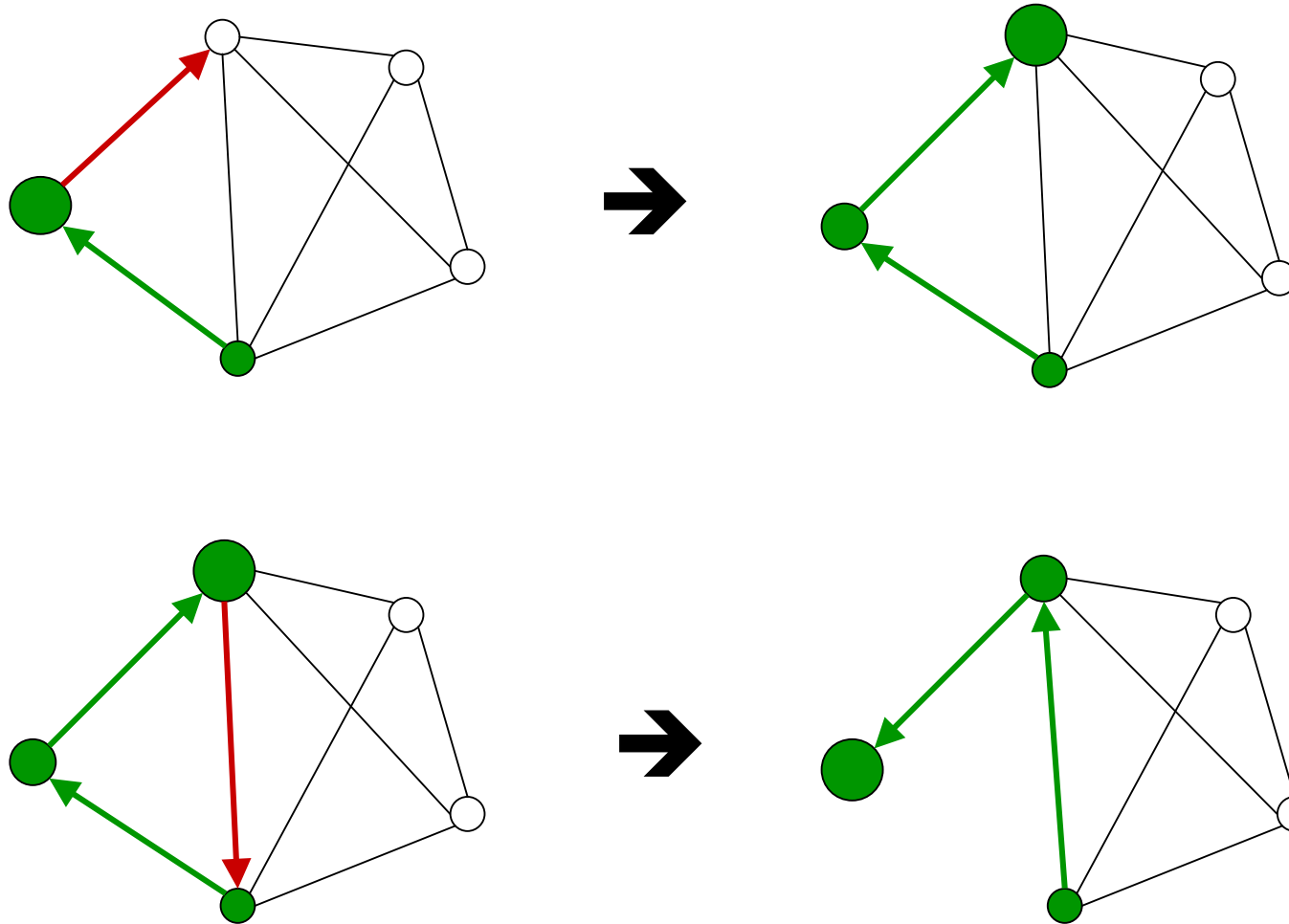
Failure Case of the Algorithm

- When the snake **does not have** any edge to eat next (before a Hamiltonian cycle is found), we stop and report **"FAIL"**.

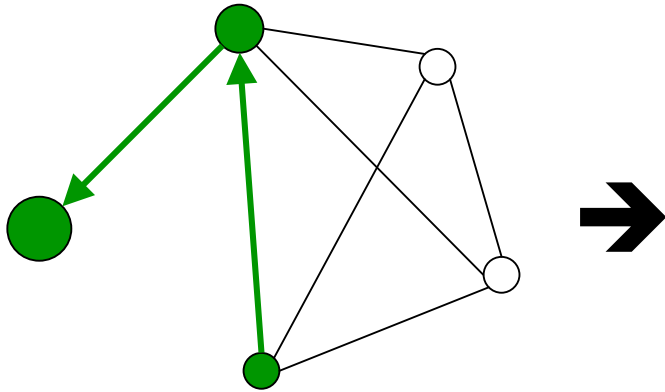
E.g., Let's start with another vertex:



The Algorithm (version 1)



The Algorithm (version 1)



When the snake has nothing
to eat, it dies

Report No Hamiltonian Cycle

The Algorithm (version 1)

The previous algorithm turns out to be very simple, and is likely to be efficient when there are many edges in the graph

Question: When will the algorithm fail?

... difficult to analyze directly ...

Next, we will modify the algorithm to make it a bit "stupid", but then we can analyze its performance more easily ...

The Algorithm (version 1)

Now, let us define three **subroutines**:

Suppose $P = v_1, v_2, \dots, v_k$ is current path,
 $v_k = \text{head}$

IsHamilton(P, v) {

1. If $v = v_1$ and all vertices are visited,
return TRUE
2. Else, return FALSE

}

The Algorithm (version 1)

Rotate(P, j) {

1. Update head to v_{j+1}

2. Update $P = v_1, v_2, \dots, v_j, v_k, v_{k-1}, \dots, v_{j+2}, v_{j+1}$

}

Extend(P, v) {

1. Update the head to v

2. Update $P = v_1, v_2, \dots, v_k, v$

}

Then, the current algorithm becomes:

1. Choose a random node as head
2. Repeat until no unused edge is adjacent to head
 - 2.1 Let $P = v_1, v_2, \dots, v_k$, where $v_k = \text{head}$
 - 2.2 Choose an unused edge adjacent to v_k , say (v_k, v) , uniformly at random.
Mark this edge as used
 - (i) If $\text{IsHamilton}(P, v)$, DONE
 - (ii) Else, if v is unvisited, $\text{Extend}(P, v)$
 - (iii) Else, $v = v_j$ for some j . $\text{Rotate}(P, j)$

- In the modified algorithm, a used edge can be traversed again (with some prob):
 1. Choose a random node as head
 2. Repeat until all adjacent nodes of head were visited from head at least once
 - 2.1 Let $P = v_1, v_2, \dots, v_k$, where $v_k = \text{head}$
 - 2.2 Let $x = \# \text{nodes visited}$ from v_k before
 - 2.3 Perform randomly one of the following:
 - (a) prob = $1/n$:
Reverse P , make v_1 as new head

(b) prob = x/n :

Choose a random node v that is visited from v_k previously

(c) prob = $1 - 1/n - x/n$:

Choose a random node v adjacent to v_k , which was not visited from v_k previously.
Mark v as visited from v_k

2.4 For (b) and (c)

(i) If $\text{IsHamilton}(P, v)$, DONE

(ii) Else, if v is not on P , $\text{Extend}(P, v)$

(iii) Else, $v = v_j$ for some j . $\text{Rotate}(P, j)$

The Algorithm (version 2)

Some properties of the modified algorithm:
(a good test of our understanding...)

- If an edge (u,v) is used before, it may be used again
- A node not visited from the current **head** before may have been visited from some other node already
- If u is visited from v before, v may not be visited from u (i.e., visit has direction)

The Algorithm (version 2)

- The modified algorithm seems to work for a directed graph
- Let us define a **new** random graph model and check the performance of our algo :

For each vertex v ,

For each vertex $u \neq v$,

Set u to be its adjacent vertex with probability q , independently

Remark: in this model, if u is an adjacent vertex of v , then v may not be an adjacent vertex to u

Performance in New Model

Let G be a random graph from the **new** model

Suppose we run the modified algorithm on G
and it does not **FAIL** for first t steps

Let $V_t = \text{head}$ after t^{th} steps

Lemma: After the t^{th} step, if there is at least one adjacent vertex of V_t unvisited from V_t , then for any vertex u ,

$$\Pr(V_{t+1} = u \mid V_t = u_t, V_{t-1} = u_{t-1}, \dots, V_0 = u_0) = 1/n$$

Remark: The head seems to be chosen uniformly at random at each step, regardless the history of the process

Proof

Let $P = v_1, v_2, \dots, v_k$ be the current path
(note: no special relationship between k and t)


The vertex u is in one of four cases :

Case 1: $u = v_1$

Case 2: $u = v_{j+1}$, and v_k visits v_j before

Case 3: $u = v_{j+1}$, but v_k not visit v_j before

Case 4: u is not on P



Cases 1–3: u is on P

Case 1 and Case 2

Case 1 ($u = v_1$) :

- The only way that u becomes next head is when path is reversed
→ probability = $1/n$

Case 2 ($u = v_{j+1}$, and v_k visits v_j before before) :

u is next head when we choose to perform 2.3(b) and then v_j is picked

→ probability = $x/n * 1/x = 1/n$

Case 3

Case 3 ($u = v_{j+1}$, but v_k not visit v_j before) :

u is next head when we choose to perform 2.3(c) and v_j is picked

$$\rightarrow \text{prob} = (1 - 1/n - x/n)(1/(n-1-x)) = 1/n$$

Reason:

Each vertex was set to an adjacent vertex of v_k independently with the same probability q ;
By symmetry, each $n-1-x$ unvisited vertex (excluding head) is picked with same probability

Case 4

Case 4 (u is not on P):

u is next head if we choose to perform 2.3(c) and then u is picked

$$\rightarrow \text{prob} = (1 - 1/n - x/n)(1/(n-1-x)) = 1/n$$

Reason:

Again, by symmetry, each $n-1-x$ unvisited vertex (excluding head) is picked with same probability

Conclusion: Any vertex becomes next head with same probability $1/n$

Performance in New Model (2)

- From previous lemma, we see that finding **Hamiltonian path** (not cycle) in new model is exactly coupon collector's problem:

Finding a new vertex to add, when the path has $n-k$ vertex (i.e., k globally unvisited vertex), is k/n

- Once all vertices are on the path, we get a **Hamiltonian cycle** if we run 2.3(b) or 2.3(c), and the vertex chosen is v_1

Performance in New Model (3)

Hence,

if at least one adjacent vertex of head is unvisited from head before at each step

we can expect Hamiltonian path to be found in $O(n \log n)$ steps w.h.p.,

and then get a Hamiltonian cycle in $O(n \log n)$ steps w.h.p. (why?)

Next, we show that when $q \geq (20 \ln n) / n$, the if condition happens w.h.p. as well

Performance in New Model (4)

Lemma: Suppose the modified algorithm is run on a random graph from the **new** model, with $q \geq (20 \ln n) / n$

Then for sufficiently large n , the algorithm will successfully find a Hamiltonian cycle in $3n \ln n$ iterations of Step 2 **w.h.p.**

How to prove?

(Note: We do not assume input graph has a Hamilton cycle)

Proof

For the algorithm to **fail**, one of the following events must happen:

E_1 : It runs for $3n \ln n$ steps, so that at each step, there is adjacent unvisited vertex from **head**; Yet, the algorithm **does not** get a **Hamiltonian cycle**

E_2 : At least one vertex has got **no** adjacent unvisited vertex in the first $3n \ln n$ steps

In other words, $\Pr(\text{fail}) \leq \Pr(E_1) + \Pr(E_2)$

Proof: Bounding $\Pr(E_1)$

- Previously, we have shown that as long as there is an adjacent unvisited vertex from head, the next head is chosen uniformly at random in each step
- Probability that a particular vertex not on current path P after $2n \ln n$ steps is:
at most $(1 - 1/n)^{2n \ln n} \leq e^{-2 \ln n} = 1/n^2$
- $\Pr(\text{some vertex not on } P \text{ after } 2n \ln n \text{ steps})$
 $\leq 1/n$

Proof: Bounding $\Pr(E_1)$

- If all vertices are on P , the Hamiltonian path will become a cycle with probability $1/n$ in each step

→ the probability that after all vertices are on P , we cannot get a Hamiltonian cycle in $n \ln n$ steps is at most

$$(1 - 1/n)^{n \ln n} \leq e^{-\ln n} = 1/n$$

→ $\Pr(E_1) \leq 1/n + 1/n = 2/n$... (why?)

Proof: Bounding $\Pr(E_2)$

Now, we bound $\Pr(E_2)$, the probability that at least one vertex has **no** adjacent unvisited vertex in the first $3n \ln n$ steps

Consider the following events:

E_{2a} : There is a vertex v such that v has been a head at least $10 \ln n$ times
(so that its adjacent vertices are used up)

E_{2b} : There is a vertex which has fewer than $10 \ln n$ adjacent vertices initially

Proof: Bounding $\Pr(E_{2a})$

For E_2 to happen, E_{2a} or E_{2b} must happen

→ In other words, $\Pr(E_2) \leq \Pr(E_{2a}) + \Pr(E_{2b})$

For E_{2a} to happen

Let $X_j = \#$ times vertex j is head

Recall: for any vertex u ,

$$\Pr(u = \text{head in the next step}) = 1/n$$

$$\rightarrow X_j = \text{Bin}(3n \ln n, 1/n)$$

$$\mu = E[X_j] = 3 \ln n$$

Proof: Bounding $\Pr(E_{2a})$

$$\begin{aligned} \rightarrow \Pr(E_{2a}) &\leq \sum_{j=1 \text{ to } n} \Pr(X_j \geq 10 \ln n) \dots (\text{why?}) \\ &= n \Pr(X_1 \geq 10 \ln n) \\ &\leq n \Pr(X_1 \geq 9 \ln n) \\ &\leq n \Pr(X_1 \geq (1+2) \mu) \\ &\leq n (e^2 / (1+2)^{(1+2)})^\mu \dots (\text{why?}) \\ &= n (e^2/27)^{3 \ln n} \\ &\leq n (e^{-1})^{3 \ln n} = n (1/n^3) \\ &\leq 1/n \end{aligned}$$

Proof: Bounding $\Pr(E_{2b})$

- Next, we bound $\Pr(E_{2b})$, the probability that some vertex initially has fewer than $10 \ln n$ adjacent vertices
- Let $Y_j = \#$ adjacent vertices of j initially
 - $Y_j = \text{Bin}(n-1, q)$
 - $\mu = E[Y_j] = (n-1)q$
 - $\geq (n-1) 20 \ln n / n$
 - $\geq 19 \ln n$... for sufficiently large n

Proof: Bounding $\Pr(E_{2b})$

$$\begin{aligned} \rightarrow \Pr(E_{2b}) &\leq \sum_{j=1 \text{ to } n} \Pr(Y_j \leq 10 \ln n) \quad \dots \text{(why?)} \\ &= n \Pr(Y_1 \leq 10 \ln n) \\ &\leq n \Pr(Y_1 \leq (1 - 9/19) \mu) \\ &\leq n e^{-\mu (9/19)^2 / 2} \quad \dots \text{(why?)} \\ &\leq n e^{-81 \ln n / 38} \\ &\leq n e^{-2 \ln n} = n (1/n^2) \\ &= 1/n \end{aligned}$$

Proof: Bounding $\Pr(\text{fail})$

In other words,

$$\begin{aligned}\Pr(E_2) &\leq \Pr(E_{2a}) + \Pr(E_{2b}) \\ &\leq 1/n + 1/n = 2/n\end{aligned}$$

Combining with previous results, we have:

$$\begin{aligned}\Pr(\text{fail}) &\leq \Pr(E_1) + \Pr(E_2) \\ &\leq 2/n + 2/n = 4/n\end{aligned}$$

for sufficiently large n

Performance in $G_{n,p}$

Finally, we relate the new model with $G_{n,p}$:

Theorem: For sufficiently large n , by initializing the adjacent vertices appropriately, we can apply the modified algorithm and find a Hamiltonian cycle in a randomly chosen graph from $G_{n,p}$ w.h.p., when $p \geq 40 \ln n / n$

How to initialize appropriately?

Proof

- Instead of working with the random graph G chosen from $G_{n,p}$, we show how to **convert** G randomly to some directed graph G' and apply the modified algorithm
- Our conversion is carefully done so that we obtain the random directed graph G' as if we choose G' from the **new** model

Let q in $[0,1]$ be a value such that $p = 2q - q^2$

Proof

We use the following process:

For each edge (u,v) in G ,

- (1) With probability $q(1-q)/p$, set u to be v 's adjacent vertex but v is not u 's adjacent
- (2) With probability $q(1-q)/p$, set v to be u 's adjacent vertex but u is not v 's adjacent
- (3) With remaining probability (q^2/p) , set u to be v 's adjacent vertex AND v to be u 's adjacent vertex

Proof (2)

Now, for any vertex u , it is initially an adjacent vertex of v with probability:

$$p \left(\frac{q(1-q)}{p} + \frac{q^2}{p} \right) = q$$

Also, for any two vertices u and v , the probability that u is v 's adjacent vertex AND v is u 's adjacent vertex is

$$p \left(\frac{q^2}{p} \right) = q^2$$

Proof (3)

Thus,

$\Pr(u \text{ is } v\text{'s adjacent vertex AND } v \text{ is } u\text{'s adjacent vertex})$

$$= q^2$$

$$= \Pr(u \text{ is } v\text{'s adjacent vertex}) \times \Pr(v \text{ is } u\text{'s adjacent vertex})$$

→ In other words, exactly the **new** model !!!
(adjacent vertex are independently assigned with probability q)

Proof (4)

- Obviously, if a Hamiltonian cycle can be found in the directed graph G' , we can find a Hamiltonian cycle in the original G
- Since $p = 2q - q^2 \leq 2q$, for $p \geq 40 \ln n / n$, we have
$$q \geq p/2 \geq 20 \ln n / n$$
- By previous theorem, we can find a Hamiltonian cycle for G' (thus, G) w.h.p.