

# Random Graphs

A probabilistic model of graphs for probabilistic analysis of graph algorithms.

$G(n, p)$  random graph model: Given a  $n$ -vertex labeled graph, an (undirected) edge occurs between a pair of vertices with probability  $p$ , independently of other pairs.

The probability of a graph  $G$  of  $k$  edges is given by

$$p^k (1 - p)^{\binom{n}{2} - k}$$

The probability that a graph  $G \in G(n, p)$  contains  $k$  edges is binomially distributed:

$$\binom{\binom{n}{2}}{k} p^k (1 - p)^{\binom{n}{2} - k}$$

# Algorithm for finding a Hamiltonian Path

1. Choose an arbitrary vertex  $x_0$  to start the path.  
 $HEAD = x_0$ .
2. Repeat until all vertices are connected:
  - (a) Choose a random vertex (say  $u$ ) from HEAD's adjacency list.
  - (b) remove  $(HEAD, u)$  from HEAD's and  $u$ 's lists.
  - (c) **If**  $u$  is not in the path then  $HEAD = u$ ;  
**else** use the edge to ROTATE path and set HEAD to the successor of  $u$  in the path.

# Algorithm for finding a Hamiltonian Path

1. Choose an arbitrary vertex  $x_0$  to start the path.  
 $HEAD = x_0$ .
2. Repeat until all vertices are connected:
  - (a) Choose a random vertex (say  $u$ ) from HEAD's adjacency list.
  - (b) remove  $(HEAD, u)$  from HEAD's and  $u$ 's lists.
  - (c) **If**  $u$  is not in the path then  $HEAD = u$ ;  
**else** use the edge to ROTATE path and set HEAD to the successor of  $u$  in the path.

# Rotation

Let

$$P = v_1, v_2, \dots, v_k$$

be a simple path in an undirected graph  $G$ . Let  $(v_k, v_i)$  be an edge of  $G$ . Then

$$P' = v_1, \dots, v_i, v_k, v_{k-1}, \dots, v_{i+2}, v_{i+1}$$

is a rotated simple path in  $G$ .

**Theorem 1.** *If no adjacency list is empty before termination, then whp:*

- 1. The algorithm finds a Hamiltonian path;*
- 2. The algorithm terminates after  $O(n \log n)$  iterations of the while loop;*
- 3. No more than  $a \log n$  edges (for some constant  $a > 0$ ) are removed from any list.*

**Proof.** Consider a "less efficient" algorithm that for each vertex  $u$  keeps two lists:

1.  $new\_edges(u)$  - adjacent edges that were not used yet;
2.  $old\_edges(u)$  - edges that were already used.

When  $u$  is at the head of the path we choose:

1. A random element in  $old\_edges(u)$  with probability  $\frac{|old\_edges(u)|}{n}$ .
2.  $u$  with probability  $1/n$ .
3. with probability  $1 - \frac{1}{n} - \frac{|old\_edges(u)|}{n}$ , a random element in  $new\_edges(u)$  (and move it to  $old\_edges(u)$ ).

Make the successor (in the path) of the chosen node to be the new head by rotation. (In case  $u$  is chosen, then the successor is the last vertex in the path).

The number of iterations of the modified algorithm **stochastically dominates** the number of iterations of the original algorithm i.e.,

$$\forall x \quad \Pr(I_{mod} > x) \geq \Pr(I_{orig} > x)$$

The number of edges removed from a  $new\_edges(u)$  by the modified algorithm has the same distribution as in the original algorithm.

The probability that a given vertex becomes HEAD at a given iteration of the modified algorithm is  $1/n$ .

By the coupon collector analysis, if no list is empty before termination, then whp:

1. The algorithm terminates with an Hamiltonian path in  $3n \ln n$  iterations.
2. No vertex is chosen more than  $9 \ln n$  times.

□

**Corollary 1.** *If  $p \geq \frac{36 \log n}{n}$  and  $G \in G(n, p)$ , then whp  $G$  has a Hamiltonian path and the algorithm will find it.*

**Proof.** What should be  $p$  so that whp every vertex has more than  $18 \ln n$  neighbors?

The expected number of neighbors for a vertex is  $(n - 1)p$ ; using the Chernoff bound and the union bound we can show that if  $p \geq \frac{36 \ln n}{n}$ , then every vertex has more than  $18 \ln n$  neighbors. □