1 4-cycles ← cycle w/ 4 vertices

We use $G(n,p)$ to denote the distribution of graphs obtained by taking n vertices and for each pair of vertices i,j placing edge $\{i,j\}$ independently with probability p.

(a) Compute the expected number of edges in $G(n,p)$?

possible edges $= \binom{n}{2}$

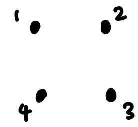
$X_i =$ indicator i^{th} edge is in $G(n,p)$

$$\mathbb{E}[X] = \sum_{i=1}^{\binom{n}{2}} \mathbb{E}[X_i] = p\binom{n}{2}$$

(b) Compute the expected number of 4-cycles in $G(n,p)$?

possible groups of 4 vertices $= \binom{n}{4}$

$X_i =$ indicator i^{th} group is a 4 cycle



$$\frac{4!}{2\cdot 4} = 3 \text{ possible cycles}$$

$$\mathbb{E}[X_i] = 3p^4$$

$$\mathbb{E}[X] = \sum_{i=1}^{\binom{n}{4}} \mathbb{E}[X_i] = 3p^4\binom{n}{4}$$

(c) Give a polynomial time randomized algorithm that takes in n as input and in poly(n)-time outputs a graph G such that G has no 4-cycles and the expected number of edges in G is $\Omega(n^{4/3})$.

hints: can post process a random graph
find probability of placing an edge, $p = ?$

Algo:
- generate random graph $G \sim G(n,p)$
- delete an edge in each 4-cycle

} poly(n) - time ✓
no 4-cycles ✓

$\mathbb{E}$[remaining edges]:

$$\mathbb{E}[\#\text{ edges}] - \mathbb{E}[\#\text{ 4-cycles}] = p\binom{n}{2} - 3p^4\binom{n}{4} = \Omega(n^{4/3})$$

Solve for p

$$p\,\frac{n(n-1)}{2} - 3p^4\,\frac{n(n-1)(n-2)(n-3)}{4} \geq \frac{pn^2 - 3p^4n^4}{4}$$

$$pn^2 = cn^{4/3} \quad \text{when} \quad p = cn^{-2/3}$$

$$(cn^{-2/3})^4 n^4 = c^4 n^{-8/3} n^{12/3} = c^4 n^{4/3}$$

Therefore set $p = \Theta(n^{-2/3})$.

2 Universal Hashing

Let $[m]$ denote the set $\{0, 1, ..., m-1\}$. Recall that a family of functions $\mathcal{H}$ is *universal* if for any $x \neq y$, $\Pr_{h \sim \mathcal{H}}[h(x) = h(y)] \leq 1/m$. That is, the chance that $h(x) = h(y)$ if we sample h uniformly at random from $\mathcal{H}$ is at most $1/m$.

For each of the following families of hash functions, determine whether or not it is universal. If it is universal, determine how many random bits are needed to choose a function from the family.

(a) $H = \{h_{a_1,a_2} : a_1, a_2 \in [m]\}$, where m is a fixed prime and

$$h_{a_1,a_2}(x_1, x_2) = a_1x_1 + a_2x_2 \mod m$$

Notice that each of these functions has signature $h_{a_1,a_2} : [m]^2 \to [m]$, that is, it maps a pair of integers in $[m]$ to a single integer in $[m]$.

$x = (x_1, x_2) \qquad y = (y_1, y_2)$

$a_1x_1 + a_2x_2 \equiv a_1y_1 + a_2y_2 \mod m$

$a_1(x_1 - y_1) \equiv a_2(y_2 - x_2) \mod m$

since m is prime, $(x_1 - y_1)$ has unique inverse

only equal if $a_1 = c(x_1 - y_1)^{-1}$ where $c \equiv a_2(y_2 - x_2)$

so a_1 needs to be one specific value in $[m]$

happens with probability $1/m$

UNIVERSAL

bits: $2\log m$

a_1, a_2 random integers in $[0, \dots, m-1]$

(b) H is as before, except that now $m = 2^k$ for $k > 1$ is some fixed power of 2.

NOT UNIVERSAL

$(x_1, x_2) = (0, 2^{k-1}) \qquad (y_1, y_2) = (2^{k-1}, 0)$

$a_1x_1 + a_2x_2 = a_2 2^{k-1} \mod 2^k$

$a_1y_1 + a_2y_2 = a_1 2^{k-1} \mod 2^k$

if a_1, a_2 even, $h(x) = h(y) = 0$

$\mathbb{P}[h(x) = h(y)] = \frac{1}{2} > \frac{1}{m} \qquad \frac{1}{m} = \frac{1}{2^k}$

(c) H is the set of all functions $f : [m] \to [m-1]$.

Exercise for the reader :)

3 Polynomial Identity Testing

Suppose we are given a polynomial $p(x_1, \ldots, x_k)$ over the reals such that p is not in any normal form. For example, we might be given the polynomial:

$$p(x_1, x_2, x_3) = (3x_1 + 2x_2)(2x_3 - 2x_1)(x_1 + x_2 + 3x_3) + x_2$$

We want to test if p is equal to 0, meaning that $p(a_1, a_2, \ldots, a_k) = 0$ on all real inputs $a_1, a_2, \ldots a_k$. This suggests a simple algorithm to check if p equals 0: test p on some $a_1, \ldots, a_k$ and say p equals 0 if and only if $p(a_1, \ldots, a_k) = 0$. However, this algorithm will fail if $p(a_1, \ldots, a_k) = 0$ and p does not equal 0. We will show through the following questions that if the degree of p is d, then for any finite set of reals S, by randomly choosing $a_1, \ldots, a_k \in S$, we get $Pr[p(a_1, \ldots, a_k) = 0] \leq \frac{d}{|S|}$. So by choosing a large enough subset, S, we are very likely to correctly determine whether p is equal to 0 or not. Note that the degree of, say, $p(x, y) = x^2y^2 + x^3$ is 4, since the monomial x^2y^2 has degree $4 = 2 + 2$. You may assume here that all basic operations on the reals take constant time.

(a) First let's see why a probabilistic algorithm might be necessary. Give a naive deterministic algorithm to test if a given polynomial p is equal to 0. What is the worst-case runtime of your algorithm?

multiply polynomial out and check if every coefficient is 0

exponential time

if $p(x_1, x_2) = (x_1 + x_2) \cdots (kx_1 + kx_2)$, multiplication takes c^k time

where c is a constant, k is number of parenthesis

(b) Show that if p is univariate and degree d, then $Pr[p(a_1) = 0] \leq \frac{d}{|S|}$

only one var $p(x)$

d degree polynomial has at most d zeros

worst case all d zeros in $|S|$

$\mathbb{P}[p(a_1) = 0] = \frac{d}{|S|}$ in worst case

(c) (Challenge question) Show by induction on k that for all nonzero degree-d polynomials p on k variables:

$$Pr[p(a_1, \ldots, a_k) = 0] \leq \frac{d}{|S|}$$

Hint: Write $p(x_1, \ldots, x_n) = \Sigma_{i=0}^{d} p_i(x_1, \ldots, x_{n-1}) x_n^i$

(d) Use the bound you found in the last part to show a way to check if two polynomials p and q are identical (i.e., yield the same outputs on all inputs).

$p = q$ if $p - q = 0$

create polynomial $r(x_1 \ldots x_k) = p(x_1 \ldots x_k) - q(x_1 \ldots x_k)$

apply algorithm on r

4 Monte Carlo Games

Let's suppose we have a Monte Carlo algorithm (a randomized algorithm which has a deterministic bound on its runtime, but which only outputs the correct answer some of the time). Call this algorithm A; then $A(x, r)$ is the output of A on input x and random bits r. In this question, we will think of A as a distribution over many deterministic algorithms. Convince yourself that this makes sense: after all, if we fix a setting to the random bits r, we get $A_r(x)$, which is a deterministic algorithm (which may be wrong on some inputs). Let's fix a set of algorithms S (say, polynomial-time algorithms). Note that A has whatever property defines S if and only if it is a distribution over only algorithms in S (for example, we say a Monte Carlo algorithm is polynomial time if and only if it runs in polynomial time for all settings to the randomness, which is equivalent to all the deterministic algorithms in its distribution running in polynomial time).

We will define a function $c(a, x)$ which indicates whether the deterministic algorithm $a \in S$ is correct on input x; $c(a, x) = 1$ if a is correct on input x, and 0 if it is incorrect.

Let's use this function to define a zero-sum game; the row player will choose a and the column player will choose x; then a payoff of $c(a, x)$ will go to the row player.

(a) Describe the action and goal of the row and column players. Interpret these in the setting of 'correctness of the randomly chosen algorithm' that we constructed the game from. *Hint: Since $c(a, x)$ is an indicator,* $\mathbb{E}[c(a, x)] = \Pr[c(a, x) = 1]$.

row player choose algo to maximize worst case

$$\max_{a \in S} \min_x \mathbb{E}[c(a,x)] = \max_{a \in S} \min_x \mathbb{P}[c(a,x) = 1]$$

col player choose input to minimize prob any algo would be correct

$$\min_x \max_{a \in S} \mathbb{E}[c(a,x)] = \min_x \max_{a \in S} \mathbb{P}[c(a,x) = 1]$$

(b) Using zero-sum game duality in conjunction with your interpretation above, what can we say about a problem if we know there exists a polynomial-time randomized algorithm which is correct with probability 2/3 on all inputs? What can we say if we know that there is a distribution of inputs on which no deterministic algorithm is correct with probability 2/3?

Hint: use the fact that a randomized algorithm induces a distribution over deterministic algorithms.

Statement 1:

lowest value col player can achieve is $2/3$

there is distribution D_a for which $\mathbb{P}_{a \sim D_a}[c(a,x) = 1] \geq 2/3$

by weak duality

there is deterministic algorithm that is correct w.p. $2/3$ on all inputs

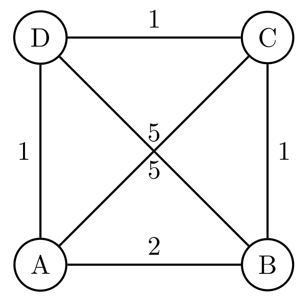
Statement 2:

No good randomized algo that can achieve correctness with prob $2/3$

contrapositive of statement 1

5 Traveling Salesman Problem

In the lecture, we learned an approximation algorithm for the Traveling Salesman Problem based on computing an MST and a depth first traversal. Suppose we run this approximation algorithm on the following graph:



The algorithm will return different tours based on the choices it makes during its depth first traversal.

1. Which DFS traversal leads to the best possible output tour?

2. Which DFS traversal leads to the worst possible output tour?

3. What is the approximation ratio given by the algorithm in the worst case for the above instance? Why is it worse than 2? (*Hint*: Consider the triangle inequality on the graph).