2MMD30: Graphs and Algorithms Lecture 12 by Jesper Nederlof, 23/03/2016

Isolation, Vector Coding and Counting Modulo 2.

Today we'll go into more depth on randomized techniques for probabilistic exponential time algorithms. In particular, we'll stick to the k-path problem mostly and discuss a relatively recent breakthrough result on it. See `https://www.youtube.com/watch?v=httHBboc6wY` for some recent video about this for non-experts (which doesn't tell us anything substantial but it might be fun too watch).

12.1 Reminder of Some Useful Linear Algebra

Let $\mathbb{Z}_2$ denote the set $\{0,1\}$. For $x,y \in \mathbb{Z}_2$, $x+y$ denotes $(x+y)\%2$. Similarly, $\mathbb{Z}_2^k$ denotes the set of k-dimensional binary vectors $\{0,1\}^k$ and given $x,y \in \mathbb{Z}_2^k$, $z = x+y$ denotes the vector in which the i-th coordinate z_i satisfies $z_i = x_i + y_i$. We let $\langle x,y\rangle$ denote the inner product $(\sum_{i=1}^k x_i y_i)\%2$. Two vectors x,y are called orthogonal if $\langle x,y\rangle = 0$.

A set of vectors $x^1,\ldots,x^n \in \mathbb{Z}_2^k$ is called *linearly independent* if $\sum_{i=1}^n \lambda_i x^i = 0$ implies $\lambda_1 = \lambda_2 = \cdots = \lambda_n = 0$. The rank $\mathsf{rk}(x^1,\ldots,x^n)$ is the maximum size $|B|$ over all subsets $B \subseteq \{1,\ldots,n\}$ such that $\{x^i\}_{i\in B}$ is linearly independent. Let $\mathsf{span}(x^1,\ldots,x^n) = \{\sum_{i=1}^n \lambda_i x^i : \lambda_i \in \{0,1\}\}$ denote the linear span of $x^1,\ldots,x^n$.

Theorem 12.1 (Rank-Nullity Theorem, convenient version)**.** *If $x^1,\ldots,x^n \in \mathbb{Z}_2^k$, the number of vectors $y \in \mathbb{Z}_2^k$ satisfying $\langle x^i, y\rangle = 0$ for every i is $2^{k-\mathsf{rk}(x^1,\ldots,x^n)}$.*

12.1.1 Proof of Rank-Nullity Theorem (not examined)

If $A \in \mathbb{Z}_2^{n\times k}$, and A has row vectors $a^1,\ldots,a^n \in \mathbb{Z}_2^k$ we let $\mathsf{rk}(A)$ denote $\mathsf{rk}(a^1,\ldots,a^n)$. If $N = \{x \in \mathbb{Z}_2^k : Ax \equiv_2 \mathbf{0}\}$, and $N = \{b^1,\ldots,b^m\}$, then the *nullity* is defined as $\mathsf{nul}(A) = \mathsf{rk}(b^1,\ldots,b^m)$. The set $N = \{b^1,\ldots,b^m\}$ is called the *null space* or *kernel* of A.

Theorem 12.2 (Rank-Nullity Theorem, classic version)**.** *If $A \in \mathbb{Z}_2^{n\times k}$, $\mathsf{rk}(A) + \mathsf{nul}(A) = k$.*

Proof. We see that $|\mathsf{span}(a^1,\ldots,a^n)| = 2^{\mathsf{rk}(A)}$, and thus $\mathsf{rk}(\mathsf{span}(a^1,\ldots,a^n)) = \mathsf{rk}(a^1,\ldots,a^n)$. It follows that we can add distinct rows in A without influencing the rank (i.e. applying elementary row operations does not influence the rank). It is easy to see, and well-known, that we can keep applying such operations to rewrite A into *reduced row-echelon form*, i.e. (i) rows with only zeroes

$$\begin{bmatrix} 1 & 0 & a_1 & 0 & b_1 \\ 0 & 1 & a_2 & 0 & b_2 \\ 0 & 0 & 0 & 1 & b_3 \end{bmatrix}$$

Figure 12.1: Example of a matrix in reduced row-echelon form, where $a_1, a_2, b_1, b_2, b_3 \in \mathbb{Z}_2$ are unspecified. In this example, $n = 3$ and $k = 6$, the rank is 3 the nullity is 3, the pivot columns are columns 1,2 and 4 and the non-pivot columns are 3 and 5. To get an element $x \in \mathbb{Z}_2^5$ in the null space by, set $x_1 = x_3a_1 + x_5b_1$, $x_2 = x_3a_2 + x_5b_2$ and $x_4 = x_5b_3$.

appear last (*ii*) columns with leading ones only have the leading one as non-zero entry (a leading one is a one that is the first non-zero entry of a row)[1].

Let $N = \{x \in \{0,1\}^k : Ax \equiv_2 \mathbf{0}\}$, be the null space of A where $\mathbf{0}$ is the n-dimensional vector with only zeroes. So $\mathsf{nul}(A) = \mathsf{rk}(N)$. By the above we may assume that A is in reduced row-echelon form. Then a solution x can be constructed by making independent decisions whether to include the non-pivot columns (i.e. columns not including leading ones) and given a decision for each such column, there is a unique way to include the pivot columns to have all included columns sum to $\mathbf{0}$. □

12.2 Vector Coding

In the color-coding approach, we saw two algorithms for k-path. The second one was faster because we studied an event that was more likely to happen (i.e. the path only needed to be colorful instead of consecutively numbered). Now we'll be using a different way of 'coloring' where we do not assign a number to every vertex but a vector from $\mathbb{Z}_2^k$. This requires a bit more technical work, but the upside is that we'll be studying an event that happens with constant probability, so there's no need for independent repetitions.

Algorithm `kpath3`$(G = (V, E), k)$ — G is directed
Output: Whether G has a k-path, with one-sided constant error probability
1: **for** $v \in V$ **do**
2: Pick $c(v) \in \mathbb{Z}_2^k$ uniformly at random
3: **return** `linindpath`$'(G, k, c)$

Algorithm 1: $O^*(2^k)$ time algorithm for k-path with constant one-sided error probability.

Here, we refer to a k-path with vertices $p_1, \ldots, p_k$ as being *linearly independent* if the vectors $c(p_1), \ldots, c(p_k)$ are linearly independent. We assume for now that `linindpath`$'$ returns whether G has a linearly independent k-path (wrt c) with one-sided constant error probability. We'll see later that such an algorithm can be implemented such that it runs in $O^*(2^k)$ time. Therefore, for now we only need to prove that we introduce false negatives with at most constant probability by restricting our attention to linearly independent k-paths:

[1] Note that if we are not in $\mathbb{Z}_2$, there are more restrictions.

Lemma 12.1. *Choosing $c(v) \in \mathbb{Z}_2^k$ uniformly and independently at random for every $v \in V$, we have that a k-path is linearly independent with probability at least $1/4$.*

Proof. Let $p_1, \ldots, p_k$ be the vertices on the k-path. We need to lower bound the probability that the vectors $\{c(p_i)\}_{1 \le i \le k}$ are linearly independent. To do this, consider the process where we pick the vectors one by one and consider the probability that the newly picked vector is in the span of the already picked vectors. For the vectors to be linearly independent, it is sufficient and necessary to avoid this event in every iteration so we have, using Bayes' rule $\Pr[A \cap B] = \Pr[A|B]\Pr[B]$:

$$\begin{aligned}
\Pr[\{c(p_1), \ldots, c(p_k)\} \text{ are LI}] &= \prod_{i=1}^{k} \Pr\left[\{c(p_1), \ldots, c(p_i)\} \text{ are LI} \middle| \{c(p_1), \ldots, c(p_{i-1})\} \text{ are LI}\right] \\
&= \prod_{i=1}^{k} \Pr\left[c(p_i) \notin \mathsf{span}(c(p_1), \ldots, c(p_{i-1})) \middle| \{c(p_1), \ldots, c(p_{i-1})\} \text{ are LI}\right] \\
&= \prod_{i=1}^{k} \left(1 - \frac{2^{i-1}}{2^k}\right) \\
&= \prod_{i=1}^{k} \left(1 - \frac{2^{i-k}}{2}\right) \qquad \text{using } 1 - x/2 \ge 2^{-x} \text{ for } x \in [0,1] \\
&\ge \prod_{i=1}^{k} 2^{-2^{i-k}} = 2^{-\sum_{i=1}^{k} 2^{i-k}} \ge 2^{-2} = 1/4. \quad \text{using } \textstyle\sum_{i=1}^{k} 2^{i-k} \le 2
\end{aligned}$$

□

12.3 Counting Linearly Independent k-Paths Modulo 2

Why would it be easier to determine the existence of linearly independent k-paths than the existence of k-paths? Recalling the first color-coding approach from Section 11.2.1 and the inclusion exclusion approach from Exercise 11.6, the crucial point is that the types of walks we are looking for automatically are paths: in a DAG indeed a walk automatically is a path, and a colorful walk also needs to be a path. This is not different in our new setting. Suppose that $(p_1, \ldots, p_k)$ is a walk and the vectors $c(p_1), \ldots, c(p_k)$ are linearly independent, then $c_1, \ldots, c_k$ needs to be a path: if it wouldn't, $p_i = p_j$ for some $i < j$ and $c(p_i) = c(p_j)$ are linearly dependent. So since it's just the same, we may focus on determining the existence of linearly independent k-walks (e.g. walks on k vertices) as well.

So before we continue, let's revisit the part of Section 6.6 from Lecture 6 concerning walks[2]. Given a directed graph $G = (V, E)$ and vertices s, t and integer k we defined $W_A(s, t, l)$ as the number of walks of length l (i.e. using l edges, so $l+1$ vertices) from s to t in $G[A]$. Here, we define $w_A(l) = \sum_{s,t} w_A(s, t, l)$, i.e. the number of walks on l edges. We saw in lecture 6 that $w_A(s, t, l)$ can be computed in polynomial time, so $w_A(l)$ can be computed in polynomial time as well.

Unfortunately, dynamic programming or inclusion exclusion do not right away give us a fast algorithms for finding linearly independent paths. However, inspired by the inclusion exclusion

[2]We'll define W_A where A is a set of allowed vertices rather than forbidden since it's more convenience here, but this is just a minor notation change.

principle, we note that if the vectors $c(p_1), \ldots, c(p_k)$ are linearly dependent, we have some freedom in picking vectors orthogonal to these vectors. In particular, for $y \in \mathbb{Z}_2^k$ let $ORTH(y)$ denote $\{v \in V : \langle c(v), y\rangle = 0\}$, then the rank-nullity theorem implies that

$$\begin{aligned}\sum_{y\in\mathbb{Z}_2^k} w_{ORTH(y)}(k) &= \left|\left\{\big(y,(p_1,\ldots,p_k)\big) : (p_1,\ldots,p_k) \text{ walk of } G \text{ on } k\text{-vertices} \wedge \forall i : \langle c(p_i), y\rangle = 0\right\}\right| \\ &= \sum_{(p_1,\ldots,p_k) \text{ walk of } G \text{ on } k \text{ vertices}} 2^{k-\mathsf{rk}(c(p_1),c(p_2),\ldots,c(p_k))} \\ &\equiv_2 \sum_{(p_1,\ldots,p_k) \text{ LI walk of } G \text{ on } k\text{-vertices}} 1 \\ &= |\{(p_1,\ldots,p_k) \text{ LI walk of } G \text{ on } k \text{ vertices}\}|.\end{aligned}$$

And indeed we do know how to evaluate $\sum_{y\in\mathbb{Z}_2^k} w_{ORTH(y)}(y)$ in $O^*(2^k)$ time. So we did not manage yet to determine whether a linearly independent k-path exists, but we did manage to find out whether there is an even number of not in $O^*(2^k)$ time!

12.4 Isolation lemma

Of course, it could very well be that there is an even number of linearly independent k-paths, in which case we are not able the determine whether a k-path exists, so how are we going to use the algorithm that determines the parity of the number of k-paths? The idea is introduce some more randomness in terms of weights. This is useful in a very general sense so let us introduce the appropriate terminology

Definition 12.1. *Given a set U, set family $\mathcal{F} \subseteq 2^U$ and a function $\omega : U \to \mathbb{Z}$, a set $S \in \mathcal{F}$ is called a* minimizer of ω in $\mathcal{F}$ *if $\omega(S) = \min_{S'\in\mathcal{F}} \omega(S')$. The function ω is said to* isolate *the set family $\mathcal{F} \subseteq 2^U$ if there is a unique minimizer of ω in $\mathcal{F}$.*

Recall here, that for $X \subseteq U$, $\omega(X)$ denotes $\sum_{u\in X} \omega(u)$. For $X \subseteq U$ we'll refer to $\omega(X)$ as the *weight* of X.

Lemma 12.2 (Isolation Lemma)**.** *Let $\mathcal{F} \subseteq 2^U$ be a set family over a universe U with $|\mathcal{F}| > 0$. For each $u \in U$, choose a weight $\omega(u) \in \{1, 2, \ldots, N\}$ uniformly and independently at random. Then*

$$\Pr[\omega \text{ isolates } \mathcal{F}] \geq 1 - \frac{|U|}{N}$$

Proof. For every element $e \in U$, define

$$a(e) = \min_{e\notin S\in\mathcal{F}} \omega(S) - \min_{e\in S\in\mathcal{F}} \omega(S \setminus \{e\}).$$

Notice that for every element $e \in U$, $a(e)$ does not depend on $\omega(e)$ so the random variables $a(e)$ and $\omega(e)$ are independent. Hence, taking probability over all weight functions $\omega : U \to \{1, 2, \ldots, N\}$ uniformly at random, we know that for every element $e \in U$, $\Pr[a(e) = \omega(e)] \leq \frac{1}{N}$. Now assume $S_1, S_2 \in \mathcal{F}$ are both minimizers of ω in $\mathcal{F}$ such that $S_1 \neq S_2$. Let $e \in S_2 \setminus S_1$. Then we know that

$$\min_{e\notin S\in\mathcal{F}} \omega(S) = \omega(S_1) = \omega(S_2) = \min_{e\in S\in\mathcal{F}} \omega(S \setminus e) + \omega(e),$$

and subtracting $\min_{e \in S \in \mathcal{F}} \omega(S \setminus e)$ from both sides results in $a(e) = \omega(e)$. Then, we know that the probability that ω does not isolate $\mathcal{F}$ is

$$\Pr[\exists \text{two distinct minimizers of } \omega \text{ in } \mathcal{F}] \leq \Pr[\exists e : a(e) = \omega(e)] \leq \sum_{e \in U} \Pr[a(e) = \omega(e)] \leq \frac{|U|}{N},$$

where the second last inequality follows from the union bound, i.e. $\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$. $\square$

12.5 Finding Linearly Independent k-Paths

Suppose $G = (V, E)$ is directed graph, let k be an integer and let $c : V \to \mathbb{Z}_2^k$. We can uniquely identify a path on k vertices $(p_1, \ldots, p_k)$ with the $k-1$ edges $\{(p_1, p_2), (p_2, p_3), \ldots, (p_{k-1}, p_k)\} \subseteq E$, so if we let $U = E$, $\mathcal{F}$ be the edge sets of all linearly independent k-paths and pick $\omega(e) \in \{1, \ldots, 2|E|\}$, uniformly and independently at random for every $e \in E$, then by the isolation lemma

$$\Pr_{\omega}[\exists \text{a unique minimum weight linearly independent } k\text{-path}] \geq 1/2.$$

The idea now is to compute the number of linearly independent k-paths of a specific weight; then we will have an odd count for the minimum weight with probability at least $1/2$. Specifically, let $w_A^W(k)$ be the number of walks on k vertices in $G[A]$ of weight W, i.e.

$$w_A^W(k) = |\{\text{walks } (p_1, \ldots, p_k) \text{ in } G[A] : \omega(\{(p_1, p_2), (p_2, p_3), \ldots, (p_l, p_k)\}) = W\}|,$$

then by exactly the same reasoning as in Section 12.3 we see that

$$\sum_{y \in \mathbb{Z}_2^k} w_{ORTH(y)}^W(k) \equiv_2 |\{(p_1, \ldots, p_k) \text{ LI walk of } G \text{ on } k \text{ vertices of weight } W\}|.$$

Since $W = 2|E|$, $w_{ORTH(y)}^W(k)$ can be computed in polynomial time, and it follows that we can count the number of linearly independent k-paths modulo 2 in G of weight W in $O^*(2^k)$ time.

Algorithm `linindkpaths'`(G, k, c) G is directed, $c : V \to \mathbb{Z}_2^k$
Output: Whether G has a linearly independent walk on k vertices, with constant one-sided error probability.

```
for e ∈ E do
    Pick ω(e) ∈ {1, ..., 2|E|} uniformly at random
for W = 1, ..., |E|k do
    sum = 0
    for y ∈ Z_2^k do
        term = w^W_{ORTH(y)}(k)
        sum = (sum + term)%2
    if sum = 1 then return true
return false
```

Algorithm 2: $O^*(2^k)$ time randomized algorithm for k-path.

12.6 Björklund's algorithm (the algorithms are not examined)

In the previous section we saw that working modulo 2 can be very useful and a powerful lemma allowed to use counting modulo 2 algorithms to solve the decision variant of a problem. Given this approach, a natural question is: can we do less work to filter out the k-walks that are not k-paths by arguing that they will always come in pairs anyway? We will see that if G is undirected we can, so all graphs in this section will be undirected.

Suppose we have a k-walk $(p_1, \ldots, p_k)$ from s to t that is not a k-path, so there are $i < j$ such that $p_i = p_j$. Then the sequence with the part between p_i and p_j reversed, that is, the sequence $(p_1, \ldots, p_j, p_{j-1}, \ldots, p_{i+1}, p_i, \ldots, p_k)$ also is a k-walk! Based on this, let us see which k-walks come in pairs, so they will cancel modulo 2 and we do not need to worry about them. Given a k-walk that is not a k-path let i be the minimum such that p_i occurs twice, and let j be the minimum such that $p_i = p_j$. The mapping ϕ inverting the subsequence p_i, p_j in the walk is its own inverse!

So indeed, all walks P such that $\phi(P) \neq P$ are their own inverse, and we only need to take care of the fixed points of this mapping, i.e. the sequences such that $p_i, \ldots, p_j$ is a palindrome[3].

12.6.1 Bipartite graphs

Definition 12.2. *Let G be a bipartite graph with parts (V_1, V_2) and fix $s \in V_1$ and $t \in V_2$. For every $v \in V_1$, assign a vector $c(v) \in \{0,1\}^{k/2}$. Say a k-walk $(p_1, \ldots, p_k)$ from s to t is* linearly independent *if the vectors $c(p_1), c(p_3), \ldots, c(p_{k-1})$ are linearly independent. Say it is* admissible *if it does not contain a, b, a as a subsequence for $b \in V_1, a \in V_2$. A* candidate walk *is a linearly independent admissible k-walk from s to t.*

We will show that we can restrict ourselves to filtering for the candidate walks; the candidate walks that are not k-paths will vanish modulo 2:

Lemma 12.3. *The mapping ϕ is a pairing on the set of candidate walks that are not k-paths.*

Proof. Restricted to candidate walks, ϕ will be its own inverse. So it is sufficient to show that ϕ has no candidate walk as a fixed point and the image of a candidate walk is also a candidate walk.

Let $P = (p_1, \ldots, p_i, \ldots, p_j, \ldots, p_k)$ be as before. Let us first argue that it is not a fixed point: since the walk is linearly independent, $p_i \in V_2$, and since P is admissible it cannot be that $i+2 = j$. Thus in order to be a fixed point, the sub-sequence needs to be of the type p_i, x, S, y, p_j for a non-empty sub-sequence S. However, $a, b \in V_1$ and by the linear independence property $a \neq b$ so reversal gives a different sequence.

To see that $\phi(P)$ also is a candidate walk note that linear independence is determined only by $\{p_1, \ldots, p_k\}$, which is clearly invariant. It remains to show that $\phi(P)$ does not have a, b, a as a sub-sequence for $v_y \in V_1$, but this can only happen when it has already in P, or $p_{i-2} = p_i$ or $p_{j+2} = p_j$, which contradicts P being admissible. □

Let $w^W_{ORTH(y)}(s, t, k)$ denote the number of admissible k-walks of weight W from s to t in $G[ORTH(y)]$, then

$$\sum_{y \in \{0,1\}^{k/2}} w^W_{s,t,ORTH(y)}(k) \equiv_2 |\{\text{candidate walks of weight } W\}|$$

[3]It is tempting to try to do this directly by counting walks which do not directly backtrack, but this is hard since the pairing may introduce some backtrack again (consider 1634631 and 1643631).

which is congruent to the number of k-paths from s to t of weight b modulo 2 by Lemma 12.3. Then, $w^W_{ORTH(y)}(s,t,k)$ can be computed in polynomial time similarly as before. Let $w^W_{ORTH(y)}(s,t,k)$ denote the number of admissible k-walks of weight W from s to t in $G[ORTH(y)]$ such that p is the second last vertex, then for $s,t \in A$

$$w^W_A(s,t,k) = \begin{cases} 0 & \text{if } k=2 \text{ and } (s,t) \notin E \vee W \neq \omega(s,t), \\ 1 & \text{if } k=2,\ (s,t)\in E \text{ and } W=\omega(s,t), \\ \sum\limits_{\substack{t_1 \in N^-(t)\cap A \\ t_2 \in N^-(t_1)\cap A\setminus\{t\}}} w_A^{W-\omega(\{(t_1,t),(t_2,t_1)\})}(s,t_2,k-2) & \text{otherwise,} \end{cases}$$

Algorithm `kpathbip`(G,k) G is bipartite with parts V_1, V_2 and undirected
Output: Whether G has a linearly independent walk on k vertices starting in V_1 and ending in V_2, with constant one-sided error probability.

```
1: for s ∈ V_1, t ∈ V_2 do
2:   for e ∈ E do
3:     Pick ω(e) ∈ {1,...,2|E|} uniformly at random
4:   for v ∈ V_1 do
5:     Pick c(v) ∈ Z_2^{k/2} uniformly at random
6:   for W = 1,...,|E|k do
7:     sum = 0
8:     for y ∈ Z_2^{k/2} do
9:       term = w^W_{ORTH(y)}(s,t,k)      admissible k-walks from s to t in G[ORTH(y)]
10:      sum = (sum + term)%2
11:    if sum = 1 then return true
12: return false
```

Algorithm 3: $O^*(2^{k/2})$ time randomized algorithm for k-path in bipartite undirected graphs.

12.6.2 General graphs (not examined)

Let us fix a partition $V = V_1 \cup V_2$ of the vertices and $s \in V_1$ and $t \in V_2$. Say a k-walk $(p_1,\ldots,p_k)$ is *d-split* if $|\{i : p_i \in V_1\}| + |\{i : p_i, p_{i+1} \in V_2\}| = d$.

Fix an integer d, and assign a d-dimensional vector $c(v) \in \mathbb{Z}_2^d$ to every $v \in V_1$ and edge $e \in E \times (V_2 \times V_2)$. Call a d-split k-walk *linearly independent* if the set of vectors $\{c(p_i) : p_i \in V_1\} \cup \{c(p_i,p_{i+1}) : p_i, p_{i+1} \in V_2\}$ is linearly independent. A *candidate walk* is a k-walk from s to t that is admissible (as defined in the previous section), d-split and linearly independent.

Lemma 12.4. *For every W, the parity of the number of candidate walks of weight W equals the number of linearly independent d-split k-paths of weight W.*

For a partition V_1, V_2 of V, $s \in V_1, t \in V_2$ integers l, d, W, $F \subseteq V_1$ and $F' \subseteq E \cap (V_2 \cap V_2)$ let $w^W_{A,A'}(s,t,l,d)$ denote the number of d-split admissible k-walks in the graph (A,A') of weight W. Since such a walk is a candidate walk if and only if it is additionally linearly independent we have

similarly to the previous sections that

$$\#\text{candidate walks of weight } W \equiv_2 \sum_{y \in \mathbb{Z}_2^d} w^W_{ORTH(y),ORTH'(y)}(s,t,k,d),$$

where $ORTH'(y)$ denotes all $e \in E$ such that $\langle c(e), x\rangle = 0$.

Using a dynamic programming algorithm very similar to the one we saw in the previous section, $w^W_{F,F'}(s,t,l,d)$ can be computed in polynomial time (assuming $W \leq 2|E|$), so we can compute the parity of the number of candidate walks of weight W for every W in $O^*(2^d)$ time. Thus using Lemma 12.4 and the isolation lemma as before, we know how to determine whether G has a k-path that is d-split in $O^*(2^d)$ time. In particular, this is checked in Line 4 to Line 16 in the following algorithm

Algorithm `kpath4`(G,k) — G undirected

Output: Whether G has a linearly independent walk on k vertices starting in V_1 and ending in V_2, with constant one-sided error probability.

1: **for** $t = 1,\ldots,k$ **do**
   Run k independent trials to boost the success probability
2: Pick a partition V_1, V_2 by including every vertex uniformly in either set
3: **for** $d = 0,\ldots,\lceil 3k/4\rceil$ **do**
4: **for** $s \in V_1$, $t \in V_2$ **do**
5: **for** $e \in E$ **do**
6: Pick $\omega(e) \in \{1,\ldots,2|E|\}$ uniformly at random
7: **for** $v \in V_1$ **do**
8: Pick $c(v) \in \mathbb{Z}_2^d$ uniformly at random
9: **for** $e \in E \cap (V_2 \times V_2)$ **do**
10: Pick $c(e) \in \mathbb{Z}_2^d$ uniformly at random
11: **for** $W = 1,\ldots,|E|k$ **do**
12: $sum = 0$
13: **for** $y \in \mathbb{Z}_2^d$ **do**
14: $term = w^W_{ORTH(y),ORTH'(y)}(s,t,k,d)$ — Compute in poly time using Ex. 12.8
15: $sum = (sum + term)\%2$
16: **if** $sum = 1$ **then return true**
17: **return false**

Algorithm 4: $O^*(2^{3k/4})$ time randomized algorithm for k-path in undirected graphs.

It remains to show that if a k-path exists, it will be d-split for some $d \leq \lceil 3k/4\rceil$ in some iteration of the loop at Line 3 with probability at least constant probability. As usual, let's first see what the probability is this happens in some particular iteration.

Recall that Markov's inequality states that for any non-negative variable X, $\Pr(X \geq a) \leq \mathbb{E}[X]/a$. If we fix a k-path $p_1,\ldots,p_k$ and let denote $X = |\{i : p_i \in V_1\}| + |\{i : p_i, p_{i+1} \in V_2\}|$, we see that

$$\mathbb{E}[X] = k/2 + (k-1)/4 = \frac{3k-1}{4},$$

so applying Markov's inequality we have

$$\Pr[X \geq \lceil 3k/4 \rceil] \leq \frac{3k-1}{3k} = 1 - 1/3k.$$

Thus the probability that in every iteration of the Loop at Line 3, $p_1, \ldots, p_k$ is not d-split by V_1, V_2 for some $d \leq \lceil 3k/4 \rceil$ is at most $(1 - \frac{1}{3k})^k \leq e^{-3}$.

12.7 Exercises

Excercise 12.1. Why does the linearly independent path problem contain the colorful path problem as a special case? How does the approach outlined in Section 12.3 count the number of k-paths modulo 2 in this case?

Excercise 12.2. Modify/use Algorithm `linindkpaths'` to solve k-path in undirected graphs.

Excercise 12.3. An alternative way of using the isolation lemma would have been to assign random weights to every vertex rather than edge. Why this is not a good idea?

Excercise 12.4. A triangle in an undirected graph $G = (V, E)$ is a triple of vertices (u, v, w) with a $(u, v), (v, w), (u, w) \in E$. A k-triangle-packing is a collection of triangles $T_1, \ldots, T_k$ that are mutually disjoint. Given an algorithm that given G and integer k determines whether G has a triangle packing in $O^*(8^k)$ time with constant one-sided error probability.

Excercise 12.5. Give a polynomial time algorithm that computes the parity of the number of perfect matchings of a bipartite (multi-)graph. Assume that computing the determinant of a matrix can be done in polynomial time.

Excercise 12.6. Let $G = (V, E)$ be a bipartite graph with parts V_1, V_2. Show that $|\{X \subseteq V_1 : N(X) = V_2\}| \equiv_2 |\{X \subseteq V : X \text{ independent set}\}|$. Hint: Group the independent sets on $X \cap V_1$, how many sets $Y \subseteq V_2$ such that $X \cup Y$ is an independent set are there?

From here, you may also want to revisit exercises of previous lectures.

Excercise 12.7.[Not examined] In the 3-dimensional matching problem we are given three sets A, B, C with $|A| = |B| = |C| = n$ and triples $T_1, \ldots, T_n \subseteq A \times B \times C$. A matching is a partition $S_1, \ldots, S_n$ of $A \cup B \cup C$. Compute the parity of the number of matchings in $O^*(2^n)$ time. Hint: call an n-tuple $S_1, \ldots, S_n$ a *pseudo-matching* if for every element $e \in A \cup B$ there is exactly one i such that $e \in S_i$. Use inclusion exclusion with U the set of all pseudo-matchings and for every $i \in C$ a property P_i being all pseudo-matchings containing vertex i. Use Exercise 12.5 to compute $|\bigcap_{i \in F} \overline{P_i}|$ modulo 2 in polynomial time.

Excercise 12.8.[Not examined] Show how to compute $w^W_{ORTH(y),ORTH'(y)}(s, t, k, d)$ in polynomial time (assuming $W \leq 2|E|$). Hint: Use dynamic programming similarly as we did for computing $w^W_F(s, t, l)$, but we need yet another variable d in the recurrence to keep track of how many times vertices from V_1 and edges from $E \cap (V_2 \times V_2)$ are used.

Excercise 12.9.[Not examined] Prove Lemma 12.4. Hint: The proof is very similar to the proof of Lemma 12.3.