1 Baby Brooks

Notational note. I use $\mathcal{G}$ to denote the Greedy algorithm.

We wish to prove that if $G = (V, E)$ is connected and non-regular, then $\chi(G) \leq \Delta$.

Let $x_0 \in V$ be s.t. $d(x_0) = \delta$. Since G is connected, running BFS from x_0 adds all vertices to the BFS tree. Let O^{-1} be the ordering of the vertices s.t. z is the ith vertex if it was the ith one to be added by BFS. Trivially, x_0 is the first vertex in O^{-1}. Let O be the reverse order, with x_0 last. We will prove $\mathcal{G}$ colors G with at most Δ colors if it uses the ordering O.

Observe that, in the BFS run, every $x \neq x_0$ is inserted by a neighbor that was already in the tree. In other words, in the O^{-1} order, every vertex has a neighbor that precedes him in the order. Consequently, in O, every $x \neq x_0$ has a neighbor that succeeds him in the order.

It follows that the worst case scenario for the coloring of $x \neq x_0$ is that it has $d(x) - 1$ preceding neighbors. $\therefore \mathcal{G}$ eliminates at most $d(x) - 1 \leq \Delta - 1$ colors. Then x can be colored with a color in $\{1, \ldots, \Delta\}$.

When $\mathcal{G}$ reaches x_0 it eliminates at most $d(x_0) = \delta$ colors. Since G is non-regular $\delta < \Delta$. $\therefore$ There is at least one color for x_0 in $\{1, 2, \ldots, \Delta\}$.

2 Max flow, min cut

Let f a flow over a network $\mathcal{N}$. We want to prove two things: *(1)* $v(f) \leq Cap(S)$ for any cut S and *(2)* f is maximal iff there is a cut S s.t. $v(f) = Cap(S)$.

(1) We know $v(f) = f(S, \overline{S}) - f(\overline{S}, S)$. Since $f(A, B)$ is a sum over f and $0 \leq f(\overrightarrow{ab}) \leq c(\overrightarrow{ab})$ for any $\overrightarrow{ab} \in E$,

$$v(f) = f(S, \overline{S}) - f(\overline{S}, S) \leq f(S, \overline{S})$$

The same logic implies $f(S, \overline{S}) \leq c(S, \overline{S}) = Cap(S)$. Then $v(f) \leq f(S, \overline{S}) \leq Cap(S)$. ■

(2: $\Leftarrow$) Assume there is a cut S s.t. $v(f) = Cap(S)$. Let g an arbitrary flow. Then $v(g) \leq Cap(S) = v(f)$. Then f is maximal. Furthermore, it is trivial by definition of Cap that $Cap(T) \geq v(f)$ for any cut T. Then $Cap(T) \geq Cap(S) \Rightarrow S$ is minimal.

(2 : $\Rightarrow$) Assume f is maximal. Let

$$S = \{s\} \cup \{x \in V : \exists f\text{-camino aumentante entre } s \text{ y } x\}$$

S is a cut because, if $t \in S$, there is an augmenting path $s \ldots t$ and the flow can be augmented, which contradicts that f is maximal.

Recall that $v(f) = f(S, \overline{S}) - f(\overline{S}, S)$. The first term in the difference is

$$f(S, \overline{S}) = \sum_{x \in S, z \notin S, \overrightarrow{xz} \in E} f(\overrightarrow{xz})$$

Let $\overrightarrow{xz} \in E$ a side in the range of the sum above. Then there is an augmenting path $s \ldots x$ and there is no augmenting path $s \ldots z$. But $\overrightarrow{xz} \in E$ and $s \ldots x \ldots z$ is a path. Since it cannot be an augmenting path, we must have $f(\overrightarrow{xz}) = c(\overrightarrow{xz})$. Then $f(\overrightarrow{xz}) = c(\overrightarrow{xz})$ for all $x \in S$, $z \notin S, \overrightarrow{xz} \in E$. Therefore

$$f(S, \overline{S}) = \sum_{\ldots} f(\overrightarrow{xz}) = \sum_{\ldots} c(\overrightarrow{xz}) = Cap(S)$$

Now consider the second term in the difference:

$$f(\overline{S}, S) = \sum_{w \notin S, x \in S, \overrightarrow{wx} \in E} f(\overrightarrow{wx})$$

Let $\overrightarrow{wx}$ an arbitrary side in the sum above. Again, there must be an augmenting path $s \ldots x$, but not one $s \ldots w$. But $\overrightarrow{wx}$ is a side, and then $s \ldots \overleftarrow{xw}$ is not augmenting only if $f(\overleftarrow{xw}) = 0$. This means $f(\overrightarrow{wx}) = 0$ for all $\overrightarrow{wx}$ in the range of the sum above.

$\therefore v(f) = Cap(S) - 0 = Cap(S)$. ■

3 Networks

3.1 Edmond-Karp: Complexity

The complexity of E.K. is determined by the complexity of finding the f_i-augmenting paths, the number of such paths, and the complexity of updating the flow with such paths.

To find f_i-augmenting paths, EK uses BFS with a complexity $O(m)$. Updating the flow has complexity $O(n)$. So the question is what is the number φ of f_i augmenting paths (or runs of BFS) that are used.

The number of such paths is bounded by the number of sides times the number of times a side may become critical. We shall determine how many times a side may become critical.

Let $f_1, f_2, \ldots$ be the iterations of EK. Let $\overrightarrow{xz}$ be a side that became critical at iteration k. There are two options: it either saturated being forward or emptied being backward.

Saturated being forward. If $\overrightarrow{xz}$ saturated being forward, then $p_k = s \ldots \overrightarrow{xz} \ldots t$ is the form of the augmenting path used to determine f_k.

Assume $\overrightarrow{xz}$ becomes critical again at some iteration j, so that $p_j = s \ldots \overrightarrow{xz} \ldots t$ is an augmenting path. For this to occur there must exist some iteration i, with $k < i < j$, such that $p_i = s \ldots \overleftarrow{xz} \ldots t$ is an augmenting path. In other words, z must have returned some of the flow to x. Since we are using EK, the augmenting paths are of minimal length, and the distance between two vertices never decreases. This means

$$\begin{aligned} d_j(t) &\geq d_i(x) + b_i(x) \\ &= d_i(z) + 1 + b_i(x) \\ &\geq d_k(z) + 1 + d_k(x) \\ &= d_k(x) + 1 + 1 + b_k(x) \\ &= d_k(t) + 2 \end{aligned}$$

Emptied being backward. Now, assume $\overrightarrow{xz}$ is a side that saturated being backward on the kth iteration; this is, that $p_k = s \ldots \overleftarrow{xz} \ldots t$ was the path found. Assume it empties again at some iteration j, so that $p_j = s \ldots \overleftarrow{xz} \ldots t$ is the f_j path. Then there is some i, with $k < i < j$, such that $p_i = s \ldots \overrightarrow{xz} \ldots t$ is a path; i.e. the side was used forward. Now, all these paths are of minimal length, because we are using E.K. And in all succesive paths, the distance from s to t never decreases. Then,

$$\begin{aligned} d_j(t) &\geq d_i(t) \\ &= d_i(z) + b_i(z) \\ &= d_i(x) + 1 + b_i(z) \\ &\geq d_k(x) + 1 + b_k(z) \\ &= d_k(z) + 1 + 1 + b_k(z) \\ &= d_k(t) + 2 \end{aligned}$$

In both cases, the distance from s to t had to increase by at least two unites. But the distance from s to t is bounded by n; this is, it is $O(n)$. Then a side may become critical $O(\frac{n}{2}) = O(n)$ times. Since there are m sides, the number of times an arbitrary side will become critical is $O(mn)$.

$\therefore$ The complexity is $O(mn)\ [O(n) + O(m)] = O(mn)O(m) = O(m^2n)$.

3.2 Edmond-Karp: Augmenting paths are non-decreasing

Let $A = \{x \in V : d_{k+1}(x) < d_k(x)\}$ and assume $A \neq \emptyset$. Let $x_0 \in A$ be the vertex whose distance $d_{k+1}(x_0)$ from s is minimal; i.e. $d_{k+1}(x_0) \leq d_{k+1}(y)\ \forall y \in A$. Since $x_0 \in A$, $d_{k+1}(x_0) < d_k(x_0) \leq \infty$. Then there exists a $\mathcal{P}_{k+1} = s \dots x_0$ of minimal length. Let z be the predecessor of x_0 in this path.

By definition of d_f, the length of the path is $d_{k+1}(x_0)$. Because the path is of minimal length to x_0, it is of minimal length to any predecessor of x_0 in it, including z. Then $d_{k+1}(z) = d_{k+1}(x_0) - 1$. This implies $z \notin A$.

There are two possible cases: either $\overrightarrow{xz} \in E$ or $\overrightarrow{zx} \in E$.

(Case 1): If $\overrightarrow{zx_0} \in E$, then $d_{k+1}(z) < d_{k+1}(x_0)$. Since $z \notin A$,

$$d_k(z) \leq d_{k+1}(z) < d_{k+1}(x_0) < \infty$$

Since $d_k(z) < \infty$, there is an f_k-augmenting path from s to z. Then, in principle, $s \dots zx$ could be an augmenting path. But if this were the case,

$$d_k(x_0) \leq d_k(z) + 1 \leq d_{k+1}(z) + 1 = d_{k+1}(x)$$

which implies $x_0 \notin A$ $(\perp)$. $s \dots zx$ is not f_k-a.p. This can only happen if $f_k(\overrightarrow{zx_0}) = c(\overrightarrow{zx_0})$ (the side is saturated). Since the side is used in a f_{k+1}-a.p. it must be the case that $f_{k+1}(\overrightarrow{zx_0}) < c(\overrightarrow{zx_0})$. This means $\overrightarrow{zx}$ was used backwards in the kth iteration, or rather that $f_k = s \dots \overleftarrow{x_0 z} \dots t$.

Since this is Edmond-Karp, augmenting paths are of minimal length. Then

$$\begin{aligned} d_k(z) &= d_k(x_0) + 1 \\ &> d_{k+1}(x_0) + 1 \\ &= d_{k+1}(z) + 2 \\ &\geq d_k(z) + 2 \end{aligned}$$

which is absurd.

1.3.2 If $\overrightarrow{xz}$ is a side then again

$$d_k(z) \leq d_{k+1}(z) < d_{k+1}(x_0) < \infty$$

Then there is an f_k-a.p. $s \dots z$ and, at least in principle, $s \dots \overrightarrow{zx}$ could also be augmenting. But if this were the case, we would have

$$d_k(x_0) = d_k(z) + 1 \leq d_{k+1}(z) + 1 = d_{k+1}(x_0)$$

which would imply $x_0 \notin A$ $(\perp)$. Then $s \dots \overrightarrow{zx_0}$ is not augmenting, which means the side $\overrightarrow{x_0 z}$ cannot be used backwards. This can only be true if $f_k(\overrightarrow{x_0 z} = 0$. But since the side is used backwards in the f_{k+1}-augmenting path, it must be used forward in this iteration. Which means $s \dots \overrightarrow{x_0 z} \dots t$ is augmenting. But then

$$\begin{aligned} d_k(z) &= d_k(x_0) + 1 \\ &> d_{k+1}(x_0) + 1 \\ &= d_{k+1}(z) + 2 \\ &\geq d_k(z) + 2 \end{aligned}$$

which is absurd.

Conclusion. In both cases a contradiction arises. The contradiction comes from assuming $A \neq \emptyset$. Then $A = \emptyset$. ■

4 Dinitz: Complexity

We will prove the complexity of Dinitz is $O(n^2m)$.

Recall that Dinitz builds succesive auxiliary networks with BFS, uses DFS to find blocking paths in them, and uses these paths to update the flow. Since the level of t augments in each succesive network, there are at most $O(n)$ auxiliary networks. So the complexity of Dinitz is

$$O(n)\ [\text{Comp. of building A.N.} + \text{Comp. of finding blocking flow in A.N.}]$$

The complexity of building the A.N. is the complexity of BFS, which is $O(m)$. The process of finding the blocking path varies is the Western and the original algorithms.

Original version. The original Dinitz algorithm enforced the following invariant: In any given auxiliary network, all vertices have edges connecting to the next level. This invariant implies that the DFS run always reaches t without need to backtrack. In consequence, the complexity of the DFS run is $O(n)$. The complexity of updating the flow is $O(n)$, and each time this happens at least one side is saturated and removed. Then there are $O(m)$ paths. $\therefore$ The complexity of finding the paths and updating the flow is $O(n \times m)$.

However, there is some extra cost associated to preserving this invariant. Each time a path is found, saturated sides are removed with a prunning operation. The prunning operation goes from the highest to the lowest levels in the network, checks for vertices whose exiting edge is saturated, and deletes them. This means that, for each vertex, a process of $O(1)$ complexity checks if it is saturated. This happens each time a path is found, so $O(n \times m)$ times. Once these sides have been detected, they and their neighbors are removed, which has complexity $O(d(x))$. Since this happens at worst for all vertices, the deleting operation is $O\left(\sum_{x \in V} d(x)\right) = O(m)$. The total cost of enforcing the invariant is then $O(n \times m) + O(m)$, which means the complexity of finding the blocking paths and updating the flow is $O(n \times m) + O(n \times m) + O(m) = O(n \times m)$. Then the complexity of Dinitz is $O(n^2m)$.

Western version. The Western version finds blocking paths and updates the flow inside a while loop with three clauses. The first clause is running BFS to advance; i.e. setting $x = s$ and iteratively making x be the first neighbor of x until t is reached. This clause executes as long as there is a neighbor and we call it A.

The second clause considers the case where x, the vertex we are traversing, has no neighbors in the next level. Then the algorithm backtracks to its predecessor, removes the side leading from it to x, and attempts A once more. We call this part R.

The last clause considers the case where t is reached. Here, the flow is augmented using the path found and all saturated sides are removed.

Thus, finding the blocking flow and updating the path can be modeled as a word $A \dots IA \dots RA \dots RA \dots I$; i.e. as a succession of words of the form $A \dots X$ with $X \in \{R, I\}$.

The process R has complexity $O(1)$ because it simply involves going backwards and deleting a side. The process A has complexity $O(1)$ because it simply involves moving forward, but at most $O(n)$ successions of A may occur. This means $O(A \dots A) = O(n)$. The process I has complexity $O(n)$ because the flow is updated across at most all vertices. Then $O(A \dots I) = O(n) + O(n) = O(n)$ and $O(A \dots R) = O(n) + O(1) = O(n)$. The question is how many words of the form $A \dots X$ exist. At worst, all paths are traversed in the process of finding t, so there are $O(m)$ words of this form. Then the complexity of the run is

$$O(m)\,[O(n) + O(n)] = O(nm)$$

Then the complexity of Dinitz is $O(n^2 m)$.

4.1 Wave: Complexity

We know the distance between s and t in auxiliary networks is increasing. The distance is bounded by n. $\therefore$ There are $O(n)$ auxiliary networks. Now, each auxiliary network is first constructed and then used to find a blocking flow. Then the complexity of Wave is

$$O(n)\,[F+B]$$

where F is the complexity of finding a blocking flow in an auxiliary network and B the complexity of building an auxiliary network. To build the auxiliary network we use BFS. $\therefore B = O(m)$. Let us examine F.

To find blocking flows we attempt to balance in forward and backward waves. When going forward, let V be the steps where a side is saturated, $\overline{V}$ those where a side is not saturated. When going backward, let S be the steps where a side is emptied, $\overline{S}$ those there a side is not emptied.

Complexity of V. Assume $\overrightarrow{xz}$ is a side and is saturated in a forward wave. To saturate again it must empty at least a little bit before. If it empties, then z was blocked and returned the flow to x; and since z is blocked, $\overrightarrow{xz}$ will never again be used. Then each side can saturate at most once. $\therefore$ the complexity of V is $O(m)$.

Complexity of S. Assume $\overrightarrow{zx}$ is a side and it empties in a backward wave. Since x returned flow, it is blocked and z will not send flow to x a gain. Then $\overrightarrow{zx}$ cannot be emptied ever again. $\therefore$ The complexity of S is $O(m)$.

Complexity of $\overline{V}$. When a vertex sends flow to its neighbors, it saturates all sides except perhaps one. Then, for any given vertex, each forward wave will send partial flow through at most one side. $\therefore$ The complexity of $\overline{V}$ is $O(n)\times\varphi$, with φ the number of forward waves.

Complexity of $\overline{S}$. When a vertex returns flow to its predecessors, it empties all sides except perhaps one. Then, for any given vertex in a backward wave, at most one side is partially emptied. $\therefore$ The complexity of $\overline{S}$ is $O(n) = \psi$ where ψ is the number of backward waves.

Now, it is trivialy to see $\varphi = \psi$ and there are at most n forward waves. $\therefore\ O(\varphi) = n$.

$\therefore$ The complexity of $\overline{S}, \overline{V}$ are both $O(n)\times O(n) = O(n^2)$.

$\therefore F = O(n^2) + O(n^2) + O(m) + O(m) = O(n^2) + O(m)$.

But $O(n^2) + O(m) = O(n^2)$, because $m \le \binom{n}{2} = O(n^2)$.

$\therefore F = O(n^2)$.

$\therefore$ The complexity of Dinitz is $O(n)\left[O(m) + O(n^2)\right] = O(n)O(n^2) = O(n^3)$

5 Codes

5.1 Hamming bound

Let $C \subseteq \{0,1\}^n$. We want to prove

$$|C| = \frac{2^n}{\sum_{i=0}^{t} \binom{n}{i}}$$

Let $A = \bigcup_{v \in C} D_t(v)$. Recall that $D_t(v)$ is the set of words in $\{0,1\}^n$ that are at a Hamming distance of t or less from v.

Let $S_v(r) = \{w \in \{0,1\}^n : d_H(v,w) = r\}$. Then it follows by definition that $D_t(v) = \bigcup_{r=0}^{t} S_v(r)$. Of course, this union is disjoint. It follows that

$$A = \bigcup_{v \in C} \bigcup_{r=0}^{t} S_v(r)$$

and

$$|A| = |C| \times \sum_{r=0}^{t} |S_v(r)|$$

So now we must only determine $|S_v(r)|$. But this is easy to do if we consider that any $w \in S_v(r)$ differs from v by exactly r bits, and is fully determined by this difference. In other words, there is a bijection between any $w \in S_v(r)$ and the set of the r bits out of all n possible bits that make up the difference between w and v. This readily entails that $|S_v(r)| = \binom{n}{r}$. This readily gives

$$|A| = |C| \times \sum_{r=0}^{t} \binom{n}{r}$$
$$\Rightarrow |C| = \frac{|A|}{\sum_{r=0}^{t} \binom{n}{r}|}$$

We do not know the cardinality of A, but since $A \subseteq \{0,1\}^n$ we know $|A| \leq 2^n$. Then

$$|C| \leq \frac{2^n}{\sum_{r=0}^{t} \binom{n}{r}}$$

5.2 $\delta(C) = \min\{j : \exists S \subseteq H_{*n} : |S| = j \wedge S$ is LD$\}$

Notation. I use H_{*n} to denote the set with the n columns of H. I use $H^{(i)}$ to denote the ith column of H.

Let $s = \min\{j : \exists S \subseteq H_{*n} : |S| = j \wedge S \text{ is LD}\}$. This implies there are s columns $H^{(j_1)}, \ldots, H^{(j_s)}$ s.t. $\sum x_i H^{(j_i)} = 0$ for $x_1, \ldots, x_s$ not all null.

(1) Let $w := \sum x_i e_{j_i}$ where e_k is the vector with all zeroes except at the kth coordinate. Since not all x_i are zeroes, $w \neq 0$. Now,

$$\begin{aligned} Hw^t &= H\left(x_1 e_{j_1} + \ldots + x_s e_{j_s}\right)^t \\ &= x_1 H e^t_{j_1} + \ldots + x_s H e^t_{j_s} \\ &= \sum x_i H^{(j_i)} && \left\{\text{Because } He^t_j = H^{(j)}\right\} \\ &= 0 \end{aligned}$$

Then $w \in Nu(H) = C$. But $|w| \leq s$ and $w \neq 0$. We know $\delta = \min\{|x| : x \in C, c \neq 0\}$.

$\therefore\ \delta \leq |w| \leq s$.

(2) Let $v \in C$ s.t. $\delta = |v|$. Then there are $i_1, \ldots, i_\delta$ s.t. $v = e_{i_1} + \ldots + e_{i_\delta}$. Since $v \in C, Hv^t = 0$, which using the same logic as before gives $\sum H^{(i_j)} = Hv^t = 0$.

This implies $\left\{H^{(i_1)}, \ldots, H^{(i_\delta)}\right\}$ is LD.

$\therefore s \leq \delta$.

(3) Points *(1)* and *(3)* imply $s = \delta$.

5.3 Three statements around a generating polynomial

Let C a code of length n and dimension k with generating polynomial $g(x)$. We will prove:

1. $C = \{p(x) : gr(p) < n \wedge g(x) \mid p(x)\} := C_1$
2. $C = \{v(x) \odot g(x) : v(x) \in F[x]\} := C_2$
3. $gr(g) = n - k$
4. $g(x) \mid (1 + x^n)$

(1 and 2) : We shall prove $C_1 \subseteq C_2 \subseteq C \subseteq C_1$.

Let $p(x) \in C_1$.Then there is some $q(x)$ s.t. $p(x) = g(x)q(x)$ and $gr\ (g(x)q(x)) < n$.

$\therefore\ g(x)q(x) = g(x) \odot q(x) \in C_2$.

$\therefore\ C_1 \subseteq C_2$.

Now let $p(x) = v(x) \odot g(x) \in C_2$ with $v(x)$ an arbitrary polynomial. Then

$$\begin{aligned} p(x) &= \left(v_0 + v_1 x + \ldots + v_{gr(v)} x^{gr(v)}\right) \odot g(x) \\ &= v_0 \odot g(x) + v_1\,(x \odot g(x)) + v_2\,(x_2 \odot g(x)) + \ldots + v_{gr(v)}\left(x^{gr(v)} \odot g(x)\right) \\ &= v_0 g(x) + v_1 Rot\,(g(x)) + v_2 Rot^2\,(g(x)) + \ldots + v_{gr(v)} Rot^{gr(v)}\,(g(x)) \end{aligned}$$

All rotations of $g(x)$ belong to C.

$\therefore p(x) \in C$.

$\therefore C_2 \subseteq C$.

Now let $p(x) \in C$. By definition, $gr(p) < n$, which implies $p(x) = p(x) \mod (1+x^n)$. We know

$$p(x) = g(x)q(x) + r(x)$$

for some $q(x), r(x)$ s.t. $gr(r) < gr(g)$. Then

$$\begin{aligned} p(x) &= (g(x)q(x) + r(x)) \mod (1 + x^n) \\ &= g(x) \odot q(x) + (r(x) \mod (1 + x^n)) \\ &= g(x) \odot q(x) + r(x) && \{\text{Since } gr(r) < gr(g) < n\} \end{aligned}$$

$\therefore r(x) = p(x) + g(x) \odot q(x)$.

We know $p \in C$ and $g(x) \odot q(x) \in C^2 \subseteq C$.

$\therefore r(x) \in C$.

But since g is generating polynomial, it is the unique polynomial with the least non-null degree in C.

$\therefore\ gr(r) < gr(g) \Rightarrow r(x) = 0.$

$\therefore g(x) \mid p(x)$ and then $p(x) \in C_1$.

$\therefore C \subseteq C_1$

(3) Let $p(x) \in C$. Then there is $q(x)$ s.t. $p(x) = g(x)q(x)$ with $n > gr(p) = gr(g) + gr(q)$. This readily implies $gr(q) < n - gr(g) < n$. Then $g(x)q(x) \in C$.

This entails there is a bijection between C and the set of polynomials of degree $n - gr(g)$. Then

$$\begin{aligned}
|C| &= |\{p(x) : gr(p) < n - gr(g)\}| \\
\iff\ 2^k &= 2^{n-gr(g)} \\
\iff\ k &= n - gr(g) \\
\iff\ gr(g) &= n - k \ \blacksquare
\end{aligned}$$

(4) Divide $(1 + x^n)$ by $g(x)$ to obtain

$$1 + x^n = g(x)q(x) + r(x)$$

with $gr(r) < gr(g)$. Taking the modulus,

$$\begin{aligned}
0 &= (1 + x^n) \mod (1 + x^n) \\
&= (g(x)q(x) + r(x)) \mod (1 + x^n) \\
&= (g(x) \odot q(x)) + (r(x) \mod 1 + x^n) \\
g(x) \odot q(x) &= r(x)
\end{aligned}$$

because $gr(r) < gr(g) < n$.

$\therefore\ r(x) = g(x) \odot q(x) \in C$.

But $gr(r) < gr(g)$. $\therefore r(x) = 0$ and $g(x) \mid (1 + x^n)$.

6 Matchings

6.1 Konig

We want to prove that any bipartite and regular graph $G = (V, E)$ has a perfect matching. Let X, Y be the two parts of G. For any $W \subseteq V$ let $E_W := \{wu \in E : w \in W\}$.

(1) Let $S \subseteq X$ and $l \in E_S$. It follows that

$$\exists x \in S, y \in Y : l = xy = yx$$

$\therefore$ $y \in \Gamma(x)$. And since $x \in S$ we have $y \in \Gamma(S)$ and $l \in E_{\Gamma(S)}$.

$\therefore$ $E_S \subseteq E_{\Gamma(S)}$ and $|E_S| \le |E_{\Gamma(S)}|$.

(2) Let us calculate $|E_W|$ when $W \subseteq X$.

Observe that $E_W = \bigcup_{w \in W} \{wv : v \in \Gamma(w)\}$. Furthermore, the union is disjoint, because $wv \in E_W \Rightarrow w \in X \Rightarrow v \in Y$. Then

$$|E_W| = \sum_{w \in W} |\Gamma(w)| = \sum_{w \in W} d(w)$$

Since G is regular, $d(w) = \delta = \Delta$.

$\therefore$ $|E_W| = \Delta |W|$

(3) Using what we established in *(1)*, it follows from *(2)* that

$$|S|\Delta \le |\Gamma(S)|\Delta \Rightarrow |S| \le |\Gamma(S)|$$

This holds for any $S \subseteq X$. Then Hall's theorem implies there is a complete matching from X to Y. To prove it is perfect, we must prove $|X| = |Y|$.

But since X, Y are the two parts of G, $E = E_X = E_Y$. Then $|E_X| = |E_Y|$, which implies $|X|\Delta = |Y|\delta \Rightarrow |X| = |Y|$.

Alternatively, since there is a complete matching from X to Y, $|X| \le |Y|$. But the choice of X over Y was arbitrary, and then the same holds for Y. Then $|X| = |Y|$.

In both caes the matching is perfect.

6.2 Hall

Let $G = (V, E)$ a bipartite graph with parts X and Y, and let $Z \in \{X, Y\}$. We want to prove that there is a complete matching from X to Y iff $\forall S \subseteq Z : |S| \leq |\Gamma(S)|$.

($\Rightarrow$) The proof is trivial, because if such matching exists, it induces an injective function $f : X \to Y$ s.t. $f(x) \in \Gamma(x)$. Since it is an injection, $|f(S)| = |S|$ for any S. Then $f(S) \subseteq \Gamma(S) \Rightarrow |S| \leq |\Gamma(S)|$.

($\Leftarrow$) Assume the Hall condition $|S| \leq |\Gamma(S)|$ holds. Assume that, after running the algorithm to find a maximal matching, an incomplete matching is found. We will build $S \subseteq X$ that violates our assumption (we could use $S \subseteq Y$ without loss of generality).

(1) Let S_0 be the set of rows unmatched and $T_1 = \Gamma(S_0)$. Observe that, by assumption, $S_0 \neq \emptyset$, and all columns in T_1 have a match that is not in S_0. Let S_1 the set of rows matching columns of T_1 and $T_2 = \Gamma(S_1) - T_1$. Generally,

$$S_i = \text{Rows matching with } T_i$$
$$T_{i+1} = \Gamma(S_i) - \bigcup_{j=0}^{j=i} T_j$$

The algorithm stops only when it is revising a row and this row has no available neighbors; this is, it only stops passing from a S_i to a T_{i+1} when $T_{i+1} = \emptyset$. Furthermore, since each column only labels a single row (that of its match), and T_i "creates" S_i, we have $|S_j| = |T_j|$.

Define $S = \bigcup S_i, T = \bigcup T_i$, and note that all the S_i are disjoint and all the T_i are disjoint. Then

$$\begin{aligned} |S| &= \sum |S_i| \\ &= |S_0| + \sum |T_i| \\ &= |S_0| + |T| \end{aligned}$$

$\therefore$ $|S| > |T|$ (since $S_0 \neq \emptyset$).

We must only prove now that $T = \Gamma(S)$.

(1) T are the labeled columns, and each column is labeled by a row in S. Each row only labels its neighbors. This implies $T \subset \Gamma(S)$.

(2) Assume $y \in \Gamma(S)$ and $y \notin T$. Then y was not labeled. But since $y \in \Gamma(S)$ there is an $x \in S$ s.t. $y \in \Gamma(x)$. Then each time the algorithm passes through x it should label y, which contradicts the fact that y is not labeled. Then $y \in T$. Then $\Gamma(S) \subseteq T$.

$\therefore$ $\Gamma(S) = T$ and $|S| > |\Gamma(S)|$. But this contradicts the hypothesis that the Hall condition holds. The contradiction comes from assuming there wasn't a complete matching. $\therefore$ There is a complete matching. ■

7 P-NP

7.1 2-Color is polynomial

To prove 2-color is polynomial, we must provide an algorithm $\mathcal{A}$ that correctly decides whether any given $G = (V, E)$ is 2-colorable in polynomial time. We will first provide $\mathcal{A}$ and then show its correctness and its belonging to P.

The algorithm takes an arbitrary non-colored vertex to be the root of its connected component and colors it with 1. Within each connected component, it runs BFS from the given root to explore it. Each time BFS pivots over a vertex $p \in V$ and enqueues its neighbors, the algorithm also colors each neighbor with $3 - c(p)$, thus ensuring that all colors are in the range $\{1, 2\}$.

It is important to note that the color of any given vertex is fully determined by the parity of its level in the BFS tree. Since the root at level zero is set to 1, all vertices in the second level are colored with 2, those in the third with 1, and so on.

```
$j := 0$
**while** $j < n$ **do**
    $r$ = arbitrary non-colored vertex
    $c(r) = 1$
    $j = j + 1$
    $queue = \{r\}$
    **while** $queue \neq \emptyset$ **do**
        $p = pop(queue)$
        **for** $x \in \Gamma(p)$ **do**
            **if** $x$ not colored **do**
                $c(x) = 3 - c(p)$
                $j = j + 1$
                $push(queue, x)$
            **fi**
        **od**
    **od**
**od**
**for** $\{xy\} \in E$ **do**
    **if** $c(x) = c(y)$ **do** *return* **False fi**
**od**
*return* **True**
```

The inner while runs BFS with a slight modification to color vertices when they are enqueued and is thus $O(m)$. It is executed per each connected component and the number of connected components is $O(n)$. $\therefore$ The algorithm is polynomial.

That the algorithm correctly decides that a graph is two-colorable is trivial. To prove that it also correctly decides that a graph is not two-colorable, we shall prove a negative answer entails the graph contains an odd cycle.

Assume the algorithm was executed over $G = (V, E)$ and returned **False**. Then there is some $\overrightarrow{xy} \in E$, in a particular connected component $C \subseteq G$, s.t. $c(x) = c(y)$. Let us presume, without loss of generality, that x was enqueued before y. Let us denote with r the root of C from which BFS was ran.

Assume x enqueues y. Then $c(y) = 3 - c(x) \neq c(x)$, a contradiction. Then x does not enqueue y. But this can only happen if, when x is the at front of the queue, y was already enqueued by some other vertex.

In particular, there is a vertex w that is the vertex of greater level common to x and y in the BFS tree—i.e. the vertex from which x and y diverge—. Let $\eta(w)$ be the level of w in the BFS tree.

Consider the cycle $w \dots xy \dots w$, which exists because all these vertices belong to C. There are $\eta(x) - \eta(w)$ edges from w to x, and $\eta(y) - \eta(w)$ edges from y to w. There is one extra edge for xy. The total amount is then

$$\eta(x) - \eta(w) + \eta(y) - \eta(w) + 1 = \eta(x) + \eta(y) - 2\eta(w) + 1$$

By assumption, $\eta(x)$ and $\eta(y)$ are both greater than $\eta(w)$. $\therefore \eta(x) + \eta(y) > 2\eta(w)$ and length of the path is greater than zero (sanity check).

Since $c(x) = c(y)$, $\eta(x) \equiv \eta(y) \mod 2$ and therefore $\eta(x) + \eta(y)$ is even. Then the length of the cycle is odd.

$\therefore C_{2k+1} \subseteq C$ and $\chi(G) \geq 3$.

7.2 3SAT es NP-Completo

.

Let $B = B_1 \wedge \ldots B_m$ an instance of SAT with variables $x_1, \ldots, x_n$. We build an instance of 3-SAT by transforming each B_i into an E_i as follows:

Complete.

$$E_i = (e_1 \vee e_2 \vee y_1) \wedge (\overline{y_1} \vee y_2 \vee e_3) \wedge (\overline{y_2} \vee y_3 \vee e_4) \vee \ldots (\overline{y_{k-3}} \vee e_{k-1} \vee e_k)$$

We want to prove

$$\exists \overrightarrow{b} : B(\overrightarrow{b}) = 1 \iff \exists \overrightarrow{\alpha} : \tilde{B}(\overrightarrow{b}, \overrightarrow{\alpha}) = 1$$

($\Leftarrow$) Asume $B(\overrightarrow{b}) = 0$. Then $D_i(\overrightarrow{b}) = 0$ for some i. Let $e_1, \ldots, e_k$ be the literals in D_i.

If $k = 3$ a contradiction ensues trivially. If $k = 2$, then $D_i = e_1 \vee e_2$ and then $E_i = (e_1 \vee e_2 \vee y_1) \wedge (e_1 \vee e_2 \vee \overline{y_1})$. Since $D_i = 0$, $e_1 \vee e_2 = 0$ and therefore $e_1 = e_2 = 0$ From this follows $E_i = y_1 \wedge \overline{y_1} = 1$. $(\perp)$

If $k = 1$ then $e_1 = 0$ and therefore $E_i = (y_1 \vee y_2) \wedge (y_1 \vee \overline{y_2}) \wedge (\overline{y_1} \vee y_2) \wedge (\overline{y_1} \vee \overline{y_2}) = 0$. But by assumption $E_i = 1(\perp)$.

If $k \geq 4$ we must observe that, since $D_i(\overrightarrow{b}) = 0$, we have $e_1 = e_2 = \ldots = e_k = 0$. Then this literals are neutral elements in the disjunctions and can be ignored. Since $E_i(\overrightarrow{b}, \overrightarrow{\alpha}) = 1$, its first term is true; in other words, $e_1 \vee e_2 \vee y_1 = 1 \Rightarrow y_1 = 1$. In all the following cases (except the last), $E_i = \overline{y_{i-1}} \vee y_i$ must be true; this is, $y_i \Rightarrow y_{i+1}$ is true. But the last term is $\overline{y_{k-3}}$, which cannot be true because y_1 and $y_1 \Rightarrow y_2 \Rightarrow \ldots \Rightarrow y_{k-3}$. $(\perp)$

($\Rightarrow$) Assume $B(\overrightarrow{b}) = 1$. For $k = 1, k = 2$, define $y_i = 0$ for all i. $\therefore D_i(\overrightarrow{b}) = 1 \Rightarrow E_i(\overrightarrow{b}, \overrightarrow{\alpha}) = 1$. For $k = 3$ the result is trivial. Let us consider the case $k \geq 4$.

Since $D_i(\overrightarrow{b}) = 1$ is a true disjunction, at least one e_r is true under $\overrightarrow{b}$. Define the following assignment:

$$\begin{aligned} &y_1 = y_2 = \ldots = y_{r-2} = 1 \\ &y_i = 0 \text{ para todos los demás } i \end{aligned}$$

Then

$$
\begin{aligned}
E(\overrightarrow{b},\overrightarrow{\alpha}) = (e_1 \vee e_2 \vee y_1) &\qquad \{\text{True because } y_1 = 1\} \\
\wedge\, (\overline{y_1} \vee y_2 \vee e_3) &\qquad \{\text{True because } y_2 = 1\} \\
\vdots \qquad\qquad & \\
\wedge(\overline{y_{r-3}} \vee y_{r-2} \vee e_{r-1}) &\qquad \{\text{True because } y_{r-2} = 1\} \\
\wedge(\overline{y_{r-2}} \vee y_{r-1} \vee e_r) &\qquad \{\text{True because } e_r = 1\} \\
\wedge(\overline{y_{r-1}} \vee y_r \vee e_{r+1}) &\qquad \{\text{True because } y_{r-1} = 0\} \\
\vdots \qquad\qquad & \\
\wedge(\overline{y_{k-3}} \vee e_{k-1} \vee e_k) &\qquad \{\text{True because } y_{k-3} = 0\}
\end{aligned}
$$

$\therefore$ Our assignment makes $\tilde{B}$ true.

7.3 3-Color es NP-Completo

We shall prove 3-color is NP complete. In order to do this, we will prove 3-SAT $\leq_\rho$ 3-Color. In other words, given an instance of 3-SAT of the form

$$B = \bigwedge i = 1^m (l_{i1} \vee l_{i2} \vee l_{i3})$$

with each literal l_{ij} a case of the variables $x_1, \ldots, x_n$, we shall provide an effective procedure that constructs a special graph $\mathcal{G}$ s.t. $\mathcal{G}$ is 3-colorable iff B is satisfiable.

(1 : Building $\mathcal{G}$) We shall define $\mathcal{G}$ by parts; namely,

1. Two special vertices s and t that are connected.
2. n triangles, each connecting the vertices in $\{t, v_i, w_i : 1 \leq i \leq n\}$
3. m triangles formed by the vertices $\{b_{i1}, b_{i2}, b_{i3} : 1 \leq i \leq m\}$
4. A tip u_{ij} each connected to b_{ij} and s.

Now let us define

$$\psi(l_{ij}) = \begin{cases} v_k & l_{ij} = x_k \\ w_k & l_{ij} = \overline{x_k} \end{cases}$$

Then we also include in $\mathcal{G}$ the sides $\left\{u_{ij}\ \psi(l_{ij} : 1 \leq i \leq m, 1 \leq j \leq 3)\right\}$. In other words, we connect each tip u_{ij} to either v_k or w_k, depending on what the literal l_{ij} is.

This completes the construction of $\mathcal{G}$. Now we shall prove $\mathcal{G}$ is 3-colorable iff B is satisfiable.

(2 : Proving $\Rightarrow$) Assume $\mathcal{G}$ has a proper coloring of three colors or less. Since $\mathcal{G}$ contains triangles, it must be a coloring of exactly three colors. We shall define

$$\overrightarrow{b_k} = \begin{cases} 1 & c(v_k) = c(s) \\ 0 & c(v_k) \neq c(s) \end{cases}$$

and prove that $B(\overrightarrow{b}) = 1$. Proving this equates to proving there is at least one j in $\{1, 2, 3\}$ s.t. $l_{ij}(\overrightarrow{b}) = 1$ for any arbitrary i. To prove this, we shall take u_{ij} and analyze what is color entails about the truth assignment.

The triangle $\{b_{i1}, b_{i2}, b_{i3}\}$ must contain $c(t)$ at some b_{ij_0} fixed. Take u_{ij_0}. Note that $c(s) \neq c(u_{ij_0}) \neq c(t)$. And since $\psi(u_{ij_0})$ cannot have the color of t, it must be the case that $c\left(\psi\left(u_{ij_0}\right)\right) = c(s)$. Now consider these cases.

Case 1. If $\psi(u_{ij_0}) = v_k$, it follows that $l_{ij} = x_k$.j Then $c(v_k) = c(s) \Rightarrow \overrightarrow{b_k} = 1 \Rightarrow l_{ij}(\overrightarrow{b}) = 1$. $\therefore$ $B_i(\overrightarrow{b}) = 1$.

Case 2. If $\psi(u_{ij_0}) = w_k$ then $l_{ij} = \overline{x_k}$. Since $c(w_k) = c(s)$ in this case, $c(v_k) \neq c(s)$ and $\overrightarrow{b}_k = 0$. Then $l_{ij}(\overrightarrow{b}) = 1$. $\therefore$ $B_i(\overrightarrow{b}) = 1$.

In both cases, for an arbitrary i, the coloring of $\mathcal{G}$ allows us to define an assignment ve^3 that makes $B_i(\overrightarrow{b}) = 1$. Of course, this assignment is s.t. $B(\overrightarrow{b}) = 1$. ■

(3 : Proving $\Leftarrow$) Assume B is satisfiable by a boolean vector $\overrightarrow{b}$. Then for any given i in $[1, m]$ we have $B_i(\overrightarrow{b}) = 1$. Then $l_{ij_0}(\overrightarrow{b}) = 1$ for (at least) a fixed j_0, $1 \leq j_0 \leq 3$.

Let $C = \{0, 1, 2\}$ a set of colors and define $c(s) = 0, c(t) = 1$. Let

$$c(v_k) = \begin{cases} c(s) & \overrightarrow{b}_k = 1 \\ 2 & \overrightarrow{b_k} = 0 \end{cases} \qquad c(w_k) = \begin{cases} 2 & \overrightarrow{b_k} = 1 \\ c(s) & \overrightarrow{b_k} = 0 \end{cases}$$

Clearly, $\{s, t\}$ is properly colored and $\{t, v_i, w_i\}$ is properly colored. All that is left is to color the triangles with tips.

Let

$$c(u_{ij}) = \begin{cases} 2 & j = j_0 \\ c(t) & j \neq j_0 \end{cases}$$

Of course, each $\left\{u_{ij}, s\right\}$ is properly colored. But what about $\left\{u_{ij}, \psi(l_{ij})\right\}$? Well, there are two cases to consider.

If $j = j_0$, $c(u_{ij}) = 2$ and $l_{ij}(\overrightarrow{b}) = 1$. If $\psi(l_{ij}) = v_k$, this means $x_k(\overrightarrow{b}) = 1 \Rightarrow \overrightarrow{b_k} = 1$. Then v_k is colored with $c(s) \neq c(u_{ij})$ and the coloring is correct. If $\psi(l_{ij}) = w_k$, entailing that $l_{ij} = \overline{x_k}$, then $\overrightarrow{b}_k = 0$ necessarily, in which case $c(w_k) = c(s) \neq c(u_{ij})$.

If $j \neq j_0$, then $c(u_{ij}) = c(t)$. But $\psi(l_{ij}) \in \{v_k, w_k\}$ never takes the color of t, and the coloring is correct.

All that is left is to color the triangle $\{b_{i1}, b_{i2}, b_{i3}\}$. But this is trivial. Simply let $c(b_{ij_0}) = c(s)$, ensuring that $\left\{b_{ij_0}, u_{ij_0}\right\}$ are properly colored, and color the remaining two vertices with $c(t)$ and 2 in any order.

We have used $\overrightarrow{b}$ to define a 3-coloring of $\mathcal{G}$. ■

7.4 Trisexual marriage

Let

$$B = \bigwedge_{i=1}^{m} (l_{i1} \lor l_{i2} \lor l_{i3})$$

an instance of 3-SAT with variables $x_1, \ldots, x_n$. Let

$$\begin{aligned} i &\in \{1, \ldots, n\} \\ j &\in \{1, \ldots, m\} \\ r &\in \{1, 2, 3\} \\ k &\in \{1, \ldots, m(n-1)\} \end{aligned}$$

Let G be a tripartite 3-hypergraph constructed as follows.

- $X = \{a_{ij}\} \cup \{s_j\} \cup h_k$
- $Y = \{b_{ij}\} \cup \{t_j\} \cup \{g_k\}$
- $Z = \{u_{ij}, w_{ij}\}$

for all possible combinations of their indeces. It is easy to see

$$\begin{aligned} |X| &= nm + m + m(n-1) = 2nm \\ |Y| &= nm + m + m(n-1) = 2nm \\ |Z| &= nm + nm = 2nm \end{aligned}$$

This is, all parts of the hypergraph have the same number of vertices. For its edges, we first define

$$v_{jr} = \begin{cases} u_{ij} & l_{jr} = x_i \\ w_{ij} & l_{jr} = \overline{x_i} \end{cases}$$

Then we define $E = E_0 \cup \ldots \cup E_3$ with

$$\begin{aligned} E_0 &= \{a_{ij}, b_{ij}, u_{ij}\} \\ E_1 &= \{a_{i(j+1)}, b_{ij}, w_{ij}\} \\ E_2 &= \{h_k, g_k, u_{ij}\} \cup \{h_k, g_k, w_{ij}\} \\ E_3 &= \{s_j, t_j, v_{jr}\} \end{aligned}$$

for all possible combinations of their indeces.

We shall prove there is a perfect matching over the hypergraph iff B is satisfiable.

($\Rightarrow$) Assume there is a perfect matching M. Take an arbitrary i and a fixed index j_0 such that a_{ij_0} is in a side of M. Then this side must be a side $\{a_{ij_0}, b_{ij_0}, u_{ij_0}\}$ E_0.

Since M matching, its sides are all disjoint, which means b_{ij} cannot appear in any other side of M. But since M is complete, a_{ij_0+1} must appear in some side, from which follows $\{a_i(j_0+1), b_{i(j_0+1)}, u_{i(j_0+1)}\} \in E(M)$. Inductively, it follows

$$\forall j : \{a_{ij}, b_{ij}, u_{ij}\} \in E(M)$$

We call this **Case 0**.

An analogous analysis gives as a second case that

$$\forall j : \{a_{i(j+1)}, b_{ij}, u_{ij}\} \in E(M)$$

which we call **Case 1**. Since the a_{ij} vertices only appear in E_0, E_1, if **Case 1** is not the case, we must have **Case 1**, and vice-versa.

We define

$$\overrightarrow{b}_i = \begin{cases} 1 & \textbf{Case 1} \text{ holds for } i \\ 0 & \textbf{Case 0} \text{ holds for } i \end{cases}$$

Since we want to prove $B(\overrightarrow{b}) = 1$, suffices to prove that for any j, there is some r s.t. $l_{jr}(\overrightarrow{b}) = 1$. For any arbitrary j, since M is perfect, the verices t_j, s_j must belong to the matching. Then there is some r s.t. $\{s_j, t_j, v_{jr}\} \in E(M)$.

If $l_{jr} = x_i$, then $v_{jr} = u_{ij}$ and $\{s_j, t_j, u_{ij}\} \in E(M)$. Then $\{a_{ij}, b_{ij}, u_{ij}\} \notin E(M)$, so we are in **Case 1**. Then $l_{jr}(\overrightarrow{b}) = 1$.

If $l_{jr} = \overline{x_i}$, then $v_{jr} = w_{ij}$ and $\{s_j, t_j, w_{ij}\} \in E$. But then $\{a_{i(j+1)}, b_{ij}, w_{ij}\} \notin E(M)$, so we are in **Case 0**. Then $l_{jr}(\overrightarrow{b}) = 1$.

Then, the assignment we have provided makes B_i true for any B_i.