Sampling of Multiloop Proteins

Tejas
Supervised by : Prof. Frederic Cazals

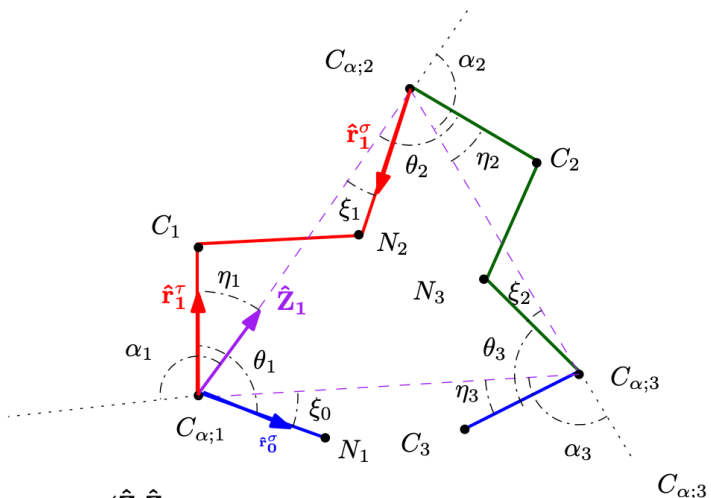
July 11, 2023

Problem

Background in graph theory

Loop sampling

- Instead of cartesian coordinates, we use internal coordinates, namely bond lengths, valence angles and torsion angles as the representation of protein molecules for analysis, because it is easier to express constraints in terms of internal coordinates.
- The objective is to generate sample torsion angles which lead to structurally valid random conformations of proteins, having multiple loops efficiently.
- A non-trivial theorem proved in 2004 says that if we consider a single tripeptide skeleton of 9 atoms and fix its legs, the remaining dihedral angles have atmost 16 solutions.



$\alpha_i = \angle \hat{\mathbf{Z}}_\mathbf{i} \hat{\mathbf{Z}}_{\mathbf{i-1}}$

$\xi_i = \angle - \hat{\mathbf{Z}}_\mathbf{i} \hat{\mathbf{r}}_\mathbf{i}^\sigma$

$\eta_i = \angle \hat{\mathbf{Z}}_\mathbf{i} \hat{\mathbf{r}}_\mathbf{i}^\tau$

θ_i: constrained valence angles

Nb: indices counted mod 3

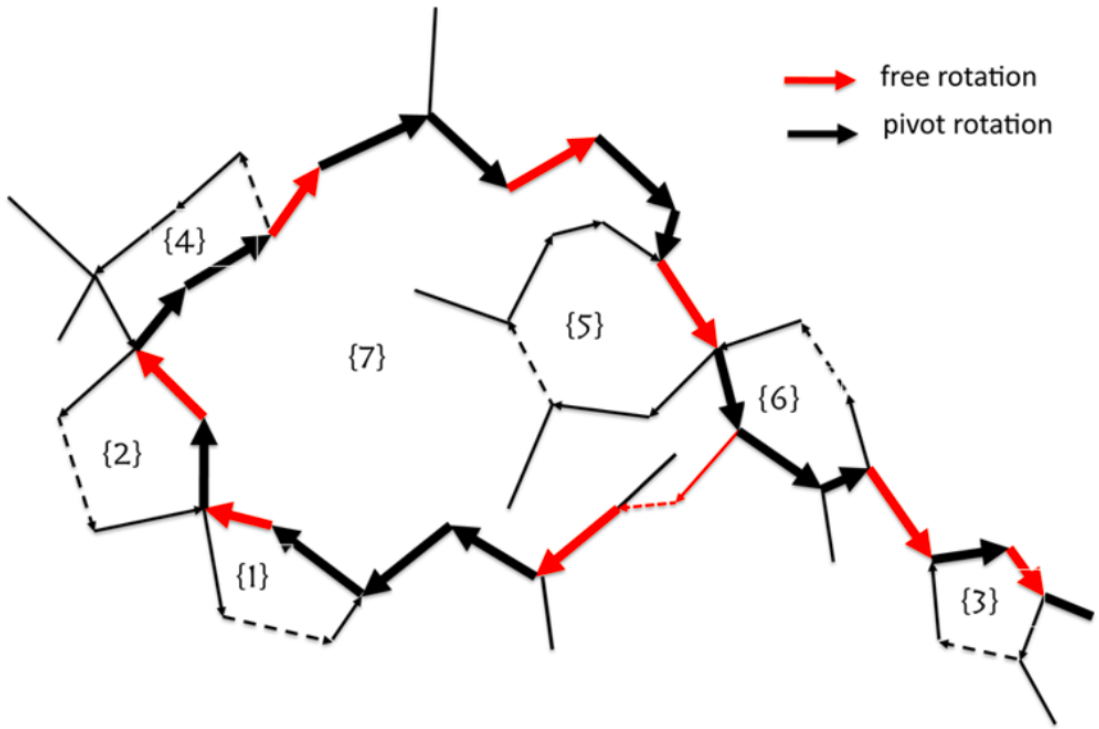
Multiloop Sampling

- In multiloop sampling, edges which are common between multiple cycles need to be sampled first as compared to independent edges otherwise the sampled loop would be structurally incorrect.
- The Loop closure problem is overconstrained for less than 6 Degrees of Freedom and underconstrained for more than 6 Degrees of Freedom.
- We need to cleverly think of an ordering of the loops in such a way that atleast 6 Degrees of Freedom are there for each Loop during the sampling procedure.

The algorithm by Coutsias et al

- The current multiloop sampling algorithm BRIKARD, explores conformational space in parallel, instead of exploring a single conformation at a time.
- In the preprocessing step, it generates a spanning tree and orders the rings in the sequence they are solved.
- While doing so , it must be maintained that each ring has atleast 6 torsional DOF that have not been set before.
- It is not gauranteed that the ordering of rings leads to a valid conformation.

The algorithm by Coutsias et al



▷Ref: Coutsias, Lexa, Wester, Pollock, Jacobson , JCTC 2016

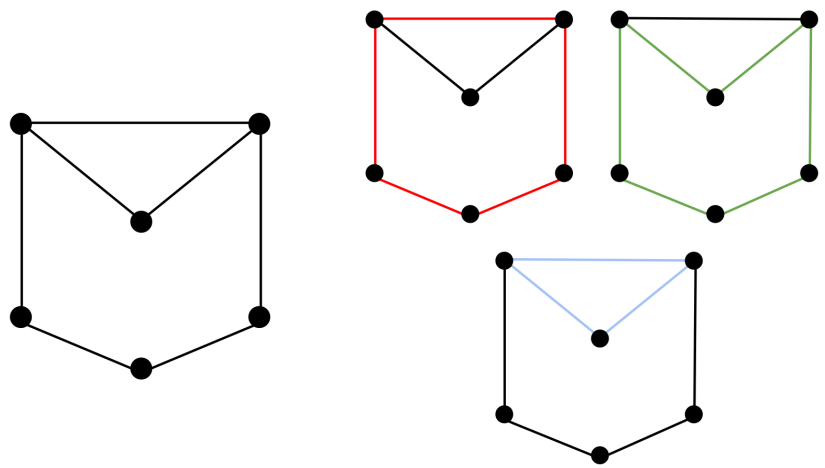
- The choice of rings may not be ideal, we can improve it by some optimization criteria.
- This may improve the running time by generating samples faster.

Problem

Background in graph theory

Cycles, Loops and Betti numbers

- A cycle is defined as any sequence of consecutive edges of a graph and Cycle Space $\mathcal{C}$ is defined as the set of all cycles of the graph.
- Sum of 2 Cycles A and B is defined as the symmetric difference of the 2 sets, i.e. cycle consisting of all the edges except the common edges.
- If we think of cycles as incidence vectors in $\{0,1\}^m$ the sum is equivalent to addition mod 2.
- For a connected graph the dimension of the cycle space, also known as the first betti number β_1 is equal to $m-(n-1)$



▷Ref: Wikipedia: https://en.wikipedia.org/wiki/Cycle_basis

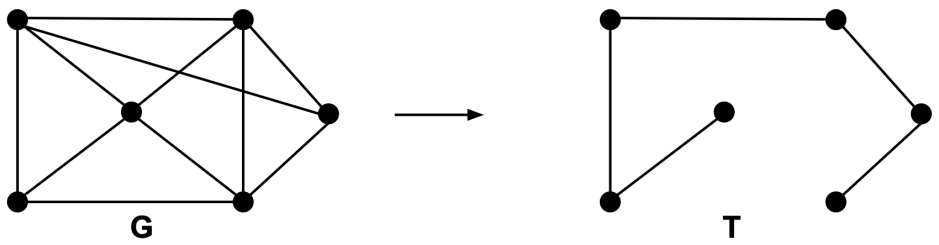
Finding a minimum cycle basis (Horton's Algorithm)

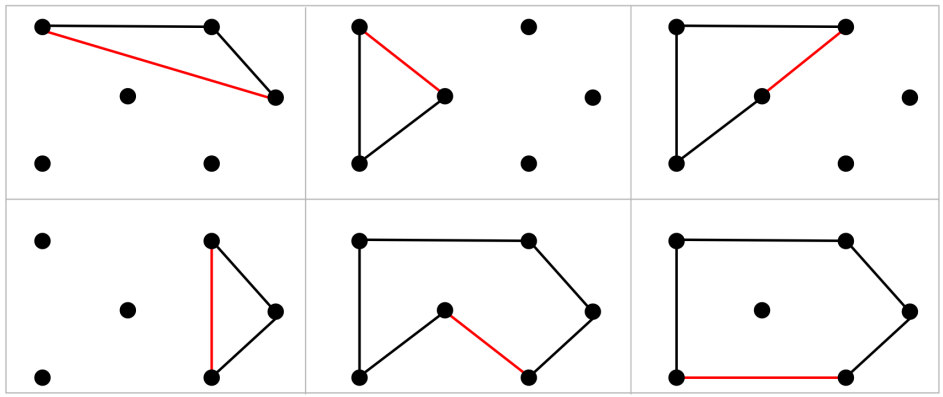
- Minimum cycle basis of a graph with a certain measure $\mu : E \to \mathbb{R}$,is defined as the set $\mathcal{B} = \{C_1, C_2, \ldots C_l\}$, which minimizes $\sum_i \mu(C_i)$ where $\mu(C_i) = \sum_{e \in C_i} \mu(e)$
- To find the minimum cycle basis of a graph we generate a large "candidate" set of cycles which is gauranteed to contain a basis.
- Fix a vertex p, generate its shortest distance spanning tree T_p.
- For every non-tree edge $e = (x, y)$, consider the cycle $C_e = T_p(p, x) + (x, y) + T_p(y, p)$
- Sort the set of cycles based on the measure, iteratively add the cycles with least weight after checking independence.
- Cycle Space forms a *matroid*, so this greedy strategy gives the correct solution
- By choosing better "candidate" sets, subsequent researchers have developed improved algorithms for this problem, reducing the worst-case time complexity for finding a minimum weight cycle basis in a graph with m edges and n vertices to $\mathcal{O}(m^2 n / log(n))$.

▷Ref: Horton, SIAM J. Computing, 1987

Spanning Tree Basis

We start with the spanning tree





Add non-tree edges to create the cycle basis

Matroids and greedy algorithms

- A matroid is a collection of independent sets, satisfying the properties
- Any subset of an Indpendent Set is also Independent.
- If an Independent set A has more elements than independent set B, we can create a bigger Indpendent set $B \cup \{x\}$ where $x \in A \setminus B$.
- Suppose the greedy algorithm picks elements $\{x_1, x_2, \ldots, x_l\}$ in order $(\mu(x_1) \leq \mu(x_2) \cdots \leq \mu(x_l))$, then for any Independent set $\{y_1, y_2, \ldots, y_l\}$ it holds that $\mu(x_i) \leq \mu(y_i)$ because otherwise let the first instance where this doesn't hold be k, so $\mu(x_k) > \mu(y_k)$, by the 2nd property of matroids, we can extend the independent set $\{x_1, x_2, \ldots, x_{k-1}\}$ using the set $\{y_1, y_2, \ldots, y_k\}$. So if we extend the set by some y_j where $j \leq k$ then the greedy algorithm would have chosen y_j because $\mu(y_j) \leq \mu(y_k) < \mu(x_k)$ and $\{x_1, x_2, \ldots, x_{k-1}, y_j\}$ is still independent. Which is a contradiction to the choice of the algorithm.
- We can see the Cycle Space is a matroid by defining independent sets as those sets which are linearly independent in the $\{0, 1\}^m$ representation of their cycles.

Our open problems

- After obtaining the set of independent loops, while sampling, torsions which are common between multiple loops are frozen.
- To use the loop closure equations, we need atleast 6 free torsions that have not been set before.
- We thus have the following decision problem at hand, to determine an ordering of L_i of the loops such that for each i, the set $T(L_i) \setminus \{t \in T(L_j) \text{ for some } j < i\}$ has atleast 6 elements. Where $T(L_i)$ is the set of torsions of loop L_i.
- To determine the set of loops itself, we need to consider the following optimization problem.
- The measure of an edge is now a function of the basis itself. We need to minimize $\sum_{e \in \mathcal{B}} \mu(e, \mathcal{B})$, where $\mu(e, \mathcal{B})$ is the multiplicity of that edge in the basis.

Implementing Cycle Basis algorithms

- To implement the minimum cycle basis finder, we use a slightly modified algorithm.
- We maintain a set of support vectors $\{S_j\}_{j>0}$ such that $\langle S_j, C_i \rangle = 0$ for all $j > i$. This helps ensure maintain while we add new elements into the basis.
- At i^{th} iteration of the algorithm, we choose C_i as the shortest cycle such that $\langle S_i, C_i \rangle = 1$, which is done via the signed graph construction.
- If $C_i = \sum_{k<i} \alpha_k C_k$ for some α_k, we would then have $\langle S_i, C_i \rangle = \sum_{k<i} \alpha_k \langle C_k, S_i \rangle = 0$. Hence we have independence of C_i
- We then update the support vectors to ensure that the invariant $\langle S_j, C_i \rangle = 0$ for all $j > i$ is satisified again

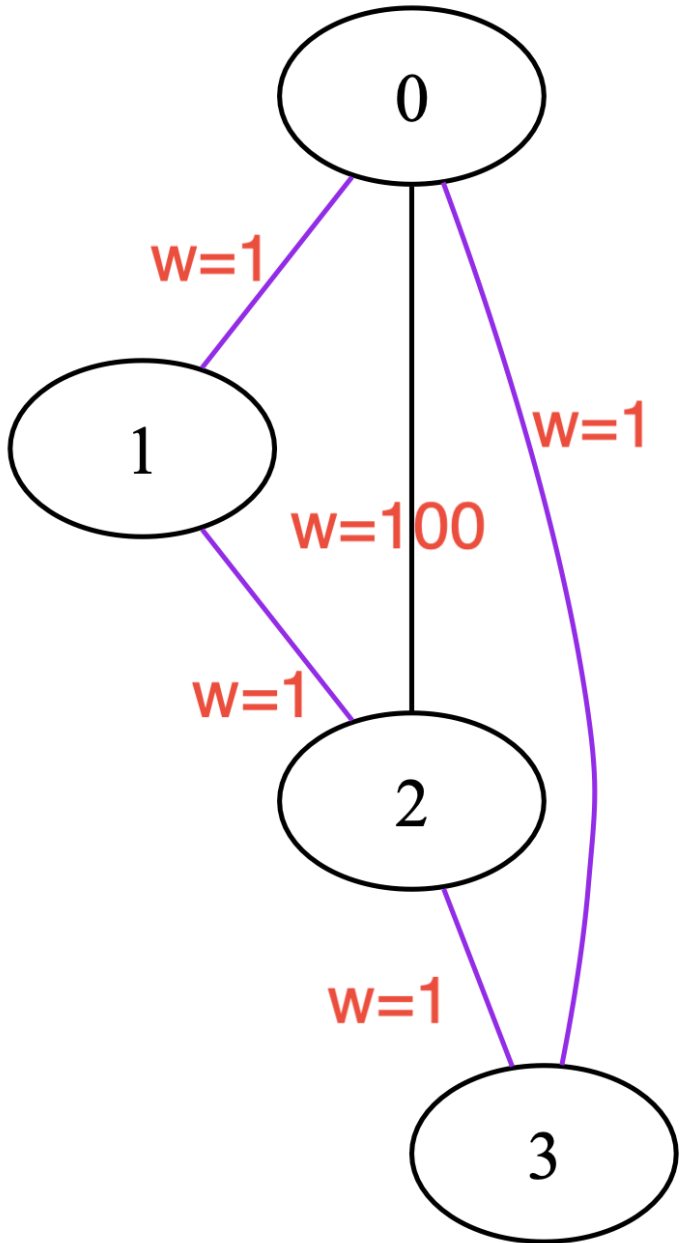
Signed Graph Construction

- To compute the shortest cycle such that $\langle S_i, C_i \rangle = 1$, we use the signed graph construction.
- For every node u in the original graph G, construct 2 nodes u^+ and u^- in the signed graph.
- For every edge $e = (u, v)$ in the original graph, if $e \in S_i$ then add edges (u^+, v^-) and (v^+, u^-). Otherwise, add edges (u^+, v^+) and (u^-, v^-) in the signed graph.
- Intuitively, we have split the graph into 2 signed levels and whenever we go from the $+$ side to the $-$ we take an edge from S_i to do so.
- Each path from $v-$ to $v+$ in the signed graph corresponds to a cycle in the original graph as $v+$ and $v-$ represent the same vertex.
- Since we are moving from the $-$ side to the $+$ side, there must be an odd number of common edges between S_i and C_i and hence $\langle S_i, C_i \rangle = 1$
- Thus to find the shortest such cycle we use dijkstra's algorithm to find the shortest path between all v^- and v^+.

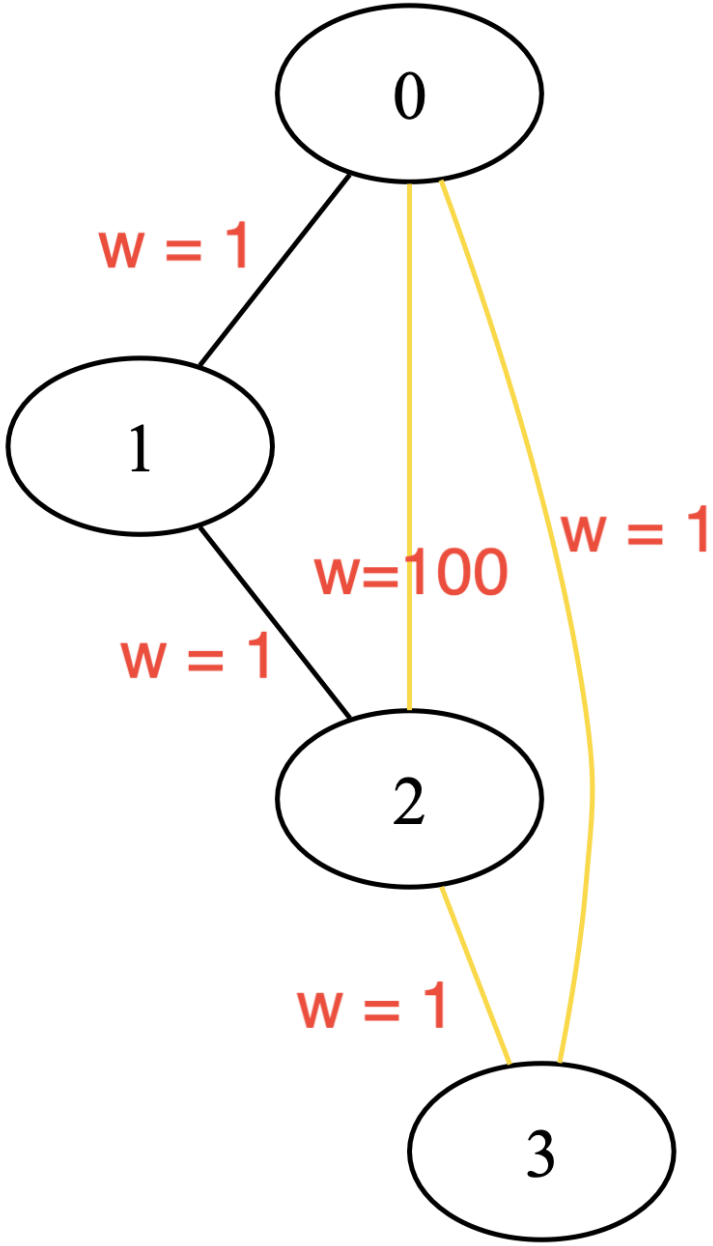
Code Structure

- The main classes used used in Implementing the minimum cycle basis finder are class T_Cycle, class T_Cycle_basis and class T_Minimum_Cycle_basis. All of them are templated classes with class GraphType as a parameter
- The class T_Cycle stores a cycle as a set of objects of type GraphEdge, m_cycle as the primary data member.
- The class T_Cycle_basis has a std::vector of T_Cycle objects, m_basis_elements as its primary data member.
- All of the classes have functions like add_edge(), begin(), end() etc.
- The primary class T_Minimum_cycle_basis implements the algorithm described above. It has functions like compute_signed_graph, shortest_indepndent_cycle for the main preprocessing step

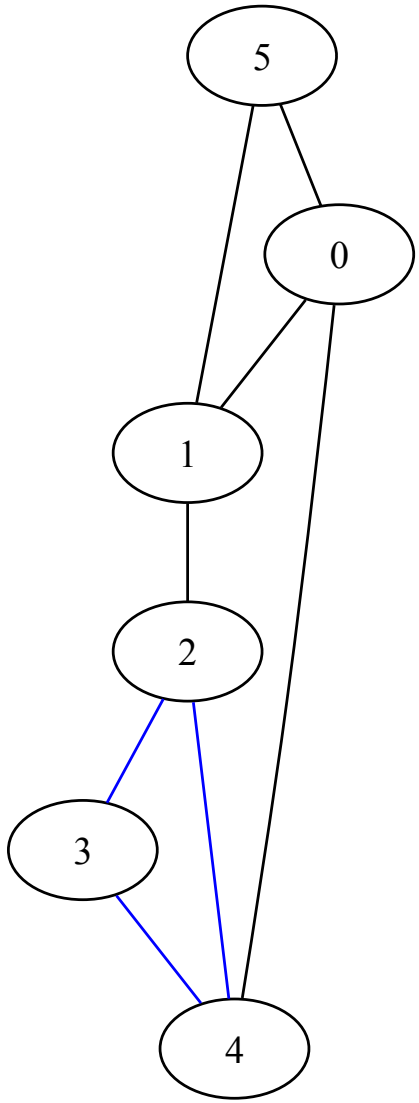
Some Examples

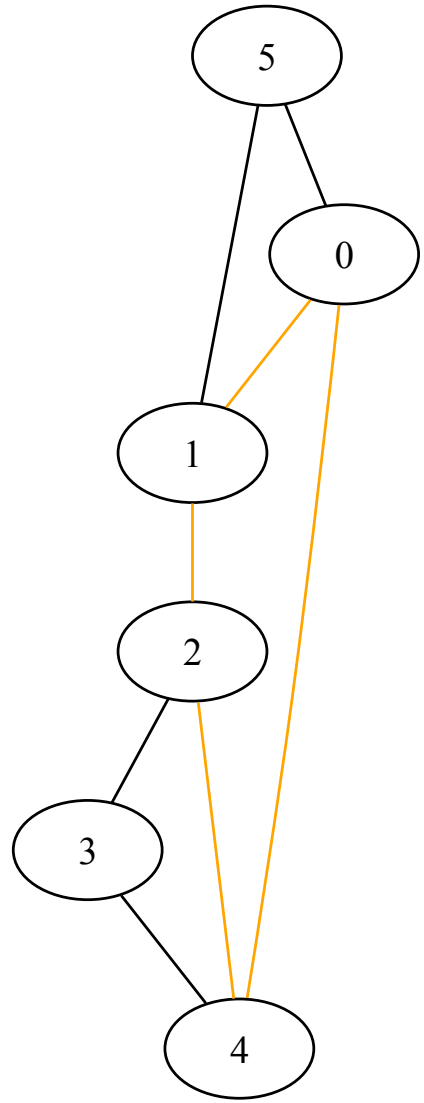


Some Examples

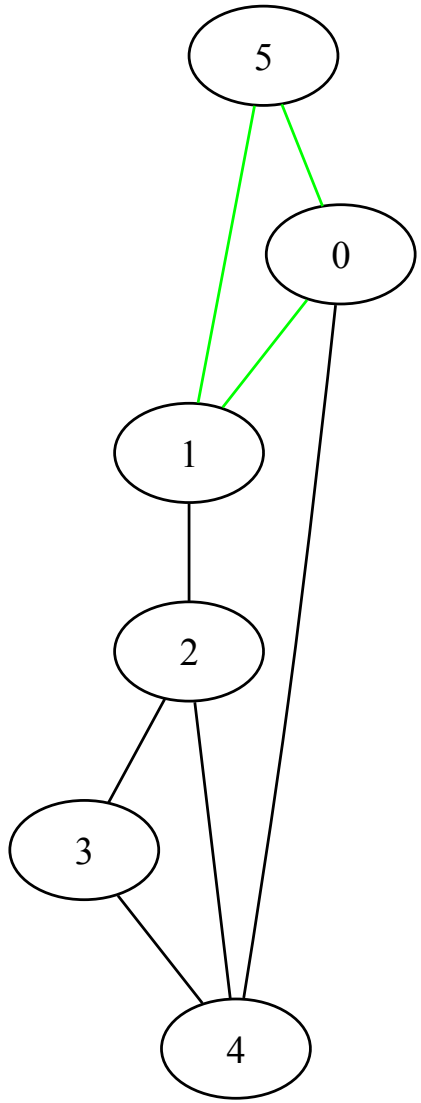


Some Examples





Some Examples



Thank You