1. ***Introduce yourself.*** Why do you take the course? What is your personal goal and expectations? What parts of topology might potentially be useful to you? How much time and energy do you plan to invest this term?

2. ***Fun with groups.*** A *group* $(\Sigma, \bullet)$ is a set Σ equipped with a binary relation $\bullet : \Sigma \times \Sigma \to \Sigma$, satisfying the following properties:

   - *Associativity*: for any three elements x, y, and z in Σ, one has $(x \bullet y) \bullet z = x \bullet (y \bullet z)$;
   - *Identity*: there is an element e in Σ such that $e \bullet x = x \bullet e = x$ for any x in Σ;
   - *Inverses*: for any x in Σ, there is an element x^{-1} in Σ such that $x^{-1} \bullet x = x \bullet x^{-1} = e$.

   A *group homomorphism* ϕ from group $(\Sigma, \bullet)$ to group $(\Pi, \circ)$ is a map $\phi : \Sigma \to \Pi$ respecting the group operations; that is, for every pair of elements x and y in Σ,

   $$\phi(x \bullet y) = \phi(x) \circ \phi(y).$$

   The *kernel* of a group homomorphism, denoted as $\ker \phi$, is the subset of elements in Σ that maps by ϕ to the identity e_Π of Π; in notation,

   $$\ker \phi := \{x \in \Sigma \colon \phi(x) = e_\Pi\}.$$

   Let G be an arbitrary graph. There are two natural objects associated with G:

   - The *vertex space* Σ_V, containing all possible subsets of vertices of G;
   - The *edge space* Σ_E, containing all subgraphs of G.

   Both the vertex space and the edge space can be made into a group under element-wise symmetric difference operation. (Verify this yourself!)

   (a) Prove that the map from Σ_E to Σ_V, taking a subgraph H of G and returns the subset of *odd-degree* vertices in H (that is, vertices that incident to an odd number of edges), is a group homomorphism. What is the kernel of this map?

   (b) Both the vertex and the edge space can be viewed as a *vector space*, where the basis is the set of vertices/edges, and the coefficients are in the two-element field $\mathbb{Z}_2$, where 1 means the element is presented and 0 if not.

   As a vector space over the two-element field $\mathbb{Z}_2$, what is the dimension of $\ker \phi$?

Groups and vector spaces will appear a few times in this class, so it would be helpful to equip yourself with basic knowledge about them. There is no need to go through a whole textbook on linear/abstract algebra; sufficient knowledge on the following items will be good enough:

- *definitions of groups, fields, vector spaces, and linear transformations*
- *subgroups, quotient groups, group homomorphisms, and first isomorphism theorem*
- *column and null spaces of a linear transformation, rank-nullity theorem*

3. ***Length-decreasing path.*** Let G be an arbitrary undirected graph, equipped with an *edge-length function* where each edge e has a non-negative length $\ell(e)$. (For sake of simplicity, you are free to assume all the edge lengths are different.)

   Design and analyze an efficient algorithm that, given two vertices s and t in G, compute a shortest path between s to t among those whose edge lengths are *monotonically decreasing*. (Or, correctly report that there are no such paths.) In other words, if the computed path from s to t uses the edges $e_1, \ldots, e_k$ in sequence, then

   $$\ell(e_1) > \cdots > \ell(e_k).$$

4. ***Winter is coming.*** A *bush* T is an unordered tree where each internal node has degree exactly 3 (and each leaf has degree 1). We refer to edges as *branches*, and both internal nodes and leaves as *nodes*.

   (a) Prove that there is a branch e^* in an n-node bush T such that each tree component of $T - e^*$ has at least $n/4$ nodes.

   Such a branch e^* is called *balanced*. We can *cut* a bush T into two by choosing a branch e^* and cut it in half. As a graph, this is equivalent to taking each component of $T - e^*$ and glue a copy of e^* back to where it was. Because we made a copy of e^* for each of the two components in $T - e^*$, the total number of branches and nodes in the final two bushes are exactly n and $n + 2$, respectively.

   Consider the following algorithm FIREWOOD.

```
FIREWOOD(T, r):
  if T has at most r nodes:
      return singleton set {T}
  find a balanced branch e* in T
  cut the bush T at e* into T1 and T2
  return FIREWOOD(T1, r) ∪ FIREWOOD(T2, r)
```

   Figure 1. Pseudocode for Firewood, cutting the original bush into roughly equal-sized chunks.

   (b) Prove that the number of bushes returned is at most $O(n/r)$ for any big enough r. *[Hint: Careful. Why does the algorithm even terminate?]*

   A branch is *trimmed* if it was ever cut during the execution of FIREWOOD. If we just execute the algorithm naïvely, some bushes returned might have up to $\Omega(r)$ trimmed branches.

   *(c) Design and analyze an algorithm so that every bush returned has $O(1)$ many trimmed branches, where the constant is independent to r.

Skills in algorithm design and analysis is a must for this course. You need the following:

- *reading pseudocode, correctness/time analysis*
- *induction, binary search, whatever-first search, spanning trees, shortest-path algorithms*
- *basic data structures like lists, queues, and binary search trees, amortization*