Sample Midterm 2 solutions

1. Let $\sum_3 = T^2 \# T^2 \# T^2$ be the connected sum of three copies of the 2-torus.

a) Draw a picture of $\sum_3$!

Source of the graphics: math.stackexchange.com

You can find a similar representation in Week 9's lecture notes, slide 13.

b) Compute the genus and the Euler characteristic of $\sum_3$!

The genus of a connected sum is additive: $g(T^2 \# T^2 \# T^2) = g(T^2) + g(T^2) + g(T^2)$. The genus of one torus is 1, because the genus is the number of "holes" of a surface, so $g(\sum_3) = 3 \cdot 1 = 3$ (it has 3 holes as well, so we wouldn't even need the additivity).

The Euler characteristic is related to the number of holes (genus g) on a surface by $\chi = 2 - 2g$ for orientable closed surfaces: $\chi = 2 - 2g = 2 - 2 \cdot 3 = -4$.

Other solution (more related to the lecture notes): $\chi = V - E + F$, where V, E, F are the number of vertices, edges, and faces in a triangulation. You can make a 3-holed torus from a 12-gon (think through, why!), with boundary word: $a_1 b_1 a_1^{-1} b_1^{-1} a_2 b_2 a_2^{-1} b_2^{-1} a_3 b_3 a_3^{-1} b_3^{-1}$. After gluing the corresponding edges, all vertices become one, edges identify in 6 pairs and the whole polygon becomes 1 face: $\chi = V - E + F = 1 - 6 + 1 = -4$.

2. Compute the Jones polynomial of the Hopf link.

We compute the Jones polynomial of the Hopf link using the Kauffman bracket and the definition from the lecture notes. Let D be the standard oriented Hopf link diagram with two positive crossings.

The Kauffman bracket satisfies the rules

$$\langle \bigcirc \rangle = 1,$$

$$\left\langle \times \right\rangle = A \left\langle \text{A-smoothing} \right\rangle + A^{-1} \left\langle \text{B-smoothing} \right\rangle,$$

and

$$\langle \bigcirc \sqcup L \rangle = (-A^2 - A^{-2})\langle L \rangle.$$

For the Hopf link there are two crossings, hence four smoothing states.

Step 1: Compute the Kauffman bracket

The contributions of the four states are:

- both A-smoothings:
$$A^2(-A^2 - A^{-2}),$$
- one A- and one B-smoothing: 1,
- the other mixed smoothing: 1,
- both B-smoothings:
$$A^{-2}(-A^2 - A^{-2}).$$

Therefore

$$\begin{aligned} \langle D \rangle &= A^2(-A^2 - A^{-2}) + 1 + 1 + A^{-2}(-A^2 - A^{-2}) \\ &= (-A^4 - 1) + 2 + (-1 - A^{-4}) \\ &= -A^4 - A^{-4}. \end{aligned}$$

Both crossings are positive; hence: $w(D) = 2$.

By definition: $V_L(t) = (-A^3)^{-w(D)}\langle D \rangle, \qquad A = t^{-1/4}.$

Thus: $V_{\text{Hopf}}(t) = (-A^3)^{-2}(-A^4 - A^{-4}) = A^{-6}(-A^4 - A^{-4}) = -A^{-2} - A^{-10}$

Substituting $A = t^{-1/4} \rightarrow$: $A^{-2} = t^{1/2},\ A^{-10} = t^{5/2}$.

Hence the Jones polynomial of the positively oriented Hopf link is

$$\boxed{V_{\text{Hopf}}(t) = -t^{1/2} - t^{5/2}}.$$

3. Compute the simplicial homology of the simplicial complex obtained from the 2-faces of the tetrahedron!

Almost the same as Week 10, exercise 4 (solutions).

4. (a) What is the crossing number of a knot? Give a Python/Sage function that can compute it.

(b) Give the Union method of the Union-Find (or Disjoint-Set) datastructure!

(a) The crossing number of a knot is the minimal number of crossings needed in any diagram representing the knot. It is a knot invariant.

Example:

$$c(\text{trefoil}) = 3.$$

In Sage/Python one can compute it as follows:

```
from sage.knots.all import Knot

def crossing_number(p, q):
    K = Knot(p, q)
    return K.dowker_notation().size()

# Example: trefoil knot 3_1
print(crossing_number(3,1))
```

Or directly using SnapPy/Sage knot tables:

```
K = Knot(3,1)
print(K)
```

(b) The Union-Find (Disjoint-Set) structure stores a partition of a set into disjoint subsets. The `Union(x,y)` operation merges the sets containing x and y.

Using union by rank:

```
parent = {}
rank = {}

def make_set(x):
```

```
    parent[x] = x
    rank[x] = 0

def find(x):
    if parent[x] != x:
        parent[x] = find(parent[x])   # path compression
    return parent[x]

def union(x, y):
    rx = find(x)
    ry = find(y)

    if rx == ry:
        return

    if rank[rx] < rank[ry]:
        parent[rx] = ry
    elif rank[rx] > rank[ry]:
        parent[ry] = rx
    else:
        parent[ry] = rx
        rank[rx] += 1
```

The operation runs in almost constant amortized time when combined with path compression.