Problem Set 0

Name: Christopher Jeff Sanchez

Problem 0-1.

$$\begin{aligned} A &= \left\{ i + \binom{5}{i} \mid i \in Z,\ 0 \le i \le 4 \right\} \\ &= \left\{ 0 + \binom{5}{0},\ 1 + \binom{5}{1},\ 2 + \binom{5}{2},\ 3 + \binom{5}{3},\ 4 + \binom{5}{4} \right\} \\ &= \{1, 6, 12, 13, 8\} \end{aligned} \tag{1}$$

$$\begin{aligned} B &= \{3i \mid i \in \{1, 2, 3, 4, 5\}\} \\ &= \{3, 6, 9, 12, 15\} \end{aligned} \tag{2}$$

(a) $A \cap B = \{6, 12\}$

(b) $|A \cup B| = 7$

(c) $|A - B| = 3$

Problem 0-2.

$$\begin{aligned} X &= \{\text{\# of heads in three coin flips}\} \\ &= \{\text{HHH, HHT, HTH, THH, TTH, THT, HTT, TTT}\} \\ &= \{3, 2, 2, 2, 1, 1, 1, 0\} \end{aligned} \tag{3}$$

$$\begin{aligned} Y &= \{\text{products of two six-sided dice}\} \\ &= \{1{*}1,\ 1{*}2,\ 2{*}1,\ ...\ 6{*}5,\ 6{*}6\} \\ &= \begin{matrix} \{1, & 2, & 3, & 4, & 5, & 6, \\ 2, & 4, & 6, & 8, & 10, & 12, \\ 3, & 6, & 9, & 12, & 15, & 18, \\ 4, & 8, & 12, & 16, & 20, & 24, \\ 5, & 10, & 15, & 20, & 25, & 30, \\ 6, & 12, & 18, & 24, & 30, & 36\}, \end{matrix} \end{aligned} \tag{4}$$

(a) $\mathrm{E}[X] = (3 \cdot 1\ +\ 2 \cdot 3\ +\ 1 \cdot 3)/8 = 12/8 = 1.5$

(b) $\mathrm{E}[Y] = 426/36 = 11.8\overline{33}$

(c) $\mathrm{E}[\{X + Y\}] = 438/44 = 9.9\overline{54}$

Problem 0-3.

$$A = 600/6 = 100 \tag{5}$$

$$B = 60 \bmod 42 = 17 \tag{6}$$

(a) $A \bmod 2 = 0,\ B \bmod 2 = 0,\ \therefore A \equiv B \pmod 2$

(b) $A \bmod 3 = 1,\ B \bmod 3 = 0,\ \therefore A \not\equiv B \pmod 3$

(c) $A \bmod 4 = 0,\ B \bmod 4 = 2,\ \therefore A \not\equiv B \pmod 4$

Problem 0-4. Prove **by induction** that $\sum_{i=1}^{n} i^3 = [\frac{n(n+1)}{2}]^2$, for any $n \geq 1$.

Proof. Let $P(n) : \sum_{i=1}^{n} i^3 = [\frac{n(n+1)}{2}]^2$.

Base case, $P(1)$:

$$\sum_{i=1}^{1} i^3 = 1 \tag{7}$$

$$\left[\frac{1(1+1)}{2}\right]^2 = 1 \tag{8}$$

Hence, base case is true. For the induction step, assuming $P(n)$ is true, we get $P(n+1)$:

$$\begin{aligned}
\sum_{i=1}^{n+1} i^3 &= (n+1)^3 + \sum_{i=1}^{n} i^3 && (9)\\
&= (n+1)^3 + \left[\frac{n(n+1)}{2}\right]^2 && (10)\\
&= \frac{4(n+1)^3 + n^2(n+1)^2}{4} = \frac{(4(n+1) + n^2)(n+1)^2}{4} && (11)\\
&= \frac{(n^2 + 4n + 1)(n+1)^2}{4} = \frac{(n+2)^2(n+1)^2}{4} && (12)\\
&= \left[\frac{(n+1)((n+1)+1)}{2}\right]^2 && (13)
\end{aligned}$$

∎

Problem 0-5. Prove **by induction** that every connected undirected graph $G = (V, E)$ for which $|E| = |V| - 1$ is acyclic.

Recall:

- A graph is connected if each pair of vertices has at least one connecting edge.
- An undirected graph simply means that all edges has no direction component.
- An acyclic graph is a graph that contains no cycles. When traversing the graph from vertex to vertex, no same vertex shall be visited twice.
- $|E| = |V| - 1$ means that there is one less edge as there are vertices.

Proof. Using induction on the number of vertices, $n = |V|$, in a graph $G = (V, E)$. Define $P(n)$ as: "Given that G is a connected undirected graph, if $|E| = |V| - 1$, then G is acyclic."

Base case, $P(1)$: $|V| = 1$, hence $|E| = 0$. A graph with only one vertex can't create a cycle since a cycle requires a nonempty seqyence of edges. Thus the base case holds.

For the induction step, we consider a "shrink-down, build-up" approach: [1] Consider an (n+1) vertex graph G' that is connected, undirected, and has $|V| = n + 1$, and $|E| = n$. We know that any connected graph has a spanning tree, and a leaf of that spanning tree can be removed such that we are left with a graph G that is still connected, undirected, and has $|V| = n$, and $|E| = n - 1$. If we assume $P(n)$, then G is acyclic. Now we see if we can get G' from G by adding a vertex.

Case 1: Connecting the vertex v_{n+1} to an edge in G. This is not possible since all edges in G are already connected to 2 vertices (since it is a connected graph with one less edge than vertices).

Case 2: Connecting the edge e_n to two vertices already in G. This is not possible since if $e_n = \{v_{n-1}, v_n\}$, then by case 1, there won't be any edge for v_{n+1} and it won't be a part of the connected graph G.

Case 3: Connecting the new vertex and edge together with a vertex in G. That is, $e_n = \{v_n, v_{n+1}\}$. Since G is said to be acyclic, we know that v_n doesn't belong to any prior cycle. For v_n and v_{n+1} to create a new cycle, there should be a path from v_n to v_{n+1} and back to v_n. Since v_{n+1} is only connected to e_n, then there is no path from v_{n+1} back to v_n without repeating the edge. Therefore, the two vertices don't create a new cycle.

By considering the three cases, we see that G remains ayclic.

■

[1] This is to avoid the "Build-up Error," where one assumes that every size n+1 graph with some property can be built-up from a size n graph with the same property. This assumption is not true by default. Note that the property in question is that of the antecedent, which is necessary to assume $P(n)$.

Problem 0-6. Submit your implementation to `alg.mit.edu`.

```python
def count_long_subarray(A):
    '''
    Input:  A     | Python Tuple of positive integers
    Output: count | number of longest increasing subarrays of A
    '''
    count = 0
    prev_a = 0
    current_subarray_size = 0
    max_subarray_size = 0

    for a in A:
        print("a =", a)

        if a > prev_a:  # while increasing
            # keep track of length of increasing-subarray
            current_subarray_size += 1
            print("subarray_size =", current_subarray_size)
            prev_a = a

            if current_subarray_size > max_subarray_size:
                max_subarray_size = current_subarray_size
                count = 1
                print(
                    "update max_subarray_size =",
                    max_subarray_size,
                    "reset count =",
                    count,
                )
            elif current_subarray_size == max_subarray_size:
                count += 1
                print("increment count =", count)

            continue

        # if not inreasing anymore, end of subarray
        prev_a = a  # the update in the 1st if-case won't be executed, update here
        current_subarray_size = 1  # 1 because current_subarray = [ prev_a ]
        print("reset current_subarray_size =", 1)

    print("COUNT =", count)
    return count
```