Problem Set 0

0pt

Name: Xinzhe Cui

Problem 0-1. $A = \{1, 6, 12, 13, 9\}$ $B = \{3, 6, 12, 15\}$

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

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

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

Problem 0-2.

(a) $E(X) = \frac{1}{2} \times 3 = \frac{2}{3}$

(b) $E(Y) = E(Y_1) \times E(Y_2) = (\frac{1}{6} \times \sum_{i=1}^{6} i)^2 = \frac{49}{4}$

(c) $E(X+Y) = E(X) + E(Y) = \frac{2}{3} + \frac{49}{4} = \frac{155}{12}$

Problem 0-3. $A = 100$ $B = 18$

(a) $A \equiv B \pmod 2$ $True$

(b) $A \equiv B \pmod 3$ $False$

(c) $A \equiv B \pmod 4$ $False$

Problem 0-4. We prove by induction that

$$\sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2.$$

Base case: When $n = 1$,

$$\sum_{i=1}^{1} i^3 = 1, \quad \left(\frac{1(1+1)}{2}\right)^2 = 1.$$

Thus the formula holds for $n = 1$.

Inductive step: Assume that

$$\sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2$$

holds for $n = k$. Then for $n = k+1$,

$$\sum_{i=1}^{k+1} i^3 = \sum_{i=1}^{k} i^3 + (k+1)^3 = \left(\frac{k(k+1)}{2}\right)^2 + (k+1)^3 = \left(\frac{(k+1)(k+2)}{2}\right)^2.$$

Therefore, by induction, the formula holds for all $n \geq 1$. □

Problem 0-5. Proof. We prove by induction on the number of vertices $|V|$.

Base case: When $|V| = 1$, the graph has no edges ($|E| = 0$). Clearly, it is acyclic.

Inductive hypothesis: Assume that for some $k \geq 1$, every connected undirected graph with $|V| = k$ and $|E| = k-1$ is acyclic.

Inductive step: Consider a connected graph $G = (V, E)$ with $|V| = k+1$ and $|E| = k$. Since $|E| = |V| - 1$, there must exist at least one vertex of degree 1 (a leaf). Remove this vertex v and its incident edge. The remaining graph has k vertices and $k-1$ edges, and it is still connected. By the inductive hypothesis, this smaller graph is acyclic. Adding back vertex v and its edge cannot create a cycle, because v had degree 1. Thus, G is acyclic.

Conclusion: By induction, every connected undirected graph G with $|E| = |V| - 1$ is acyclic.

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
    ##################
    # YOUR CODE HERE #
    ##################
    n = len(A)
    if n == 0:
        return 0

    max_len = 1        #
    count = 0          #    max_len
    curr = 1           #

    for i in range(1, n):
        if A[i] > A[i-1]:
            curr += 1
        else:
            #     i-1
            if curr > max_len:
                max_len = curr
                count = 1
            elif curr == max_len:
                count += 1
            curr = 1   #

    #
    if curr > max_len:
        max_len = curr
        count = 1
    elif curr == max_len:
        count += 1

    return count
```