Problem Set 0

Name: Your Name

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

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

(b) $|A \cup B| = |\{1, 3, 6, 9, 12, 13, 15\}| = 7$

(c) $|A - B| = |\{1, 9, 13\}| = 3$

Problem 0-2.

(a)

$$E[X] = \sum_{i=0}^{3} iP(\text{flip } i \text{ heads}) = 0\left(\frac{1}{8}\right) + 1\left(\frac{3}{8}\right) + 2\left(\frac{3}{8}\right) + 3\left(\frac{1}{8}\right) = \frac{3}{2}$$

(b) Let Z be the random variable representing the outcome of rolling one fair six-sided dice. Then $Y = X^2$, and by the properties of expectation,

$$E[Y] = E[Z^2] = E[(Z - E(Z)^2)] + E[Z]^2 = \text{Var}(Z) + E[Z]^2$$

Since Z is the discrete uniform distribution on $[1, 6]$,

$$E[Z] = \frac{1+6}{2} = \frac{7}{2}$$

$$\text{Var}(Z) = \frac{(6-1+1)^2 - 1}{12} = \frac{35}{12}$$

so

$$E[Y] = \frac{35}{12} + \left(\frac{7}{2}\right)^2 = \frac{91}{6}$$

(c) Using the linearity of expectation,

$$E[X + Y] = E[X] + E[Y] = \frac{3}{2} + \frac{91}{6} = \frac{50}{3}$$

Problem 0-3. $A = 600/6 = 100$ and $B = 60 \mod 42 = 18 \mod 42$, so

$$B = 18 + 42k$$

for some $k \in \mathbb{Z}$.

(a) Showing that $A \equiv B \mod 2$ is equivalent to showing that A and B are both even, which is true.

(b) $A \equiv 1 \mod 3$, and since 18 and 42 are both divisible by 3, $B \equiv 0 \mod 3$ so this is false.

(c) $A \equiv 0 \mod 4$, and $B \equiv 2 + 2k \mod 4$; the right hand side is 0 if and only if k is odd:

$\Rightarrow$: if $2 + 2k \equiv 0 \mod 4$, then for some $m \in \mathbb{Z}$, $2 + 2k = 4m$ which implies that $k = 2m - 1$, so k is odd.

$\Leftarrow$: if k is odd, then $k = 2m + 1$ for some $m \in \mathbb{Z}$, so

$$2 + 2k = 2 + 2(2m + 1) = 4m + 4 \equiv 0 \mod 4.$$

Problem 0-4.

Base case: for $n = 1$,

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

Inductive step: assume that

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

for $n \leq N$. Then

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

Combining terms on the right hand side,

$$(N+1)^3 + \frac{N^2(N+1)^2}{4} = \frac{4(N+1)^3 + N^2(N+1)^2}{4} = \frac{(4(N+1) + N^2)(N+1)^2}{4}$$

The term $4(N+1) + N^2$ is $(N+2)^2$, so this simplifies to

$$\frac{(N+2)^2(N+1)^2}{4} = \left[\frac{(N+1)(N+2)}{2}\right]^2$$

showing that for $n = N + 1$,

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

Problem 0-5. Induct on $|V|$:

Base case: suppose $|V| = 1$, so $|E| = 0$. Then G is the graph with one vertex and no edges, which is acylclic.

Inductive step: assume that every connected undirected graph $G = (V, E)$ with $|E| = |V| - 1$ is acyclic for $|V| \leq N$. Suppose that G is a connected undirected graph with $|V| = N + 1$ and $|E| = N$, and suppose that G contains a k-cycle for some $k \leq N + 1$. Note that k cannot equal $N + 1$, because by definition a k-cycle contains k vertices and k edges, and G has N edges, so it cannot contain an $N + 1$-cycle. Then there exists some $v \in V$ that is not part of this cycle. More specifically, we can find $v \in V$ that is not part of any cycle, and whose degree is 1: since $|E| = |V| - 1$, then

$$\sum_{v \in V} \deg v = 2|E| = 2(|V| - 1)$$

where $\deg v \geq 1$ for all $v \in V$ since G is connected. If $\deg v \geq 2$ for all $v \in V$, then the left hand side would be at least $2|V|$, which is a contradiction. Then there exists at least one $v' \in V$ whose degree is exactly 1. Additionally, we know that v' cannot be part of any cycle, because if it was its degree would be at least 2.

Let G' be the graph obtained by removing v' and its edge from G. Then G' is a connected, undirected graph with $|V| = N$, $|E| = N - 1$, containing a cycle, which is a contradiction. This shows that G cannot contain a cycle.

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

```
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 #
    ##################
    # track the current longest increasing subarray
    # and the length of the current increasing subarray
    longest_subarray_length = 0
    current_subarray_length = 0

    # iterate through the indices: for each i from 1 ... len(A) - 1,
    # if A[i - 1] < A[i], we're in an increasing subarray
    # increment the current subarray length by 1 (or 2 if it's 0)
    # compare to the longest subarray length: if it's >, update and set count to 1
    # if it's ==, increment count
    # if A[i - 1] >= A[i], then we're out of the subarray - set current length to 0
    for i in range(1, len(A)):
        if (A[i - 1] < A[i]):
            current_subarray_length += 2 if current_subarray_length == 0 else 1

            if (current_subarray_length == longest_subarray_length):
                count += 1
            elif (current_subarray_length > longest_subarray_length):
                longest_subarray_length = current_subarray_length
                count = 1
        else:
            current_subarray_length = 0
    return count
```