# USACO MAR12 Problem 'banner' Analysis

**by Nathan Pinsker**

The first step to solving this problem is to notice that we can quickly find all banners of a certain width and height. If a banner with width w and height h is allowed, then all other banners with width w and height h are allowed, and we can find the number of them using some simple arithmetic. It is also fairly straightforward to check if such a banner is allowed -- it is only if L*L <= (w*w + h*h) <= H*H and gcd(w, h) = 1.

Since the field can potentially be of size 100,000 x 100,000, even considering each possible size of the banner once will take too much time. However, if we consider each possible width that the banner can have, and if, given that width, we can quickly sum the allowed heights for the banner, then we will have solved the problem. For each width w, the sum that we want to obtain is equal to (all numbers between ceil(sqrt(L*L - w*w)) and floor(sqrt(H*H - w*w)) inclusive) - (all numbers in the same range that share a divisor with w). The first quantity is easy to obtain, while the second requires a little more work. To calculate the second quantity, we note that, although w could have a large variety of prime divisors, it does not have very many of them. This important insight allows us to quickly find the sum: we find the prime factors of w, then we use the inclusion-exclusion principle to calculate the sum of all numbers between L and H that are divisible by at least one of the numbers.

Below is Travis Hance's code.

```
#include <cstdio>

#define nmax 100005

typedef long long ll;

int prime_divs[nmax][6];
int num_prime_divs[nmax];

ll m,n,l,h,p;

inline ll sum(ll lo, ll hi, ll mul) {
        hi = hi / mul;
        lo = (lo + mul - 1) / mul;
        return ((hi - lo + 1) * (m + 1) - mul * ((hi*(hi+1) - (lo-1)*lo) / 2))
% p;
}

int main() {
        freopen("banner.in","r",stdin);
        freopen("banner.out","w",stdout);

        scanf("%lld", &m);
        scanf("%lld", &n);
        scanf("%lld", &l);
        scanf("%lld", &h);
```

```
	scanf("%lld", &p);

	for(int i = 1; i <= n; i++)
		num_prime_divs[i] = 0;
	for(int i = 2; i <= n; i++)
		if(num_prime_divs[i] == 0)
			for(int j = i; j <= n; j += i)
				prime_divs[j][num_prime_divs[j]++] = i;

	ll ans = 0;

	ll lo = l, hi = h;

	int minnh = (n < h ? n : h);
	for(ll w = 1; w <= minnh; w++) {
		while(lo > 1 && l*l - w*w <= (lo-1)*(lo-1))
			lo--;
		while(h*h - w*w < hi*hi)
			hi--;
		if(lo <= hi && lo <= m) {
			ll a = 0;
			int p2 = (1 << num_prime_divs[w]);
			for(int i = 0; i < p2; i++) {
				int i1 = i;
				ll prod = 1;
				int parity = 1;
				for(int j = 0; j < num_prime_divs[w]; j++) {
					if(i1 & 1) {
						prod *= prime_divs[w][j];
						parity *= -1;
					}
					i1 >>= 1;
				}
				a += parity * sum(lo, hi < m ? hi : m, prod);
			}
			ans = (ans + a*(n-w+1)) % p;
			if(ans < 0) ans += p;
			//printf("w = %lld, ans = %lld, lo = %lld, hi = 
%lld\n", w, ans, lo, hi);
		}
	}

	if(l <= 1 && 1 <= h)
		ans = (2*ans + n*(m+1) + m*(n+1)) % p;
	else
		ans = (2 * ans) % p;
	printf("%d\n", (int)ans);
}
```