# Counting Integer Compositions with Upper Bounds

Suppose we want the number of ordered $n$-tuples

$$(x_1, x_2, \dots, x_n)$$

of nonnegative integers satisfying

$$x_1 + x_2 + \cdots + x_n = S, \quad 0 \le x_i \le U (\forall i).$$

Denote this count by

$$F(n, S; U).$$

---

## 1. The "Unbounded" Baseline

If there were **no upper bound** on each $x_i$ (only $x_i \ge 0$), then the number of solutions to

$$x_1 + \cdots + x_n = S, \quad x_i \ge 0$$

is given by the stars-and-bars formula:

$$G(n, S) = \binom{n + S - 1}{S}.$$

---

## 2. Imposing $x_i \le U$ via Inclusion–Exclusion

We want to exclude any solution in which some $x_i > U$. Define

$$A_i = \{x : x_i \ge U + 1\}.$$

By inclusion–exclusion,

$$F(n, S; U) = |\{x : \sum x_i = S, x_i \ge 0\}| - |A_1 \cup \cdots \cup A_n|.$$

In expanded form:

$$F(n, S; U) = \sum_{r=0}^{n} (-1)^r \sum_{1 \le i_1 < \cdots < i_r \le n} |A_{i_1} \cap \cdots \cap A_{i_r}|.$$

If $r$ specific indices are forced to satisfy $x_{i_k} \ge U + 1$, shift each of those by $(U + 1)$. The total sum then becomes $S - r(U + 1)$, distributed freely among all $n$ variables:

$$|A_{i_1} \cap \cdots \cap A_{i_r}| = G(n, S - r(U + 1)) = \binom{n + (S - r(U + 1)) - 1}{S - r(U + 1)},$$

provided $S - r(U + 1) \ge 0$. Summing over all choices of $r$ variables ($\binom{n}{r}$ ways) yields the final closed-form.

---

## 3. Final Formula

$$\boxed{F(n,S;U) = \sum_{r=0}^{\lfloor S/(U+1)\rfloor} (-1)^r \binom{n}{r} \binom{n+(S-r(U+1))-1}{S-r(U+1)}.}$$

- If $S < 0$ or $S > nU$, then $F(n,S;U) = 0$.
- Otherwise, let $r_{\max} = \lfloor S/(U+1) \rfloor$.

---

## 4. Special Cases

1. **Unrestricted ($U = \infty$)**: Then $(U+1) > S$, so only $r = 0$ survives. We recover

$$F(n,S;\infty) = \binom{n+S-1}{S}.$$

2. **Binary Variables ($U = 1$)**: Each $x_i \in \{0,1\}$. We get

$$F(n,S;1) = \sum_{r=0}^{\lfloor S/2\rfloor} (-1)^r \binom{n}{r} \binom{n+(S-2r)-1}{S-2r}.$$

   A direct combinatorial argument shows this equals $\binom{n}{S}$.

3. **Ternary Bound ($U = 2$)**: Each $x_i \in \{0,1,2\}$. Then

$$F(n,S;2) = \sum_{r=0}^{\lfloor S/3\rfloor} (-1)^r \binom{n}{r} \binom{n+(S-3r)-1}{S-3r}.$$

   This was exactly the core subproblem when counting "7/8/9 slices summing to $K$."

---

## 5. Efficient Implementation

When $n$ and $S$ can be as large as $10^5$, we:

1. Precompute factorials $\text{fact}[i] = i! \bmod M$ for $i = 0 \ldots N_{\max}$,
2. Precompute inverse-factorials $\text{invFact}[i] = (i!)^{-1} \bmod M$ via

$$(i!)^{-1} = (i!)^{M-2} \bmod M, \quad M = 10^9 + 7,$$

   using fast exponentiation.
3. Then

$$\binom{n}{r} = \text{fact}[n] \times \text{invFact}[r] \times \text{invFact}[n-r] \bmod M,$$

   in $O(1)$ time per query.

Putting it all together:

```
const int MOD = 1000000007;
static const int MAXN = 200000; // big enough for n + S shifts

long long fact[MAXN+1], invFact[MAXN+1];

// (1) Fast exponentiation to compute a^p % MOD
long long modexp(long long a, long long p) {
  long long res = 1;
  while(p > 0) {
    if (p & 1) res = (res * a) % MOD;
    a = (a * a) % MOD;
    p >≥ 1;
  }
  return res;
}

// (2) Precompute factorials and inverse factorials
void initFactorials() {
  fact[0] = 1;
  for(int i = 1; i ≤ MAXN; i++) {
    fact[i] = (fact[i-1] * i) % MOD;
  }
  invFact[MAXN] = modexp(fact[MAXN], MOD - 2);
  for(int i = MAXN; i ≥ 1; i--) {
    invFact[i-1] = (invFact[i] * i) % MOD;
  }
}

// (3) Binomial coefficient nCr % MOD
long long nCr(int n, int r) {
  if (r < 0 || r > n) return 0;
  return ((fact[n] * invFact[r]) % MOD * invFact[n-r]) % MOD;
}

// (4) Inclusion-Exclusion formula for 0 ≤ xi ≤ U, sum = S
long long countBounded(int n, int S, int U) {
  // If S out of [0, nU], no solutions
  if (S < 0 || S > 1LL * n * U) return 0;
  int rmax = S / (U + 1);
  long long ans = 0;
  for(int r = 0; r ≤ rmax; r++) {
    // Choose which r variables exceed U
    long long choose = nCr(n, r);
    int rem = S - r * (U + 1);
    // Distribute rem among n without bound:
    long long ways = nCr( n + rem - 1, rem );
    long long term = (choose * ways) % MOD;
    if (r & 1) {
      term = (MOD - term) % MOD;  // subtract if r is odd
    }
    ans = (ans + term) % MOD;
  }
  return ans;
}

int main() {
  ios::sync_with_stdio(false);
  cin.tie(nullptr);

  initFactorials();

  int T;
  cin >> T;
  while(T--) {
    int n, S, U;
    cin >> n >> S >> U;
    cout << countBounded(n, S, U) << "\n";
  }
  return 0;
```

- **`initFactorials()`** populates `fact[i]` and `invFact[i]` up to `MAXN`.
- **`nCr(n,r)`** returns $\binom{n}{r}$ mod $10^9 + 7$.
- **`countBounded(n,S,U)`** implements

$$\sum_{r=0}^{\lfloor S/(U+1) \rfloor} (-1)^r \binom{n}{r} \binom{n + (S - r(U+1)) - 1}{S - r(U+1)}.$$

---

## 6. Key Takeaways

- **Stars & Bars** handles $x_i \geq 0$ with $\sum x_i = S \rightarrow \binom{n+S-1}{S}$.
- To enforce $x_i \leq U$, use **inclusion-exclusion**, subtracting solutions where any $x_i \geq U + 1$.
- The final formula is

$$F(n, S; U) = \sum_{r=0}^{\lfloor S/(U+1) \rfloor} (-1)^r \binom{n}{r} \binom{n + (S - r(U+1)) - 1}{S - r(U+1)}.$$

- Precompute factorials and inverse factorials modulo $10^9 + 7$ to answer each binomial in $O(1)$.