THINK FAST

Competitive Programming

From Problem 2 Solution in O(1)

Algebra

Summations

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

Recall: Patterns Session

- $F(n) = 1+2+3\ldots+n \Rightarrow (n * (n+1))/2$
  - $F(m, n) = m + m+1+\ldots.n =$ **$((n+m) * (n+1-m))/2$**
- $F(n) = \sum_i fx(i)$ and $fx(n) = 5 + n*3$
  - $F(n) = (n+1)(3n+10)/2$ => by evaluating and organizing
- $F(n) = 1 + 2 + 4 \ldots.\ 2^{n-1} = \sum_{k[0 - n-1]} 2^k = 2^n - 1$
- $F(n) = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots\ldots\ldots\ldots + 1/2^n = 1 - 1 / 2^n$
- $F(n) = \sum i\, F(i)$ => Compute $F(n) - F(n-1)$
- $$S_N = 1 + r + r^2 + r^3 + \ldots + r^N = \sum_{k=0}^{N} r^k = \frac{1 - r^{N+1}}{1 - r}$$

$$\sum_{i=0}^{n-1} i a^i = \frac{a - n a^n + (n-1) a^{n+1}}{(1-a)^2}$$

Power Sum

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

$$\sum_{k=1}^{n} k^2 = \frac{1}{6}\left(2n^3 + 3n^2 + n\right)$$

$$\sum_{k=1}^{n} k^3 = \frac{1}{4}\left(n^4 + 2n^3 + n^2\right)$$

$$\sum_{k=1}^{n} k^4 = \frac{1}{30}\left(6n^5 + 15n^4 + 10n^3 - n\right)$$

$$\sum_{k=1}^{n} k^5 = \frac{1}{12}\left(2n^6 + 6n^5 + 5n^4 - n^2\right)$$

$$\sum_{k=1}^{n} k^6 = \frac{1}{42}\left(6n^7 + 21n^6 + 21n^5 - 7n^3 + n\right)$$

$$\sum_{k=1}^{n} k = \frac{1}{2}n(n+1)$$

$$\sum_{k=1}^{n} k^2 = \frac{1}{6}n(n+1)(2n+1)$$

$$\sum_{k=1}^{n} k^3 = \frac{1}{4}n^2(n+1)^2$$

$$\sum_{k=1}^{n} k^4 = \frac{1}{30}n(n+1)(2n+1)\left(3n^2 + 3n - 1\right)$$

$$\sum_{k=1}^{n} k^5 = \frac{1}{12}n^2(n+1)^2\left(2n^2 + 2n - 1\right)$$

$$\sum_{k=1}^{n} k^6 = \frac{1}{42}n(n+1)(2n+1)\left(3n^4 + 6n^3 - 3n + 1\right)$$

Summation Laws

$$\sum_{k \in K} c a_k = c \sum_{k \in K} a_k$$

$$S_a = \sum_{k=1}^{20} (2 - 3k + 2k^2) = 2\sum_{k=1}^{20} 1 - 3\sum_{k=1}^{20} k + 2\sum_{k=1}^{20} k^2$$

$$S_a = 2(20) - 3\left(\frac{20(21)}{2}\right) + 2\left(\frac{(20)(21)(41)}{6}\right) = 5150$$

$$\sum_{k \in K} (a_k + b_k) = \sum_{k \in K} a_k + \sum_{k \in K} b_k$$

$$\sum_{k=10}^{50} k = \left(\sum_{k=1}^{50} k\right) - \left(\sum_{k=1}^{9} k\right)$$

$$\sum_{k \in K} a_k = \sum_{k \in K} a_{\pi(k)}$$

1 + 2 + 3 + 4 + 5 = 1 + 5 + 2 + 4 + 3 = **5 + 4 +3 + 2 + 1**

$\sum_{k = [0, n]} k = \sum_{k = [0, n]} (n-k)$ => replacement

Summations and Inequality

- Sometimes playing with summations start and end is confusing. Remember $\sum \Leftrightarrow$ inequality
- $\sum_{k = [0, n]} f(k) \Rightarrow \sum_{0 \le k \le n} f(k)$
- $\sum_{i = [0, n]} \sum_{j = [0, n]} f(i, j) \Rightarrow \sum_{0 \le i, j \le n} f(i, j)$
- $\sum_{i = [0, n]} \sum_{j = [i, n]} f(i, j) \Rightarrow \sum_{0 \le i \le j \le n} f(i, j)$
- And if some function is valid under some inequality, **convert** inequality to loop:
  - E.g. find $\sum F(k)$ IFF $5 <= F(k) <= X$
  - Answer = $\sum_{k = [0, n]} f(k)$ IFF $5 <= F(k)$ **<= X**
  - Answer = $\sum_{k = [0, n]} \sum_{g = [5, X]} f(k)$ IFF $F(k) == X$
- Let's remember more about inequality

Inequality

- $a \geq b$ and $b \geq c$, then $a \geq c$.
- $a \leq b$ and $b \leq c$, then $a \leq c$.
- if $a \geq b$ and $b > c$, then $a > c$
- if $a = b$ and $b > c$, then $a > c$
- $a \leq b$, then $a + c \leq b + c$ and $a - c \leq b - c$.
- $a \geq b$ and $c > 0$, then $ac \geq bc$ and $a/c \geq b/c$.
- $a \geq b$ and $c < 0$, then $ac \leq bc$ and $a/c \leq b/c$.
- If $a \leq b$, then $-a \geq -b$.
- $|a+b| \leq |a|+|b|$
- if a is integer, $a < 3$ $\Rightarrow$ $a \leq 2$

Inequality

- if $(a > 0 \text{ and } b > 0)$ or $(a < 0 \text{ and } b < 0)$
  - If $a \le b$, then $1/a \ge 1/b$.
  - If $a \ge b$, then $1/a \le 1/b$.
- if $(a > 0 \text{ and } b < 0)$ or $(a < 0 \text{ and } b > 0)$
  - If $a < b$, then $1/a < 1/b$.
  - If $a > b$, then $1/a > 1/b$.
- $\mathbf{1 \le k\text{-}j < k}$ $\quad$ $\mathbf{* \text{-}1}$ $\quad$ => $-1 \ge j-k > -k$ $\quad$ [Now j +ve]
  - $-1 \ge j-k > -k$ $\quad$ $+ k$ $\quad$ => $k-1 \ge j > 0$ $\quad$ [now j alone]
  - $k-1 \ge j > 0$ $\quad$ swap $\quad$ => $\mathbf{0 < j \le k-1}$ $\quad$ [better for a loop]
- $j < k+j \le n$ …. what constraints?
  - $j < k+j$ => $k > 0$
  - $k+j \le n$ => $k \le n-j$, $j \le n-k$

Inequality

- int percent = 70;
  - bool pass = result > (percent / 100.0); => doubles
  - bool pass = result * 100 > percent; => integers
- for(int i = 0; i < n/5; ++i)
  - for(int i = 0; i * 5 < n; ++i)
- for(int i = 0; i < vec.size()-1; ++i)
  - for(int i = 0; i < (int)vec.size()-1; ++i)
  - for(int i = 0; i+1 < vec.size(); ++i)
- for(int i = 0; i < sqrt(n); ++i)
  - for(int i = 0; i * i < n; ++i)

Inequality

- Check if results may overflow? More careful needed for other cases (e.g. ≥)

```
int a = 10, b = 20, c = 30, MAX = std::numeric_limits<int>::max();

if(a * b > MAX)
    return 0;

//Convert to
if(a > MAX / b)
    return 0;

// (a*b*c) > MAX ?
if(a > MAX / b || a * b > MAX / c)
    return 0;   // Check ab, then abc

// a * b + c?
if(a > MAX / b || a * b > MAX - c)
    return 0;   // Check ab, then abc
```

Replacements

- Doing replacement is trivial based on the inequality

$$
\begin{aligned}
S_n &= \sum_{1 \leqslant k \leqslant n} \sum_{1 \leqslant j < k} \frac{1}{k-j} && \text{summing first on } j \\
&= \sum_{1 \leqslant k \leqslant n} \sum_{1 \leqslant k-j < k} \frac{1}{j} && \text{replacing } j \text{ by } k - j \\
&= \sum_{1 \leqslant k \leqslant n} \sum_{0 < j \leqslant k-1} \frac{1}{j} && \text{simplifying the bounds on } j
\end{aligned}
$$

$$
\begin{aligned}
S_n &= \sum_{1 \leqslant j \leqslant n} \sum_{j < k \leqslant n} \frac{1}{k-j} && \text{summing first on } k \\
&= \sum_{1 \leqslant j \leqslant n} \sum_{j < k+j \leqslant n} \frac{1}{k} && \text{replacing } k \text{ by } k + j \\
&= \sum_{1 \leqslant j \leqslant n} \sum_{0 < k \leqslant n-j} \frac{1}{k} && \text{simplifying the bounds on } k
\end{aligned}
$$

Order of Summations

- Given 3 lists A, B, C, each of N integers, find summation of A[i]*B[j]*C[k] for all possible i, j, k
- Direct **programming** thinking: 3 nested loops to try all positions, and do the summation
- **Mathematically**, we can **Interchange** the Order of Summation, getting faster code
- $\sum_i \sum_j \sum_k A[i]*B[j]*C[k] = \sum_i A[i]*\sum_j B[j]*\sum_k C[k]$
- $= [\sum_i A[i]]*[\sum_j B[j]]*[\sum_k C[k]] = SumA*SumB*SumC$

Order of Summations: Code

```
int a[5] = {1, 2, 3, 4, 10};
int b[5] = {4, 4, 2, 1, 6};
int c[5] = {3, 3, 5, 6, 9};

int sum1 = 0;

// O(n^3)
for (int i = 0; i < 5; ++i) {
    for (int j = 0; j < 5; ++j) {
        for (int k = 0; k < 5; ++k) {
            sum1 += a[i] * b[j] * c[k];
        }
    }
}
int suma = 0, sumb = 0, sumc = 0, sum2 = 0;

// O(3n)
for (int i = 0; i < 5; ++i) suma += a[i];
for (int j = 0; j < 5; ++j) sumb += b[j];
for (int k = 0; k < 5; ++k) sumc += c[k];

sum2 = suma*sumb*sumc;

cout<<sum1<<" "<<sum2<<"\n";    // 8840 8840
```

Order of Summations

- Last example, all indices were **independent**
- $\sum_{i = [1, n]} \sum_{j = [i, n]} f(i, j) \Rightarrow \sum_{1 \le i \le j \le n} f(i, j)$
- Can we swap to sum j, then sum i? Use **inequality**
- Let's start with j, what are the limits? [1, n] as no i now
- $1 \le j \le n$. Now, let's iterate on i internally, any limits?
  - yes i bounded by [1, j]
- $\sum_{i = [1, n]} \sum_{j = [i, n]} f(i, j) \Rightarrow \sum_{j = [1, n]} \sum_{i = [1, j]}$
- $\sum_{i = [1, n]} \sum_{j = [i, n]} \sum_{k = [j, n]} = 1 \le i \le j \le k \le n$
- $\sum_{j = [1, n]} \sum_{i = [1, j]} \sum_{k = [j, n]}$ **or** $\sum_{j = [1, n]} \sum_{k = [j, n]} \sum_{i = [1, j]}$

Order of Summations

- Let say you have originally $1 \le j < k \le n$
- You decided to make replacement $k = k+j$
- $1 \le j < k + j \le n$
- How to make double sum over k, then j?
- for k: boundary is [1, n] as j not defined yet
- for j we have $j < k + j \le n$!
  - then 2 constraints
  - $0 < k$
  - $k \le n - j$
- $\sum_{k = [1, n]} \sum_{j = [1, n-k]} f(i, j)$
- Inequalities makes your life smooth :)

Harmonic Number

The sequence of ***harmonic numbers*** $\langle H_n \rangle$ is defined by the rule

$$\begin{aligned} H_n &= \sum_{k=1}^{n} \frac{1}{k} \\ &= \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n} \qquad \text{for } n \geq 0 \end{aligned}$$

Perturbation

- Technique helps to come up with facts
- Sometimes reaching these facts with simple math rules is painful

The initial step of a ***perturbation*** is to equate two expressions for S_{n+1}, the $n+1^{\text{st}}$ partial sum of the sequence $\langle x_n \rangle$.

$$S_n + x_{n+1} = x_0 + \sum_{k=1}^{n+1} x_k$$

Perturbation

$$S_n = \sum_{k=0}^{n} 2^k$$

$$\begin{aligned}
S_n + 2^{n+1} &= 2^0 + \sum_{k=1}^{n+1} 2^k = 1 + \sum_{k=1}^{n+1} 2^k \\
&= 1 + \sum_{k=0}^{n} 2^{k+1} \\
&= 1 + 2\sum_{k=0}^{n} 2^k \\
&= 1 + 2S_n \\
\Rightarrow \qquad S_n &= 2^{n+1} - 1 \qquad\qquad (\text{sol}
\end{aligned}$$

Perturbation

$$S_n = \sum_{k=0}^{n} H_k \quad \Rightarrow \quad H_{n+1} = \sum_{k=0}^{n} \frac{1}{k+1}$$

lost S_n.
try evaluating k * summand

$$S_n = \sum_{k=0}^{n} kH_k \quad \Rightarrow \quad \sum_{k=0}^{n} H_k = (n+1)H_{n+1} - (n+1)$$

$$S_n = \sum_{k=0}^{n} k^2 \quad \Rightarrow \quad \sum_{k=0}^{n} k = \frac{(n+1)^2 - (n+1)}{2} = \frac{n^2+n}{2}$$

$$S_n = \sum_{k=0}^{n} k^3 \quad \Rightarrow \quad \sum_{k=0}^{n} k^2 = \frac{2n^3+3n^2+n}{6}$$

Equation Arrangement

- Sometimes, the given equation is annoying... trying to work with it is hard
- So you may think in arranging it first
- $8x = 2y+4 \quad => \quad y = 4x - 2$
- $x^2 = 2^{2y-6} \quad => \quad y = \log_2 x+3$
- $F(n) = F(n+1) - F(n-1)$
  - Right terms one is up and one is down!
  - $F(n+1) = F(n) + F(n-1)$ …. now let $m = n+1$
  - $F(m) = F(m-1) + F(m-2)$ …. much better to solve
- Algebra [Cheat Sheet]

Computations

- Sometimes a complex summation can end up as one final formula
- Sometimes, you have to do full or partial computations
- E.g. $F(n) = \sum_{k = [1, n]} \lfloor n/k \rfloor$ for $N <= 1e9$
  - We can't find formula!
  - We also can't iterate 1e9 iterations
  - But we can observe floor value is same for many consecutive values.
  - E.g. for $n = 20$, $k = [7, 8, 9, 10]$ has $20/k = 2$

Thinking in Computations

- We can think in sum problems **symbolically** ($\sum 2n+1$) or **concretely** (e.g. list answer for n)
- Symbolically may be harder, but lets you figure out the sums, doing replacements or interchange order, evaluate summations!
- Sum of **k, k^2, k^3** are most popular needed
- Be careful from overflows. E.g. if **mod** is used it, apply i always. Use **long long** if needed

Readings

- Concrete Mathematics: Sums Chapter
- Evaluating Sums
- Summations

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً

problems

CF476-D2-C, Live Archive 3521

https://www.hackerrank.com/domains/mathematics/summations-and-algebra