Competitive Programming

From Problem 2 Solution in O(1)

Combinatorics

Permutations and Combinations 1

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

Permutations

- Permutations and Combinations play important roles in **counting**
- Permutations care with elements **order**, but combinations don't.
- What are the different arrangements (permutations) of {1, 2, 3}?
- (1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)
- How many ways? Use product rule
  - First choice is 3 possibilities. 2nd just 2. 3rd = 1
  - Generally: n * n-1 * n-2 …. *1 = n!

Permutations

- Same logic for: Given digits from 1-6, How many car plates of length 4 without repetition?
  - 6 * 5 * 4 * 3 … how to relate to 6! ? Add 2! up and down
  - 6 * 5 * 4 * 3 * 2 * 1 / 2 * 1 = 6! / 2!
  - Or generally: P(n,k) = **n! / (n-k)!** => to ways to compute
- If repetition allowed ? Direct product rule = n^k
- Your turn: How many even car plates of length 4? How many odd car plates of length 4 and digits sum divisible by 3? Can we find formula for last one, or must write code?

Permutations

- Group of 3 boy students and 4 girl students will set down on a table.
  - Table left side has 6 seats and right side has 5 seats
  - boys will set down on one side, and girls on another
- How many ways to set down the 7 students?
  - Boys can be left and girls on right …. Or reverse.
  - Compute each case and **sum** it
  - Seating boys is independent from seating girls
  - Then if X ways to set boys on left and Y for girls on right, we have total of X **multiple** Y. X and Y are permutations based (order matters). Use these notes to compute overall

Permutations

```
vector<int> permutation;
int n_perm = 4, perm_cnt = 0;
bool is_visited[4];

void get_perm(int i = 0) {
    if(i == n_perm){
        ++perm_cnt;      // finally will be 4! = 24
        return;      // you can print permutation here
    }

    for (int j = 0; j < n_perm; ++j) {
        if(is_visited[j])
            continue;

        permutation.push_back(j);
        is_visited[j] = 1;

        get_perm(i+1);

        is_visited[j] = 0;
        permutation.pop_back();
    }

}
```

Permutations

```
vector<int> p = {0, 1, 2, 3};

do {
    ++perm_cnt;
    // use p vector
} while(next_permutation(p.begin(), p.end()));

cerr<<perm_cnt; // 24

return 0;
get_perm();     cerr<<perm_cnt;
```

Permutations with repetition

- How many permutations of AAA? 6? 1?
- How many permutations of AAABB?
  - Imagine it as A1A2A3B1B2 (where A1=A2=A3, B1=B2)
  - Now think in permutation: **A1A2**B1**A3**B2
  - This is as same as **A3A1**B1**A2**B2
  - Specifically, fixing all except As, 3! of it are same!
  - Now, assume on of these 3! is fixed but B's are changed
  - 2! of them are same e.g. A3A1B1A2B2 = A3A1B2A2B1
  - Then 3! * 2! items are duplicate => 5! / (3! * 2!)
- P(n) = n!..but P(n, [c1, c2...cm]) = n!/(c1!c2!...cm!) where c1 is repeated char

Permutations with repetition

- Write code for:
- long long perm(int n, vector<int> s) which computes # of permutations with elements repeated s1, s2...sn
- Avoid over flow as possible
- Hints: say perm(8, {2, 3})
  - use a vector for n numbers of n! (numerator): 1, 2...8
  - use a vector for all s items (denominator): 1 2 1 2 3
  - Use GCD to filter the denominator values
- Update: Let function return answer % 10^9+7

Combinations

- How many ways to select some items, ignoring the order of selection + no repetition?
  - permutation : {1, 2, 5} != {2, 1, 5}
  - combinations: {1, 2, 5} == {2, 1, 5}
- If we have 4 students? how many ways to:
  - select 1 student: 4 ways {1}, {2}, {3}, {4}
  - select 2 students: 6 ways {1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4}, {3, 4}. This is called C(4, 2)
- C(n,k) = P(n,k) but with remove equal subsets
- C(n,k) = P(n,k) / k! = n! / k! (n-k)!

Combinations

- One way to compute, either 3 factorials..or do cancelations for the terms

$$\binom{n}{k} = \frac{n(n-1)\dots(n-k+1)}{k(k-1)\dots 1},$$

- In case possible overflow, put numerators in array, denominator in array and cancel terms as possible first (use GCD).

$$\binom{52}{5} = \frac{52 \times 51 \times 50 \times 49 \times 48}{5 \times 4 \times 3 \times 2 \times 1} = \frac{311{,}875{,}200}{120} = 2{,}598{,}960.$$

Combinations

$$\begin{aligned}
\binom{52}{5} &= \frac{52!}{5!47!} \\
&= \frac{52 \times 51 \times 50 \times 49 \times 48 \times \cancel{47!}}{5 \times 4 \times 3 \times 2 \times \cancel{1} \times \cancel{47!}} \\
&= \frac{52 \times 51 \times 50 \times 49 \times 48}{5 \times 4 \times 3 \times 2} \\
&= \frac{(26 \times \cancel{2}) \times (17 \times \cancel{3}) \times (10 \times \cancel{5}) \times 49 \times (12 \times \cancel{4})}{\cancel{5} \times \cancel{4} \times \cancel{3} \times \cancel{2}} \\
&= 26 \times 17 \times 10 \times 49 \times 12 \\
&= 2{,}598{,}960.
\end{aligned}$$

$$\begin{aligned}
\binom{52}{5} &= \frac{n!}{k!(n-k)!} = \frac{52!}{5!(52-5)!} = \frac{52!}{5!47!} \\
&= \frac{80{,}658{,}175{,}170{,}943{,}878{,}571{,}660{,}636{,}856{,}403{,}766{,}975{,}289{,}505{,}440{,}883{,}277{,}824{,}000{,}000{,}000{,}000}{120 \times 258{,}623{,}241{,}511{,}168{,}180{,}642{,}964{,}355{,}153{,}611{,}979{,}969{,}197{,}632{,}389{,}120{,}000{,}000{,}000} \\
&= 2{,}598{,}960.
\end{aligned}$$

Combinations

- C(1000, 2) = 499500
- C(1000, 999) = C(1000, 1) = 1000
- C(66, 33) = 7219428434016265740
- C(68, 34) = Overflow in long long
  - we may end up with small or big values :)
- C(5, k): 1 5 10 10 5 1 for k = [0 - 5]
- C(6, k): 1 6 15 20 15 6 1 for k = [0 - 6]
- Notice the symmetry!
- C(n, k) = C(n-k)

Combinations

```
long long cnt = 0;

for (int i1 = 1; i1 <= 20; ++i1) {
    for (int i2 = i1+1; i2 <= 20; ++i2) {
        for (int i3 = i2+1; i3 <=20; ++i3) {
            for (int i4 = i3+1; i4 <= 20; ++i4) {
                cnt++;
            }
        }
    }
}

cout<<cnt<<"\n";    // 4845 = C(20, 4)
// {i1, i2, i3, i4} 4 values selected without repetition
// from set {1, 2, 3.....20}
```

Combinations

```
vector<int> combination;
int n = 20, m = 4, cnt = 0;

void get_combination(int i = 0, int last_val = 0) {
    if(i == m) {
        ++cnt;      // finally will be 4845
        return;     // you can print combination here
    }

    for (int j = last_val+1; j <= n; ++j) {
        combination.push_back(j);
        // Think: dynamically create one more loop
        get_combination(i+1, j);    // backtracking
        combination.pop_back();
    }
}

int main() {
    get_combination();
    cerr<<cnt<<"\n";
```

Combinations

- Building Committee of 7 persons out of 8 women and 9 men?
- No more conditions? C(17, 7)
- Has Exactly 5 women? C(8, 5) * C(9, 2)
- Has at least 5 women? w >= 5
  - **Convert** inequality to loop: W(5) + W(6) + W(7)
  - C(8, 5) * C(9, 2)+C(8, 6) * C(9, 1)+C(8, 7) * C(9, 0)
- Has at least 0 women? Useless => C(17, 7)
- Has at least 1 women? C(17, 7) - C(9, 7)

Combinations

- Combination concept can be related to:
  - Exact / At most / At least
- Atmost(k) = SUM Exact(i) where i = [0-k]
- Exact(k) = Atmost(k) - Atmost(k-1)
- In range (start, end) =
  - SUM Exact(i) where i = [start-end]
  - Atmost(end) - Atmost(start-1)
  - Atleast(start) - Atleast(end+1)

Combinations: Think

- Using digits 1, 2, 3, 4 ...
  - How many 4-digit numbers?
  - How many 4-digit numbers with at least 1 digit repeated?
- How many three digit numbers can be formed with the digits: 0, 1, 2, 3, 4, 5?
- 10 points on a plane of which 4 are collinear, no other 3 are collinear, how many lines?
- How many ways are there to travel from the upper-left **corner** of an n × m grid to the lower-right corner by walking only down and to the right?

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً

CF294C-D2