Programming League National 2022
Final Round Editorial

Contest Team

April 2022

Contents

1 Introduction

The problems are sorted by categories, Closed Category then Open Category. Note the questions are ordered lexicographically and are NOT in order of difficulty.

Programming League Contest 2022 is hosted using Codeforces.

Here are the contest links:

- Final Round [Closed]
- Final Round [Open]

We would like to appreciate Mike Mirzayanov for Codeforces and Polygon platform. Huge thanks to our problem authors and testers.

- Tay Qi Xiang, UM' 24
- Chin Shan Hong, UM' 24
- Chong Yi Fong, UM' 24
- Lim Hong Zhi, UM' 24
- Chan Chee Sun, UM' 24
- Lei Wing Yee, UM' 24
- Marvin Chin, UM' 25
- Lim Yun Kai, ACM ICPC National Champion & IOI Bronze

2 Closed Category

2.1 A Meowy Night

Author: Chan Chee Sun

Tags: Implementation

Abridged Statement

Given a matrix of boxes with values and states, sum up values in all boxes that are in the correct state. These boxes can be on the state of 0 and 1, where if it's 0 the number with the same index can then be summed up. The states are initially in a checkered pattern that starts with 0 at row 0 and column 0 (top left). The states change rectangularly between 2 coordinates.

Solution

Create two 2-dimensional matrices, one for storing integer values and one to check the state of each box. Based on each type of query

1) " + "
- Add values into the first matrix based on the given indexes (rows and column).

2) " B "
- Change all values in the state matrix between 2 coordinates to 1 (not summed)

3) " T "
- Change all values in the state matrix between 2 coordinates to 0 (can be summed)

4) " ? "
- Compare between 2 matrices, if the state matrix shows 0 then add the value in the corresponding index in the first matrix

Time complexity : $O(r \text{ x } c)$

2.2 Aspect Ratio

Author: Lim Hong Zhi

Tags:maths**,** implementation

Abridged Statement

Determine the aspect ratio with the given resolution. If the resolution is not available, then just output a "No"

Solution

In order to get the aspect ratio, we are required to get the highest common divisor (or called highest common factor). Thus we will need to find the HCF for the width and the height of the screen resolution. We should use the Euler algorithm where it uses the modulo operator in a recursion method in order to finish the test cases in the time limit.

$$\begin{aligned} &HCF(A,B) = HCF(B, A\%B), \\ &\quad where: \\ &\qquad 1.HCF(A,0), return A \\ &\qquad 2.HCF(0,B) return B \end{aligned} \tag{1}$$

In this question, you are not able to brute force by using the division method such as 1920/1080 and compare it to the division of the aspect ratio 16/9 as there are test cases like 1920000000000001x1080000000000000000 to avoid you from doing that. Besides, you are required to get the greatest number and put it in the front for the aspect ratio if you have meet a screen resolution such as 1080/1920 so that you are able to get $16:9$.

Time complexity : $O(log \max(a,b))$, where a and b is the value in the HCF.

2.3 Choosing Module

Author: Tay Qi Xiang

Tags: Tree, Dynamic Programming

Abridged Statement

Given n courses, each with their own credit hour and prerequisite course. Find the maximum credit possible if Meow is taking M courses.

Solution

It is mentioned in the title that there is at most one direct prerequisite course for each course, and connecting the prerequisite courses of each course with it will form one or more trees. To do this we can create a node 0 and connect the root nodes of all trees to node 0. The credits of node 0 are 0, and the number of allowed elective courses should be increased by one.

After splicing the forest into a tree, we need to perform tree dynamic programming.

Consider every node in the tree: this node may be the parent of one or more nodes (prerequisites), or it may not. Let f[i][j] denote the maximum score that can be obtained by choosing j courses for node i in total (j courses include himself). The premise of selecting all its child nodes is to select this node, so f[i][1] must be equal to the weight of the i node (credits)

Traverse all the child nodes of the i node in a certain order. When traversing to the x node, we assume that all the values of f[x][k] have been obtained. Since there is no weight of the x node and its sub-tree in f[i][k], the f array of the i node can be updated through the x node. That is, f[i][k] = max(f[i][k], f[i][k - p] + f[x][p]) ($1 \leq k \leq m, 0 \leq p < k$)

The reason for p ¡ k is that when p = k, the value of k - p in f[i][k - p] is equal to 0, which means that the node i cannot be selected. Since i is a prerequisite for node x, The x node cannot be selected without selecting the i node, so p ¡ k

Note when enumerating k: Since f[i][k] is updated by f[i][k - p], then it is necessary to ensure that f[i][k] is updated when f[i][k] is updated - p] is not updated, since p is a positive integer, we can enumerate k in reverse order

Time complexity : $O(NM)$

2.4 Gas Stations

Author: Lei Wing Yee

Tags: Greedy

Abridged Statement

The task is to find the minimum starting index of a circular route. There are N gas stations and the amount of gas at station i is $arr[i]$. With an unlimited gas tank which starts at zero in the beginning of the journey, it costs $cost[i]$ of gas to travel from station i to its next station $(i + 1)$.

Solution

Create an array to store the valid starting index from where the car could touch all the stations. For each station i, fill the fuel tank with $gas[i]$ and burn the fuel by $cost[i]$. If at any point the tank is < 0 then, choose the next index as the starting point. At last, check if the total fuel available at the gas stations is greater than the total fuel burnt during the travel. Return the minimum starting index.

Time complexity : $O(n)$

2.5 Identical Strings

Author: Chan Chee Sun

Tags: String, Implementation

Abridged Statement

Given 2 strings, string 1 and string 2, when we substitute letters in string 1 for another letter, it will become string 2. For example, when we swap p for l, u for a, r for n, y for o, f for t, we will get "lannotanno" from "purryfurry". These substitutions may not intersect, where one letter can only be substituted into one other letter.

Solution

We first check if both strings are similar in length, return not identical if it is not similar.

Then we can initialize 2 arrays of size 26 where the index of each array indicates a letter a – z. One array to save which character corresponds to which letter in the second string (based on ASCII values) and another one to check if a character has already been stored in the array.

If the second array indicates that a letter has been converted, we check if the letter corresponds to the new letter in the second string.

Time complexity : $O(N)$

2.6 Mental Calculation

Author: Tay Qi Xiang

Tags: Implementation

Abridged Statement

Given n numbers, how many number in the list are the sum of the other two number in the list.

Solution

Note brute forcing every possible number is not a viable option. (Time complexity : $O(n^3)$)
We store the number in array $a[i]$can create a array $g[a[i]]$ to store if the number is present or not.

Then we can have an array $t[a[i]+a[j]]$, and a double for loop (i and j) to store the sum of numbers.

Then we compare $if(t[i] > 0 \&\& g[i])$, then ans++. We will get our answer. This method eliminates the possibility of t[i] got added multiple times by different number.

See complete code for better understanding.

Time complexity : $O(n^2)$

2.7 Power Grid

Author: Marvin Chin

Tags: Greedy

Abridged Statement

Given a connected and undirected network, remove some edges such that all points of the networks remains connected and has the minimum sum of weight possible. Then, find the difference between the sum of the weights in the initial network and the optimized network.

Solution

This is a minimum weight spanning tree question, as such the problem can be solve quickly using Kruskal's minimum spanning tree algorithm.

First, we sort all the edges in non-decreasing order of their weight. Then, we pick the smallest edge to check if it forms a cycle within the spanning tree formed so far. If a cycle is not formed, include this edge, else discard it.

To detect cycles, we can use Union-find algorithm, which performs two operations. A find() - to determine which subset an element is in; and Union() - to join two subsets into a single subset if the two subset does not belong to the same set, else we do not perform union.

This step is repeated for (V-1) edges in the network/spanning tree.

Time complexity : $O(E \, log \, V)$

2.8 Prime

Author: Lim Hong Zhi

Tags: maths**,** implementation**,** sieve

Abridged Statement

Find the sum of all the prime factors of a number and modulo the maximum of the prime numbers. Take the result and check whether it is a prime number still.

Solution

For prime factorization, you should check from the lowest prime number, which is 2 as it will eliminate all the greater factors that have the factor 2 as its factor. By slowly working up, you are able to collect all the prime factors available in the number.

You can have a list that will store the available prime factors and add it all up afterwards, or adding it up after checking the prime number is a factor to the number and constantly updating the maximum.

Then just check whether the result is a prime number, you should decrease the number of loops that is being run by setting the loop limit to the square root of the number that you wanted to check as there will be no more possibility that there is a factor that is greater than $\sqrt{x}$, if x is the result.

Time complexity : $O(logn)$

2.9 Math Formula

Author: Tay Qi Xiang

Tags: Interval Dynamic Programming

Abridged Statement

Given n numbers. Add k "×" signs and n-k-1 "+" signs such that the sum of n numbers is the largest. Note that you may add any number of parenthesis.

Solution

First, let $sum[i]$ be the prefix sum for the numbers.

Let $f[i][j]$ represents the optimal solution of inserting j multiplication signs into the first i numbers.

$$f[i][j] = max(f[i][j], f[p][j-1] \times (sum[i] - sum[p]))$$

where is the the position of the last "×" sign.

Time complexity : $O(n^3)$

2.10 Wall Painting

Author: Chin Shan Kang

Tags: Depth First Search, Matrix, Recursion

Abridged Statement

Given a $m \times n$ integer grid representing a wall, the index location of starting brick (r and c), and the $newColour$ to be painted on the starting brick. Paint the starting brick with $newColour$ and also all the bricks with the same initial colour as the starting brick which are also connected either vertically or horizontally to the starting brick, plus any bricks connected horizontally or vertically to those bricks (also with the same colour), and so on.

Solution

Let's say *colour* is the initial colour of the starting brick. Let's paint the starting brick. We change the *colour* of the brick to $newColour$, then check the 4 neighboring bricks to make sure they are valid bricks of the same *colour*, and of the valid ones, we paint those bricks, and so on. We can use Depth-First Search to perform the painting process on a target brick.

Time complexity : $O(n)$, where n is the number of bricks in the wall, we might process every bricks.

3 Open Category

3.1 Birthday Cake

Author: Tay Qi Xiang

Tags: Depth First Search (DFS), Recursion

Abridged Statement

Given a cake with sides X and Y. How to cut the cake into N pieces such that every piece has equal area. Output the maximum value for the ratio of the long side to the short side of all those N pieces to be the smallest.

Solution

We notice that N is small ($N \leq 10$), we can try and use searching. But how can we optimise our brute-force strategy?

Since we can only cut along the x-axis or y-axis, and every cut must cut the cake into two pieces, and furthermore area for the final state of each cake must be $\frac{1}{n}$.

Thus, we can deduce the size for two pieces must be :

$$(\frac{k}{n}, \frac{n-k}{n}),\ k \in \mathbb{N}$$

First of all, we assume that the current rectangle has a length of X and a width of Y. To separate into K blocks, it is not difficult to think that the shortest length mx of the separated block is $\frac{x}{k}$, and the shortest width my is the shortest $\frac{y}{k}$, and The length of each cut must be a multiple of mx or a multiple of my.

So we recursively search, each time the length or width is cut, take the maximum value of the ratio between the two divided blocks, update the minimum value of the maximum value, and we will get our answer by returning x/y when n==1.

We can let $dfs(x, y, n)$, where x is length, y is width, and n is the smallest ratio when cutting into n pieces. And then we simply search every possibility (both horizontal cut and vertical cut).

See complete code for better understanding.

Time complexity : $O(2^{2N})$

3.2 Catnip Trading

Author: Tay Qi Xiang

Tags: Greedy

Abridged Statement

Given n cats, each either wanting to buy or sell some number of catnip. Transporting 1 catnip from a_i to a_{i+1} takes 1 unit of energy and $\sum_{i=0}^{n} a_i = 0$. What is the minimum amount of energy needed such that every cat has their demand fulfilled.

Solution

Since every element is needed to become zero so just keep on accumulating the total work . Suppose the first element is non-zero then it must be transferred to the next element so whether it is positive or negative you have to transfer it all. And the work will be absolute value of the transferred element.

In this way you just need to sum it up till the last term and you will get the total work.

Time complexity : $O(n)$

3.3 Checkers

Author: Tay Qi Xiang
Acknowledgments: This question is heavily inspired by the question in Tsinghua CP training camp.

Tags: Binary Tree, Lowest Common Ancestor(LCA)

Abridged Statement

Given position A, B, C. Can those pieces be moved into X, Y, Z and if so what is the minimum number of moves. You are allowed to move a piece by 'jumping' over another piece, with the another piece as your central axis, i.e. the distance between the two pieces must stay the same. And you are only allowed the jump over 1 piece at a time.

Solution

For ease of explanation, let us denote the left piece as A, middle piece as B and the right piece as C. Note that **only one piece is allowed to be skipped at a time**. This is the key in solving the problem. If we look closely at the rules of this game, there is only 4 types of 'jumping'

1. B jump towards A : $(A, B, C) \Longrightarrow (2A - B, A, C)$
2. B jump towards C : $(A, B, C) \Longrightarrow (A, C, 2C - B)$
3. Let $d1 = B - A;\ d2 = C - B$:
   (a) If $d1 < d2$: $(A, B, C) \Longrightarrow (B, 2B - A, C)$
   (b) If $d1 > d2$: $(A, B, C) \Longrightarrow (A, 2B - C, B)$

Since only one piece is allowed to be skipped at a time, 3(a) and 3(b) will have conflict if we perform both operation. Thus, there is only 3 'jumping' method.

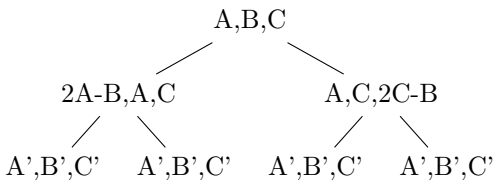
Let's look at these three formulas again, and we can find that for the first and second cases, we jump from the middle point to both sides, which is equivalent to expanding the boundary. The third type of jumping inward from both sides is just the opposite, narrowing the boundary.

Thus, this is a binary tree.

Obviously, the boundary cannot be infinitely smaller. If we shrink and shrink, we will encounter a situation where you can't shrink anymore. Let's think about the situation where we can't shrink anymore.

If we observe 3(a) and 3(b), surprisingly, we missed a situation in this discussion, which is $d1 = d2$.

If $d1 = d2$, then A will inevitably jump to the same point as C, which is violating the question requirement. So when $d1 = d2$, this state can only jump from the middle to both sides, that is, there are only two transition states. And this will be our root node, and the shape of the binary tree is as shown in the following figure:



It can be found that for each group of A, B, C there is a unique corresponding root, and any group of A, B, C (between binary trees) can be transformed into each other, and the number of transformation operations is equal to the distance between the two states on the tree.

Thus, for initial state A, B, C and final state X, Y, Z, we will need to find their LCA.

$$ans = \text{sum of initial state to LCA} + \text{sum of final state to LCA}$$

Note: One of the ways to find LCA is by using binary lifting.

If initial state A, B, C and final state X, Y, Z do not have the same root, then the answer is **NO**. Because we cannot jump out of the boundary of this tree, root different that means tree different.

However, a complication arises. What should we do since there are too many states to save?

Lets manually calculate the answer first. Initial=(1,10,11), Final= (1,2,3).
$d1 > d2 \Longrightarrow (1, 9, 10)$
$d1 > d2 \Longrightarrow (1, 8, 9)$
$d1 > d2 \Longrightarrow (1, 7, 8)$

We notice that we are doing the same calculation again and again. We notice that we have to repeat $\frac{d1-1}{d2}$ times the operation – (every time we jump $d2$ distance, and we cannot exceed A, thus the total distance is $d1 - 1$).

Hence, let $d3 = \frac{d1-1}{d2}$. We will get : $(A, B - d3 \times d1, C - d3 \times d1)$, and vice-versa for the other part.

Time complexity : $O(log^2 D)$

3.4 Chocolate Buffet

Author: Tay Qi Xiang

Tags: Math, Greedy, Binary Numbers

Abridged Statement

Meow has chocolate with volume $1, 2, 4, 8, ...$ Meow must eat at least X grams but not more than Y grams. What is the maximum number of chocolate Meow can eat?

Solution

We can re-frame the question as:
In a binary system, what is the maximum number of "1" bit from X to Y.

We can convert X into a binary number. Then, we simply start searching from the least-significant bit. Every time we find a "0" then we will greedily convert it into "1" and compare it with Y and check whether it is smaller than Y or not.

Time complexity : $O(log(Y))$

3.5 Count Equal Pair

Author: Chin Shan Hong

Tags: Two Pointers

Abridged Statement

Given two arrays a and b, sorted in non-decreasing order. Find the number of pairs (i, j) for which $a_i = b_j$.

Solution

We can solve this problem with two pointers method.

See solution file for detailed comments with code.

Time complexity : $O(n)$

3.6 Dice

Author: Tay Qi Xiang

Tags: Math, Dynamic Programming, Probability

Abridged Statement

n people stand in a row and Meow is at m^{th} place.

Every round a dice will be thrown.

1. If dice = 1, person in front of queue wins
2. If dice = 2,3,4, person in front goes to the back
3. If dice = 5,6, the person in front gets kicked out from the queue

If there is only 1 person left, the person wins. What is Meow's probability to win?

Solution

Let us denote $f(i, j)$ as the probability that the j^{th} person wins when there are i^{th} people in the queue. The boundary(base) case is $f(1, 1) = 1$ —— When there is only one person, he will definitely win. And there must exists $j \leq i$.

Next we have to consider the state transition equation.

1. If an 1 is thrown, there is $\frac{1}{6}$ probability of losing the game.
2. If $2, 3, 4$ is thrown, there $\frac{1}{2}$ probability of going to the back of the queue, that is, $f(i, j - 1)$.
3. If $5, 6$ is thrown, there is $\frac{1}{3}$ probability that head of the team will get out, all others will take a step forward, and transfer to $f(i - 1, j - 1)$.

Since we only consider the win rate, the first case (losing) is not counted in our DP equation. At the same time, we noticed that this DP sequence may form a loop, so we can't get started directly.

What should we do?

According to the question, there are n people, and Meow is ranked m^{th}. Currently, we require $f(n, m)$, and then we know that Meow has $\frac{1}{2}$ probability to go to $f(n, m-1)$, and there is $\frac{1}{3}$ probability to go to $f(n-1, m-1)$. Hence, we can deduced:

$$f(n,m) = \frac{1}{2}f(n,m-1) + \frac{1}{3}f(n-1,m-1) \tag{1}$$

This cycle repeats, and finally we will find the case of $f(n, 1)$, in which case we will have $\frac{1}{6}$ probability of winning, $\frac{1}{2}$ chance to move to the end of the queue $f(n, n)$, and another $\frac{1}{3}$ chance to lose, so

$$f(n,1) = \frac{1}{6} + \frac{1}{2}f(n,n) \tag{2}$$

We can deduce for all cases of $n-1, n-2, \cdots, 1$. This will form a continuous multivariate linear equation system.

Lets take an example. There are 4 person and Meow is on the 3^{rd}.

$$\begin{aligned}
f(4,4) &= \frac{1}{2}f(4,3) + \frac{1}{3}f(3,3) \\
f(4,3) &= \frac{1}{2}f(4,2) + \frac{1}{3}f(3,2) \\
f(4,2) &= \frac{1}{2}f(4,1) + \frac{1}{3}f(3,1) \\
f(4,1) &= \frac{1}{2}f(4,4) + \frac{1}{6}
\end{aligned}$$

$$\begin{aligned}
f(3,3) &= \frac{1}{2}f(3,2) + \frac{1}{3}f(2,2) \\
f(3,2) &= \frac{1}{2}f(3,1) + \frac{1}{3}f(2,1) \\
f(3,1) &= \frac{1}{2}f(3,3) + \frac{1}{6}
\end{aligned}$$

$$\begin{aligned}
f(2,2) &= \frac{1}{2}f(2,1) + \frac{1}{3}f(1,1) \\
f(2,1) &= \frac{1}{2}f(2,2) + \frac{1}{6}
\end{aligned}$$

$$f(1,1) = 1$$

Through observation, we can see that the system of equations has two properties:

1. Every equation's left hand side contains a single unknown variable with coefficient of 1
2. Every equation's right hand side contains an unknown with coefficient of $\frac{1}{3}$, and an unknown with coefficient $\frac{1}{2}$

Hence, we can simply use the substitution elimination method to solve this question.

Time complexity : $O(n^2)$

Extra: (this formula is not required to solve this question)
The recurrence formula for this question is:

$$f(i,j) = \sum_{k=0}^{j-1} \left(\frac{1}{3}\right)^k \times \left(\frac{1}{2}\right)^{j-k-1} \times \binom{j-1}{k} \times f(i-k,1)$$

(proof is left to the reader, lazy to type)

3.7 Railway Designer

Author: Chin Shan Hong

Tags: Minimum Spanning Tree

Abridged Statement

Given n railway stations which are fully connected with each other through railways and a train can travel in both direction from one station to another. Calculate the minimum total maintenance cost for the new railways connection which is equal to the minimum total geographical distance between each station in kilometer (km) where cycles are not allowed in the new railway connection

Solution

There are 3 important clues in the problem statement which are no cycles are allowed in the new railway connections, trains can travel in both direction (undirected), and we are required to find the minimum total maintenance cost which is equal to the minimum total geographical distance where all the stations are still connected.

This is essentially a minimum spanning tree problem. Minimum spanning tree is a subset of edges (railway connections), edge-weighted undirected graph that connect all the vertices (stations) together, without any cycles and with the minimum possible total edge weight.

To solve this problem, we can use Prim's Minimum Spanning Tree algorithm.

1. Calculate the geographical distance between each stations using the Haversine formula based on their latitude and longitude and store it into an adjacency matrix.
2. Create a set called $mstSet$ that keeps track of stations included in the MST.
3. Assign a key value to all stations in the input graph.
4. Initialize all key values as INFINITE.
5. Assign key value as 0 for the first station so that is it picked first.
6. While $mstSet$ doesn't include all stations.
   (a) Pick a station u which is not there in $mstSet$ and has minimum key value.
   (b) Include u to $mstSet$.

(c) Update key value of all adjacent stations of u. To update the key values, iterate through all adjacent stations. For every adjacent station v, if weight of edge $u-v$ is less than the previous key value of v, update the key value as weight $u-v$. The idea of using key values is to pick the minimum weight edge from cut. The key values are used only for stations which are not yet included in MST, the key value for these stations indicate the minimum weight edges connecting them to the set of stations included in MST.

7. Print the minimum total maintenance cost in integer after applying floor function.

Time complexity : $O(n^2)$

3.8 Red Notice

Author: Chin Shan Hong

Tags: Binary Search

Abridged Statement

Given n rooms, the nth room has k piles of treasures and h hours until the guard return. Find the minimum piles of treasure to steal per hour (m) such that Nolan Booth can steal all the treasures within h hours. Each hour, Nolan Booth choose some piles of treasures and steal m treasures from that pile. If the pile has less the m treasures, then he steals all of them, and won't steal any treasures during that hour.

Solution

The given problem can be solved efficiently by using binary search. Create a Boolean function to check if the chosen speed (treasures / hour), m is enough to steal all treasures within given h hours time or not. Lower limit of m is 1 treasure per hour as Nolan Booth steal 1 treasure per hour, and upper limit is the maximum treasures of all piles. Apply binary search on the possible answer range to get minimum value of m. If the Boolean function satisfies the mid value, reduce upper limit to mid, else update lower limit to $mid + 1$.

Time complexity : $O(n\ log\ w)$ (w is the max treasures from all piles)

3.9 Security System

Author: Tay Qi Xiang

Tags: Math, Combinations

Abridged Statement

There are two types of balls, black balls (signal 0) and red balls (signal 1), and there is no difference between balls of the same type. There are now n distinct boxes (storage areas) into which a black balls and b red balls are to be placed. Find the total number of ways.

Each box can hold as many balls as you want, or none. And the above $a + b$ balls do not need to be all put into the box, and even no balls can be put into the box.

Solution

First, since it is not necessary to put all of them in the box, let's discuss the situation when the number of balls of each type is determined. Let us denote the formula for n boxes, the number of solutions to put i black balls and j red balls (all put) as

$$f(n, i, j)\ (0 \leq i \leq a, 0 \leq j \leq b)$$

Thus, it is not difficult to see that the total number of possible ways is:

$$ans = \sum_{i=0}^{a} \sum_{j=0}^{b} f(n, i, j)$$

Now, the question arises, how do we find $f(n, i, j)$. Note that $f(n, i, j)$ means that we place i black ball, and then place j red ball. This two steps are **independent events**.

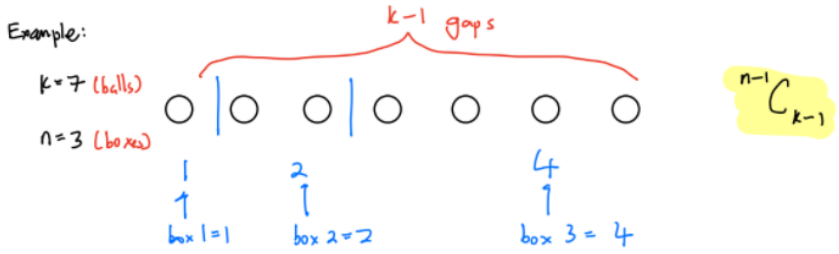
$$f(n, i, j) = g(n, i) \times g(n, j)$$

where $g(n, k)$ means the number of ways to put k similar balls into n boxes.

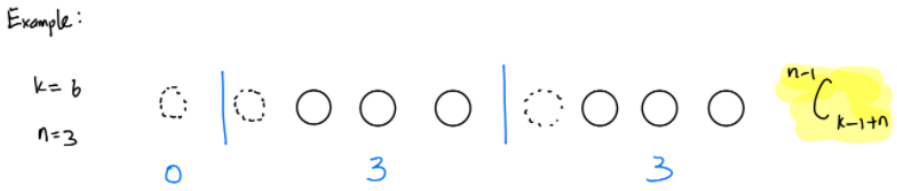
Now, back to high school mathematics – find the number of ways to put k balls into n boxes.

We now simplify the question. If we put k similar balls into n boxes, and every box must get at least one ball, how to we solve this?

k balls have $k-1$ gap, we can insert $n-1$ boards into the gaps, then the board will split the balls into k partitions. Hence, we insert $n-1$ boards into $k-1$ gaps, the formula is ${}^{n-1}C_{k-1}$.



But the problem now is that some boxes might not get any ball. Thus, we improve the condition above: add n balls such that every box has at least one ball. After putting the balls, we can then take one ball back from every box, then we will get back to our original situation. The formula is : ${}^{n-1}C_{k+n-1}$.



Hence, our formula is

$$ans = \sum_{i=0}^{a}\sum_{j=0}^{b} f(n,i,j)$$

$$ans = \sum_{i=0}^{a}\sum_{j=0}^{b} g(n,i) \times g(n,j)$$

$$ans = \sum_{i=0}^{a}\sum_{j=0}^{b} \binom{n-1}{i+n-1} \times \binom{n-1}{j+n-1}$$

Time complexity : $O(n + max(a,b) + ab)$

Note that we may use Pascal's Triangle to speed up the calculations for ${}^nC_r$.

The following is the simplified code.

```
for(int i = 0; i <= a; i++)
    for(int j = 0; j <= b; j++)
        ans += comb[n+i-1][n-1] * comb[n+j-1][n-1];
```

3.10 UM Librarian

Author: Chin Shan Hong

Tags: Brute Force, Arrays, Merge Sort

Abridged Statement

Given two arrays of integers, count the number of adjacent swaps (number of inversions) needed to arrange the elements present in the second arrays according to the index position of the elements in the first array. The elements which is at the lower index position in the first array will have a higher priority to be arranged in front of the second array. Noted that the elements in the first arrays are unique while the elements in the second arrays might be duplicated or missing as according to the elements that appear in the first array.

Solution

We can solve this problem using a HashMap or Dictionary and also enhanced merge sort. We need to store the elements as keys in the HashMap or Dictionary and their index as values.

The idea is similar to merge sort, divide the second array into two equal or almost equal half in each step until base case is reached.

Create a function $merge$ that counts the number of inversions when two halves of the second array are merged, create two indices i and j, i is the index of the first half, and j is the index of the second half. If $a[i]$ is greater than $a[j]$ in terms of the index position in the first array, then there are $(mid - i)$ inversions. Because left and right subarrays are sorted according to their index position, so all the index positions of the remaining elements in left subarray $(a[i+1], a[i+2], ...a[mid])$ will be greater than $a[j]$.

Create a recursive function to divide the second array into halves and find the answer by summing the number of inversions in the first half, the number of inversions in the second half and the number of inversions by merging the two. The base case of recursion is when there is only one element in the given half. Lastly print the answer.

Time complexity : $O(n\ log\ n)$