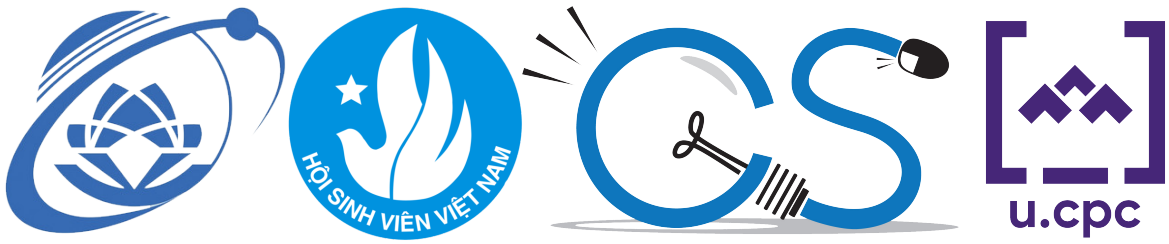
UIT Collegiate Programming Contest

Problems and Solutions

Truong An Pham Nguyen

Vietnam National University, Ho Chi Minh City
University of Information Technology
Falcuty of Computer Science

May 29, 2022



Contents

Problem A: Simple Chess

Problem A: Simple Chess

- Exhaustive search with Breadth-first search
- How to store visited node (chessboard)
- Each chessboard is 4x4 matrix $\rightarrow$ 16 cells in total $\rightarrow 2^{16} = 65536$ different boards to store
- Let each cells represented by 1 bit, convert each board into a single integer, no more than 200KiB of memory is needed.
- Time complexity: $\mathcal{O}(2^n \times n)$, with $n = 16$.

Problem B: The sound of a cat

Problem B: The sound of a cat

- Let T be the cleaned S if two consecutive characters are identical: delete 1 of them.
- If S = 'meow', there is only 1 'm' and 1 'o', then the answer is meow.
- If S = 'pur', there is only 1 'p' and there is only 1 'u' and the number of 'r' is greater than or equal to 2, then the answer is purr.
- If S = 'roar', 'r' appears twice and 'a' appears once, then the answer is roar.
- Otherwise, the answer is human noises.
- Time complexity: $\mathcal{O}(|s|)$.

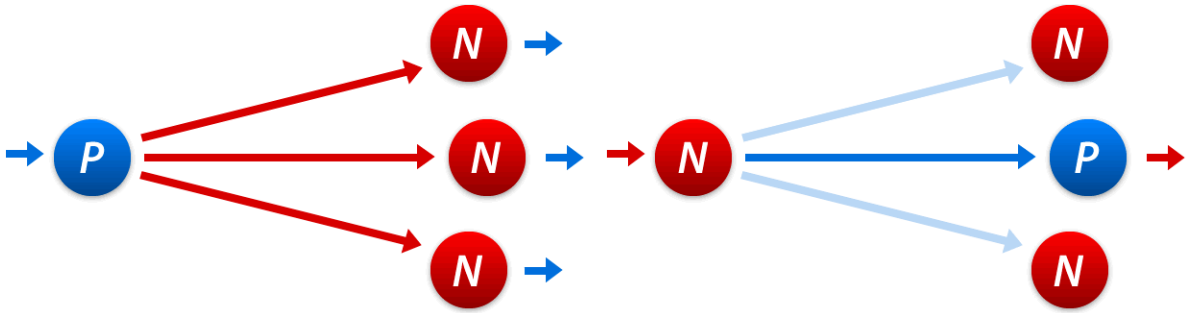
Problem C: A great invention

Problem C: A great invention

- Normal approach: for every integer i from 1 to n, find its divisors and calculate sum of them. $\rightarrow$ **TIME LIMIT EXCEEDED**.
- Reversed approach: for every integer x, find the amount of integers m from 1 to n so that x is a divisor of m. It is clear that there are $\lfloor \frac{n}{x} \rfloor$ numbers that satisfied.
- The answer is $\sum_{i=1}^{n} i \lfloor \frac{n}{i} \rfloor$.
- However, if i is a divisor of n, $\frac{n}{i}$ will also be a divisor of $n \Rightarrow$ we only need to traverse i from 1 to $\sqrt{n}$.
- Time complexity: $\mathcal{O}(\sqrt{n})$.

Problem D: Wizarding World

Wizarding World



- This game is over, because except for a finite number (2) of reflections about the line $y = x$,the coordinates are changed to non-negative integers (and at least one coordinate is changed to a positive number).
- Algorithm for solving this problem is DFS/BFS or dynamic programming (states) where the state is a pair of coordinates, and two logical variables, which denote if the 1 (2) player had reflected a point about the line $y = x$.
- Total number of states is obtained by $S \leq 4D^2$ (pairs of coordinates) $\times 4$ (Boolean variables).
- Processing of one state takes $O(N)$ actions (do not forget to try to reflect the point about the line $y = x$, if the player had not made it earlier in the game). Thereby, we have the overall asymptotic $\mathcal{O}(ND^2)$.

Problem E: A very hard problem

Problem E: A very hard problem

- First of all, you need to acquire **Dynamic Programming with Sum of Subsets (DP SOS)** to solve this problem.
- We will construct an array dp_1 which $dp_1[x]$ means the number of a_i so that $a_i|x = x$ and an array dp_2 which $dp_2[x]$ means the number of a_i so that $a_i\&x = x$.
- Let $bit_j(u)$ is the state of the j-th bit of a non-negative integer u.
- For queries type 1, let a_i satisfy $a_i\&t = x$, it is proven that:
  - If $bit_j(t) = 1$, then $bit_j(a_i) = bit_j(x)$.
  - If $bit_j(t) = 0$, then $bit_j(x) = 0$
- We will count the number of $1-$stated bit of t (cnt_1), and also the number of its $0-$stated bit (cnt_0).
  - If $cnt_1 < cnt_0$, then inclusion-exclusion technique is used with the array dp_2 for every $0-$stated bit position.
  - Otherwise, we will traverse all the $0-$stated bit position.

Problem E: A very hard problem

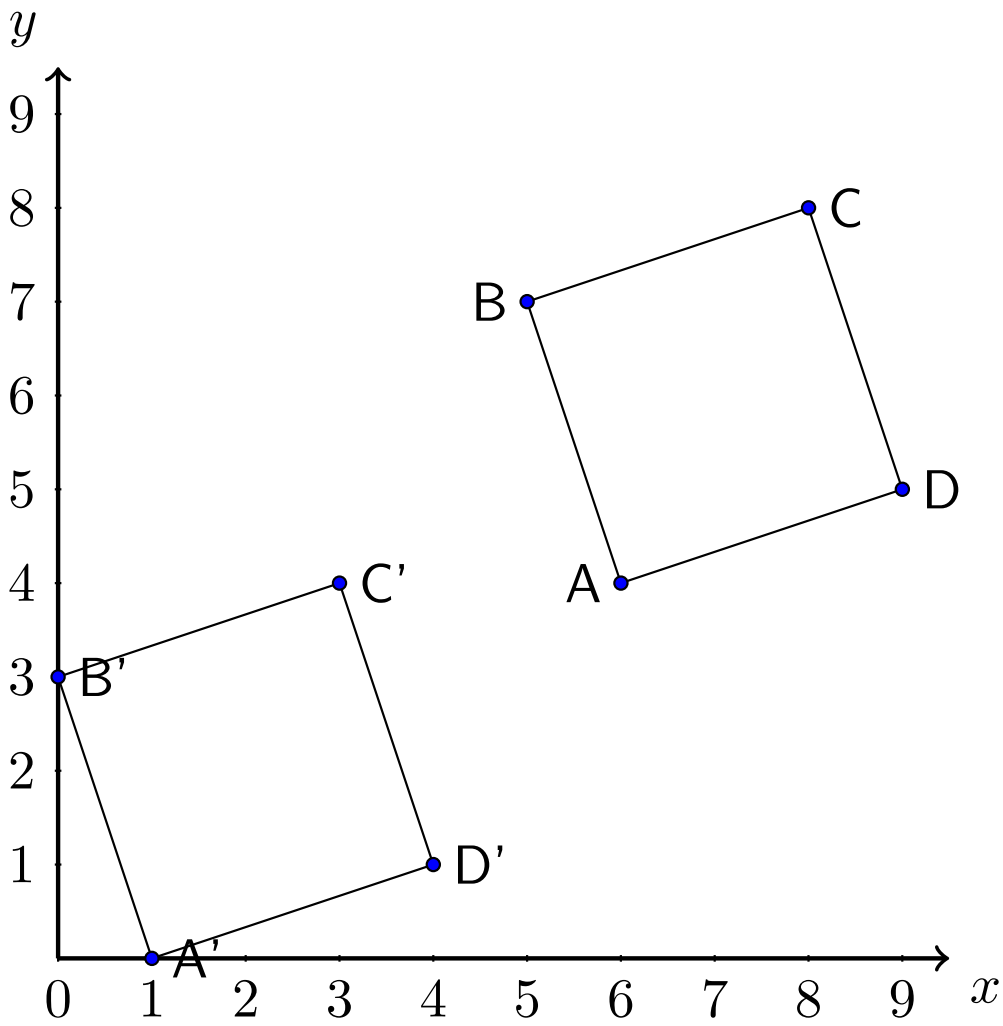
- Queries type 2 could be processed similar to type 1.
- For queries type 3 and 4, with dp_1, dp_2 and some technique and combinatorics formula, they could be processed. However, the problem requires to solve the queries for every k satisfied $l \leq k \leq r$, so please visit `https://stackoverflow.com/a/29440908` to refer to the method.
- Time complexity: $\mathcal{O}\left(2^p \times p + q' \times 2^{\frac{p}{2}} \times \frac{p}{2} + n + q\right)$, which q' means the number of queries type 1 and 2, $p = 20$ is the maximum number of bit in binary representation of a_i.

Problem F: Massive Mission

Problem F: Massive Mission

- It is very difficult if you want to find the rectangles directly from the information given by the problem.
- If we translate them to the original coordinate, it is clears to find the coordinates for 4 boundary points.

Problem F: Massive Mission



It might be very difficult if we try to find the coordinates of A, B, C and D. However, simple geometry formula could be used to find the A', B', C' and D',

Problem F: Massive Mission

- After that process, we will have a set of $4n$ points, then we should find the Convex Hull made from this set. Graham Algorithm or Mono chain technique (a.k.a Andrew Algorithm) are popular methods.
- Let $S_i(x_i, y_i)$ be the n elements of the Convex Hull. Then its area is $S = \frac{1}{2}\sum_{i=1}^{n} |x_i y_{i+1} - x_{i+1} y_i|$. Let $S_{n+1} \equiv S_1$.
- The final answer is $\frac{S_{rectangles}}{S}$.
- Time complexity: $O(n \log n)$, the complexity of creating the Convex Hull.

Problem G: Database

Problem G: Database

Naive approach for queries type 1 and 2:

- If we decided to create a new string after each query, it would cost a lot of memory.
- In the worst cases, the length of the new-made string could up to 10^5. Moreover, there are 3×10^5 queries $\Rightarrow$ Sum of length of all strings are $3 \times 10^{10} \Rightarrow$ **MEMORY LIMIT EXCEEDED**.

An optimized solution for this problem:

- For every queries type 1 and 2, a new string is made by reusing the prefix of another string which has been done before $\Rightarrow$ Use **TRIE** data structure.
- To simplify the type 3 queries, we could process all queries type 1 and 2 before processing type 3 queries.

Problem G: Database

- After that, we have completed a TRIE, then Euler Tour and Segment Tree is used to store and find the index of each node.
- For each type 3 query, we will traverse the TRIE (1) to the node which is connected to make the string s, then use Segment Tree Get function to check if there was an index that satisfy the problem condition.
- The process (1) could be up to the maximum length of a string, however, sum of length of all strings is up to 10^6, so that will process in-time for this problem.
- Time complexity: $\mathcal{O}(\sum |s_i| \log q)$.

Problem H: Mr. Courage

Problem H: Mr. Courage

- DFS could be used to check every connected component. It is clear that the count of edges which we need to add in the graph to get the odd cycle is no more than three.
- Answer to this problem is 3 if the count of edges in the graph is zero. Then the number of ways to add three edges in the graph to make an odd cycle is equal to $\frac{n(n-1)(n-2)}{6}$ where n is the number of vertices in the graph.
- Answer to this problem is 2 if there is no connected component with the number of vertices more than two. Then the number of ways to add two edges in the graph to make an odd cycle is equal to $m(n-2)$ where m is the number of edges in the graph.

Problem H: Mr. Courage

- Now we have one case when there is at least one connected component with a number of vertices more than two. Now we need to use dfs and try to split every component in two parts. If for some component we can't do it, that means that the graph already has an odd cycle, so the number of ways is 1 (that's do nothing).
- If all connected components in the graph are bipartite then we need to iterate on them. Let cnt_1 is the count of vertices in one part of the current component and cnt_2 is the number of vertices in the other part. If number of vertices in this component more than two, we need to add to answer $\frac{cnt_1(cnt_1-1)}{2}$ and $\frac{cnt_2(cnt_2-1)}{2}$.
- Time complexity: $O(n+m)$, due to the complexity of the DFS Algorithm.

Problem I: Cows

Problem I: Cows

- We need to calculate $\sum_{i=1}^{n} \frac{i(i+1)}{2}$
- We could prove that
  $$\sum_{i=1}^{n} \frac{i(i+1)}{2} = \sum_{i=1}^{n} \frac{i^2}{2} + \sum_{i=1}^{n} \frac{i}{2} = \frac{1}{2}\left[\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right]$$
  $$= \frac{n(n+1)(n+2)}{6}$$
- To calculate the answer modulo $10^9 + 7$, there are several math tricks:
  - In 3 consecutive integers, it always exists a multiple of 2 and a multiple of 3 $\Rightarrow$ Divide before calculating the product.
  - Use inverse modulo.
- Or use Python.
- Time complexity: $O(1)$.

Problem J: Consecutive Floods

Problem J: Consecutive Floods

- Let $E[x, y, z]$ be the expected value of number of drawn tickets when taking x notebook tickets, y book tickets, and z pen tickets.
- Then

$$E[x, y, z] = \begin{cases} E[x+1, y, z] + 1, & \text{with probability } \frac{x}{x+y+z} \\ E[x, y+1, z] + 1, & \text{with probability } \frac{y}{x+y+z} \\ E[x, y, z+1] + 1, & \text{with probability } \frac{z}{x+y+z} \end{cases}$$

- Init $E[100, y, z] = E[x, 100, z] = E[x, y, 100] = 0$
- Time complexity: $\mathcal{O}(n^3)$

Problem K: Quick Sort

Problem K: Quick Sort

- Quick Sort fall into worst case when it picks Largest or Smallest value every time $\leftarrow$ We should place largest value in the middle
- **Hoare partition** swaps the largest value with the last position of the array every time.
- The second largest value should be in the end.
- Repeat until filled.
- Time complexity: $\mathcal{O}(n)$.

Problem L: Stranger from multiverse

Problem L: Stranger from multiverse

- Because Khang only uses the shortest paths, we could easily recognize that Khang only moves up and right and never turns back on his way.
- Let $g(r, c)$ be the sum of $f(i, j)$ for all pairs of integers (i, j) such that $0 \leq i \leq x$ and $0 \leq y \leq c$. Then, the desired answer is $g(x_2, y_2) - g(x_2, y_1 - 1) - g(x_1 - 1, y_2) + g(x_1 - 1, y_1 - 1)$.
- Therefore, this problem can be solved if $g(x, y)$ can be calculated fast.

Problem L: Stranger from multiverse

- It is a bit kind of sudden, but it holds that $f(x+1,y) = \sum_{i=0}^{y} f(x,i)$.

  This can be explained by considering the paths from the point $(0,0)$ to the point $(x+1,y)$. Every path from the point $(0,0)$ to the point $(x+1,y)$ is either exactly one of such paths that:
  - From the point $(0,0)$, reach the point $(x,0)$, and move to $(x+1,1)$ right away, then move to $(x+1,y)$
  - From the point $(0,0)$, move to the point $(x,1)$, and move to $(x+1,1)$ right away, then move to $(x+1,y)$
  - From the point $(0,0)$, move to the point $(x,2)$, and move to $(x+1,2)$ right away, then move to $(x+1,y)$
  - ...
  - From the point $(0,0)$, move to the point (x,y), and move to $(x+1,y)$ right away, then move to $(x+1,y)$

Problem L: Stranger from multiverse

- By the way, the number of paths from the point $(0, 0)$ to the point $(x + 1, y)$ i $f(x + 1, y)$, and the number of the path such that, from the point $(0, 0)$, move to the point (x, i), and move to $(x + 1, i)$ right away, then move to $(x + 1, y)$ can be calculated as follows:
  - There are $f(x, i)$ paths such that "from the point $(0, 0)$, move to the point (x, i) is $f(x, i)$, then move downwards once, and you are forced to move to the right all the way, so there is a single path from the point (x, i) and beyond, thus the number of paths is $f(x, i)$. Therefore, it holds that $f(x + 1, y) = f(x, 0) + f(x, 1) + \cdots + f(x, y)$.
- Also, it holds that $f(x, y) = \binom{r + c}{c} = \frac{(r + c)!}{r!c!}$. Therefore, the desired answer can be denoted by the sum of $f(x, y)$, whose number of terms are $\mathcal{O}(n)$, so it is enough if $f(x, y)$ can be calculated in $O(1)$ time. To do this, you can precalculate the factorials and the inverses modulo of factorials.
- Also, by applying $f(x, y + 1) = f(0, y) + f(1, y) + ... + f(x, y)$ again, the desired answer can be represented by the sum of a constant number of $f(x, y)$.
- Time complexity: $\mathcal{O}(n \log p)$, with $n = \max(x_2, y_2)$ and $p = 10^9 + 7$, due to the complexity of linear traverse and reverse modulo calculation.

Conclusion

Conclusion

Some information about this contest:

- Seven problems were solved. The champion is UIT_Noldea from VNUHCM-UIT, which completed 5 problems in 432 min. (penalties included).
- The first accepted submission of the contest is from team Byte_the_Bit (VNUHCM-US) in the 10th min. for Problem A.
- 1336 submissions were made, however only 127 of them were correct (9.5%).
- Most solved problems are B (71 teams) and I (30 teams). No team has completed the problem E,F,G,H,L.

Conclusion

First solver of each problem:

- Problem A (10th min.): UIT_NoIdea from VNUHCM-UIT.
- Problem B (16th min.): ChickenCanFly from VNUHCM-US.
- Problem C (88th min.): TNT from VNUHCM-UIT.
- Problem D (105th min.): UIT_NoIdea from VNUHCM-UIT.
- Problem I (24th min.): Rainbow_Shine from VNUHCM-US.
- Problem J (60th min.): UIT_NoIdea from VNUHCM-UIT.
- Problem K (214th min.) TooGoodToAC from Tran Dai Nghia High School for the gifted.

Thank you!

See you next year.