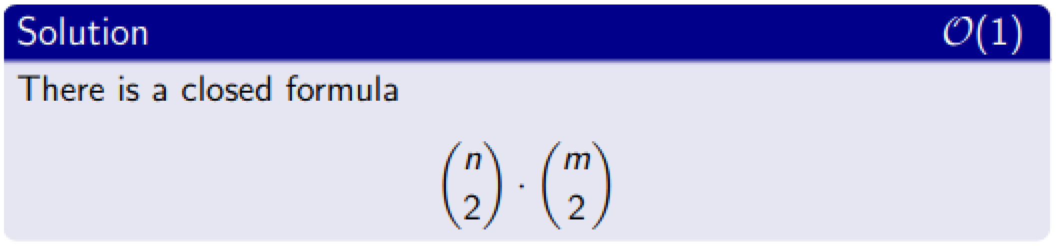
Fall 24 Div Compete Week 2 - Solutions

(taken from editorials of original problems)

Problem A

(Source: ETH Zurich Competitive Programming Contest Spring 2024 Problem C)



Solution $\mathcal{O}(1)$

There is a closed formula

$$\binom{n}{2} \cdot \binom{m}{2}$$

Problem B

(Source: Dynamic Programming, SPbSU 2024, Training 1 Problem D)

Dynamic programming on subsegments. The state consists of left and right boundaries of the segment with the remaining cups. The number of consumed cups can be uniquely restored from the boundaries. The transitions are to drink the leftmost or the rightmost of the remaining cups. It is convenient to have an outer loop over the length of the segment.

Here is the DP function, for an interval i, j (meaning we have used the first i and last j cups):

$$f_{i,j} = \max\{(n-i-j) \cdot a_i + f_{i+1,j}, (n-i-j) \cdot a_j + f_{i,j-1}\}$$

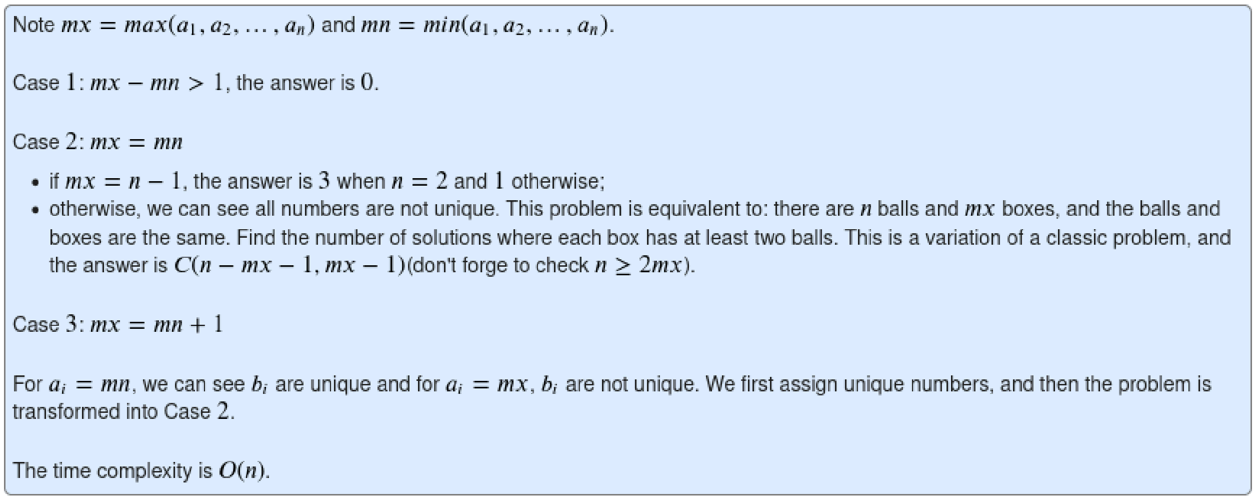
Problem C

(Source: Cupertino Informatics Tournament Online Mirror Problem A)

The solution is just *n/12*. Note that the expectation of the night movement is to stay in the same place.

Problem D

(Source: Theforces Round #34 (ABC-Forces) Problem E)



Note $mx = max(a_1, a_2, \dots, a_n)$ and $mn = min(a_1, a_2, \dots, a_n)$.

Case 1: $mx - mn > 1$, the answer is 0.

Case 2: $mx = mn$

- if $mx = n - 1$, the answer is 3 when $n = 2$ and 1 otherwise;
- otherwise, we can see all numbers are not unique. This problem is equivalent to: there are n balls and mx boxes, and the balls and boxes are the same. Find the number of solutions where each box has at least two balls. This is a variation of a classic problem, and the answer is $C(n - mx - 1, mx - 1)$(don't forge to check $n \geq 2mx$).

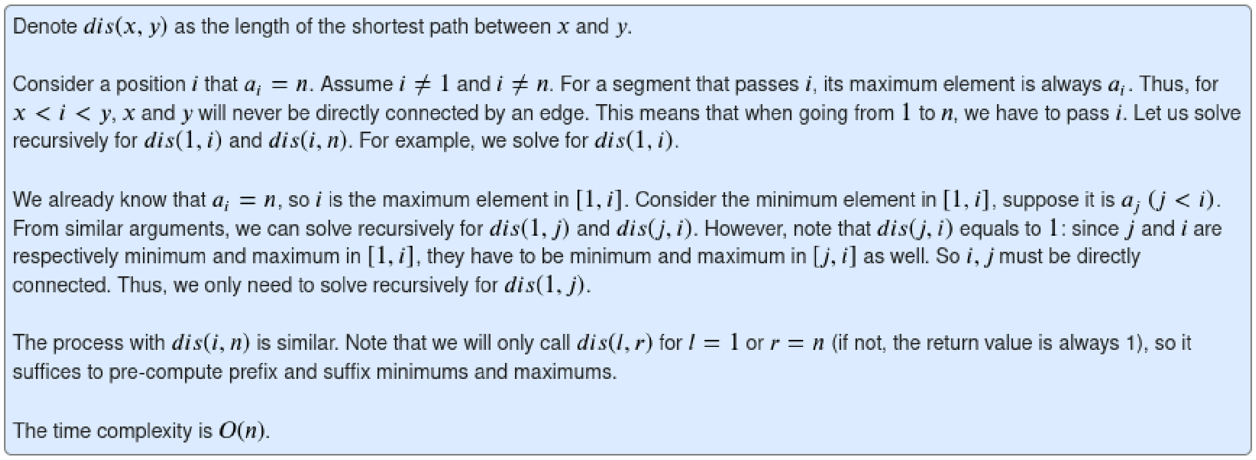
Case 3: $mx = mn + 1$

For $a_i = mn$, we can see b_i are unique and for $a_i = mx$, b_i are not unique. We first assign unique numbers, and then the problem is transformed into Case 2.

The time complexity is $O(n)$.

Problem E

(Source: Codeforces Global Round 21 Problem D)



Denote $dis(x, y)$ as the length of the shortest path between x and y.

Consider a position i that $a_i = n$. Assume $i \neq 1$ and $i \neq n$. For a segment that passes i, its maximum element is always a_i. Thus, for $x < i < y$, x and y will never be directly connected by an edge. This means that when going from 1 to n, we have to pass i. Let us solve recursively for $dis(1, i)$ and $dis(i, n)$. For example, we solve for $dis(1, i)$.

We already know that $a_i = n$, so i is the maximum element in $[1, i]$. Consider the minimum element in $[1, i]$, suppose it is a_j $(j < i)$. From similar arguments, we can solve recursively for $dis(1, j)$ and $dis(j, i)$. However, note that $dis(j, i)$ equals to 1: since j and i are respectively minimum and maximum in $[1, i]$, they have to be minimum and maximum in $[j, i]$ as well. So i, j must be directly connected. Thus, we only need to solve recursively for $dis(1, j)$.

The process with $dis(i, n)$ is similar. Note that we will only call $dis(l, r)$ for $l = 1$ or $r = n$ (if not, the return value is always 1), so it suffices to pre-compute prefix and suffix minimums and maximums.

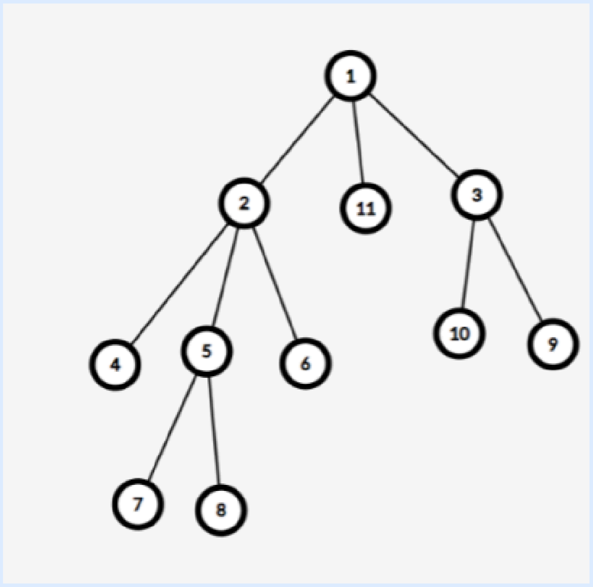
The time complexity is $O(n)$.

Problem F

(Source: Theforces Round #34 (ABC-Forces) Problem G)

Use a variant of the Euler tour. Let's call it BFS-children-first-DFS.

First, node 1 is added to the sequence, then $DFS(1)$ is performed. When $DFS(x)$ is called, all child nodes of x are added to the sequence, and then $DFS(son[x][i])$ is performed.



For example, for the tree shown above, the BFS-children-first-DFS order is $[1, 2, 11, 3, 4, 5, 6, 7, 8, 10, 9]$.

In this way, the subtree rooted at node x (excluding x itself) and all the child nodes of node x are within a contiguous segment of the sequence. This makes it easy to maintain the operations in $O(nlogn)$.