UIUC ICPC Spring Coding Contest 2019 Editorial

A - AK the Problems

This problem is not very hard but requires a small leap of faith or good game-theory intuition.

We denote the state of a forest using the parity of vertices and the parity of edges, respectively. Therefore, there are four cases we need to discuss. When the game stops, it is on $(even, even)$; thus, if we can prove that $(even, even)$ can't remain in $(even, even)$ after one move and the other states can always become $(even, even)$ after one move, we can know that $(even, even)$ is a $P-position$ and the others are $N-positions$. Here is the proof:

1. $(even, even)$:
   There isn't any move that can delete even vertices and even edges at the same time. Thus, $(even, even)$ can't remain in $(even, even)$ after one move.

2. $(even, odd)$:
   The parity of the edges is odd. Therefore, there exists at least one edge. We can delete the edge and the state becomes $(even, even)$.

3. (odd, odd):
   After any moves, a forest is still a forest. Since the state is (odd, odd), there exists at least a leaf (the vertex whose degree is 1) in the forest. We can delete this vertex and the state becomes $(even, even)$.

4. $(odd, even)$:
   The degrees sum formula in graph theory implies that the number of vertices with odd degree is even. Therefore, if the forest is in $(odd, even)$, the number of vertices with even degree is odd. That is to say, there exists at least a vertex with even degree. We can delete this vertex and the state becomes $(even, even)$.

In conclusion, since only $(even, even)$ is the $P-position$, if the intial forest is $(even, even)$, you should output "Suzukaze loses all his ratings!". Otherwise, you should output "Suzukaze becomes a grandmaster!".

B - Bigram Language Model

For each query (s, t), the nominator is the number of co-occurrences of (s, t) in the corpus, and the denominator is the number of occurrences of s as non-terminal word in a sentence. These values can be pre-processed by a single pass over the corpus.

The tricky part is that the number of distinct words appearing in the corpus w may be up to 10^5, so it is impossible to explicitly construct the co-occurrence matrix of size $w \times w$. The observation is that the sum of all elements in this matrix is bounded by the size of corpus, so we can use some sparse-matrix representation techniques (e.g. only recording which entries are non-zero and the values they hold).

C - Construct Underground System

D - Diameter

F - Fruit on the Tree

The core of this problem is asking you how many three vertices sets satisfy "any of the vertex in the set is not on the simple path of the other two vertices". Therefore, the weights do not matter. However, this number is hard to compute directly. One way to compute it is to subtract the number of sets satisfy "one of the vertex in the set is on the simple path of the other two vertices" from the total number of sets to get the answer. We just need to DFS one time on the tree and compute how many ways there are to form a set that the current vertex is on the simple path of the other two vertices for each node, which is equal to:

$$\sum_{1 \le i < j \le m} |V_i||V_j|$$

where m is the number of subtrees if we use the current vertex to be the root and $|V_i|$ is the size of the ith subtree. This can be done in $O(n)$.

G - Greenberg Mass Comparison

The core of this problem is asking you how many ways are there to partition a set of n elements into disjoint, non-empty subsets. If you know Bell number, that's exactly what we are asking for (but you still have to do the programming). If not, not a problem! Let's work out a recurrence formula for it.

Let B_n denote the answer for n. Consider the number of elements **NOT** in the subset containing element 1. If there are k such elements, then there are $\binom{n-1}{k}$ ways to choose these elements, and we can just ignore the subset containing 1 and do the partition on these k elements. Therefore,

$$B_n = \sum_{k=0}^{n-1} \binom{n-1}{k} B_k$$

with base case $B_0 = 1$.

Notice that Bell numbers can get very large quickly, so coding this problem in Python or Java will save you a lot of trouble.

J - Juicy