Problema CountBST - Soluție Oficială

Autor: Student Marian Darius, Student Adrian Budău

Prima observație este că numărul de arbori binari de căutare cu **N** noduri fără restricția de lanț este numărul Catalan de ordin **N**. Acestea pot fi calculate după formula:
C_N = Comb(2N, N)/(N+1) = (2N)!/N!/(N+1)! = Fact[2N] * InvFact[N] * InvFact[N+1] % MOD.
Acestea pot fi precalculate în diverse complexități, însă pentru 100 de puncte este necesară o implementare în ***O(N)***.
Se observă mai întâi că dacă lanțul dat are valorile **A_1, A_2, … A_k** în ordine crescătoare, oricum apare lanțul în arbore, mulțimea de valori **{A_1+1, A_1+2 … A_2 - 1}** va fi așezată în BST într-un sub-arbore compact, legat fie ca fiu stânga al nodului ce conține **A_2** fie ca fiu dreapta al nodului ce conține **A_1**, dar în orice ordine: deci putem ignora complet aceste valori și să înmulțim răspunsul global cu **CATALAN[A_2 - A_1 - 1]**. La fel putem face cu valorile **{A_2+1, A_2+2 … A_3 - 1} … {A_{k-1}+1, A_{k-1}+2 … A_k - 1}**. Ne rămân astfel de numărat formele posibile ale lanțului în sine, cât și prefixul **1, 2,...A_1 - 1** și sufixul **A_k + 1, A_k+ 2 … N**. Putem să ne imaginăm porțiunea formată din numerele de la **A_1** la **A_k** ca un singur nod, și atunci aranjarea prefixului și sufixului în jurul său va fi determinat de formula
CATALAN[prefix **+** sufix **+ 1] = CATALAN[(A_1 - 1) + (N - A_k) + 1] = CATALAN[N - A_k + A_1]**.
Pentru numărul de aranjări ale lanțului în sine, se observă că dacă fixăm una din valori ca rădăcina sub-arborelui ce conține lanțul, restul vor fi în totalitate unic determinate, dacă **se poate**. Deci defapt ne interesează în câte din cele **K** valori poate fi „agățat” lanțul pentru a forma un arbore binar valid.

Astfel, formula finală a problemei este:
CATALAN[N - A_k + A_1] * CATALAN[A_2 - A_1 - 1] * CATALAN[A_3 - A_2 - 1] *...* CATALAN[A_K - A_{k-1} - 1] * Rădăcini

Numărul de rădăcini valide se poate calcula trivial în ***O(K^2)***, însă pentru 100 de puncte este necesară o soluție mai eficientă:
Începem să construim un lanț de jos în sus, de la **A_1** spre dreapta. Când ajungem la un număr ce nu mai poate fi adăugat în sus fără să compromită proprietatea de arbore de căutare, ne oprim. Astfel, am aflat prefixul de lungime maximă al șirului care poate fi un lanț „în jos” într-un BST. Facem același lucru și pentru sufix. Acum, ca un element **A_i** să poată fi rădăcină, trebuie să se respecte trei condiții:

1. prefixul de la **A_1** la **A_i** să fie un lanț valid „în jos”
2. sufixul de la **A_i** la **A_k** să fie un lanț valid „în jos”
3. Elementele **A_{i-1}, A_i, A_{i+1}** trebuie să formeze o secvență monotonă (pentru a permite elementului **A_i** să își construiască cele două lanțuri în jos în sub-arbori diferiți)

Astfel, este nevoie de ***O(K$\log_2$K)*** timp pentru a rezolva un test după pregenerarea numerelor Catalan. În total, complexitatea asimptotică pentru 100 de puncte este ***O(N+T*K$\log_2$K)***.

S-au punctat diverse soluții parțiale pentru rezolvarea diverselor sub-probleme ale problemei cu dinamică în loc de combinatorică (de exemplu, numărul de moduri de a aranja prefixul și sufixul se poate rezolva precalculând dinamica **dp[i][j]** = numărul de moduri de a aranja un prefix de lungime **i** și un sufix de lungime **j** în jurul lanțului cerut. Această dinamică poate avea recurență atât în ***O(N)*** cât și în ***O(N^2)***, dând un timp total de precalculare ***O(N^3)*** respectiv ***O(N^4)***).