Εισαγωγή

Το πρόβλημα αυτό προέρχεται από τον διαγωνισμό ACM-ICPC Northeast North America Regional Contest 2017

Ζητούμενο: Έστω ένα δέντρο με γράμματα στις κορυφές του. Διατρέχουμε το δέντρο αυτό με τον αλγόριθμο DFS, και διατηρούμε τα γράμματα των κόμβων με τη σειρά που τους επισκεπτόμαστε (ξαναγράφοντας το γράμμα ενός κόμβου για κάθε φορά που επανερχόμαστε σε αυτόν). Δεδομένης μιας σειράς γραμμάτων, να βρεθεί το πλήθος των πιθανών δέντρων που μπορεί να την παρήγαν.

Προαπαιτούμενα: Αν και το πρόβλημα αυτό διατυπώνεται πολύ εύκολα σε ένα δέντρο, γνώσεις θεωρίας γράφων δεν είναι απαραίτητες για τη λύση του. Το πρόβλημα κατατάσσεται στην κατηγορία του Δυναμικού Προγραμματισμού (Dynamic Programming).

Το πρόβλημα αυτό έλυσαν δύο άτομα πλήρως, ενώ ένα άτομο έκανε μερική λύση.

Λύση σε $O(n^3)$

Ας θεωρήσουμε έναν "υπολαβύρινθο" του λαβυρίνθου στον οποίο εισέρχεται ο Μίλτος (δηλαδή ένα μέρος του λαβυρίνθου στο οποίο εισέρχεται σε κάποια στιγμή μέσω ενός δωματίου (το οποίο χαρακτηρίζεται ως η αρχή αυτού του υπολαβυρίνθου) και εξέρχεται μέσω του ίδιου δωματίου (καθώς ο λαβύρινθος δεν έχει κύκλους). Τότε παρατηρούμε ότι μπορούμε να βρούμε ένα σημείο στις σημειώσεις του Μίλτου το οποίο περιγράφει από μόνο του έναν λαβύρινθο. Μάλιστα, θα δούμε ότι εάν λύσουμε το πρόβλημα για όλους τους υπολαβυρίνθους, τότε μπορούμε να ενώσουμε τις απαντήσεις για να πάρουμε τη συνολική λύση.

Η κειμενοσειρά που παράγει ένας υπολαβύρινθος πρέπει πάντα να αρχίζει και να τελειώνει με το ίδιο χρώμα (καθώς το τελευταίο δωμάτιο είναι και το πρώτο). Έπειτα, εάν αυτός ο υπολαβύρινθος αποτελείται μόνο από ένα δωμάτιο, η κειμενοσειρά που θα παράξει περιέχει μόνο ένα γράμμα. Αλλιώς, εάν αφαιρέσουμε το πρώτο και το τελευταίο γράμμα, θα πρέπει η κειμενοσειρά που παίρνουμε να περιγράφει είτε έναν, είτε παραπάνω υπολαβυρίνθους. Καθώς ένας υπολαβύρινθος στην κειμενοσειρά ξεκινά και τελειώνει με τον ίδιο χαρακτήρα, μένει μόνο να θεωρήσουμε όλες τις πιθανές περιπτώσεις για τον πρώτο υπολαβύρινθο που επισκεπτόμαστε.

Έστω ένας δισδιάστατος πίνακας dp, έτσι ώστε το $dp[i][j]$ να περιγράφει το πλήθος των πιθανών λαβυρίνθων που μπορεί να δημιούργησαν την κειμενοσειρά από τη θέση i έως τη θέση j. Έστω επίσης str η κειμενοσειρά που μας δίνει ο Μίλτος. Οπότε έχουμε τις εξής παρατηρήσεις:

- $dp[i][i] = 1$ για κάθε i.
- Αν η κειμενοσειρά $str[i \dots j]$ δεν περιγράφει λαβύρινθο, τότε $dp[i][j] = 0$.
- $dp[i][j] = \sum_k dp[i+1][k-1] * dp[k][j]$ για κάθε k έτσι ώστε $str[i] = str[k]$ και $i < k \leq j$.

Συνεπώς η απάντηση θα βρίσκεται στο $dp[0][N-1]$ (χρησιμοποιώντας το 0 ως αρχή).

Ακολουθεί η ενδεικτική υλοποίηση σε `c++`.

```c++
#include <bits/stdc++.h>
#define MAXN 2048

unsigned long long dp[MAXN][MAXN];

char inorder[MAXN];
```

```
6
7  int main() {
8      int N;
9      scanf("%s", inorder);
10     N = strlen(inorder);
11
12     for(int i=0; i<N; ++i) {
13         dp[i][i] = 1;
14     }
15     for(int j=2; j<N; j+=2) {
16         for(int i=0; i+j<N; ++i) {
17             if(inorder[i]==inorder[i+j]) {
18                 dp[i][i+j] = dp[i+1][i+j-1];
19                 for(int k=2; k<j; k+=2) {
20                     if(inorder[i] == inorder[i+k]) {
21                         dp[i][i+j] += dp[i+1][i+k-1]*dp[i+k][i+j];
22                     }
23                 }
24             }
25         }
26     }
27     printf("%llu\n", dp[0][N-1]);
28     return 0;
29 }
```