Sablon - Descrierea soluției

Autor prof. Radu Vișinescu, C.N. "I.L. Caragiale" Ploiești

Soluție 32 puncte (prof. Radu Vișinescu):

Se programează o funcţie recursivă care decide dacă un şir de caractere S se potriveşte cu o expresie şablon E. Cazurile se separă în funcţie de primul caracter al expresiei E respectiv al şirului S. Exista şi un caz în care se execută mai multe apeluri recursive şi anume la întâlnirea caracterului * în expresia şablon E. Funcția se apelează din programul principal pentru fiecare valoare din șirul de cuvinte.

Soluție 48 puncte (prof. Daniel Popa):

Se transformă șablonul într-o expresie regulată care este folosită pentru căutare folosind libraria regex. Avantaj: Cod scurt. Dezavantaj: funcționează doar sub Linux, se consumă multă memorie, algorimul implementat în librărie nu este foarte performant.

Soluție 64 puncte (student- Mihai Nițu):

Pentru a verifica daca un cuvant se potriveste cu un sablon dat, se poate utiliza un algoritm greedy sau o relatie de recurenta. Algoritmul greedy consta in verificarea daca capetele sablonului (prefixul si sufixul pana la '*') coincide cu capetele cuvantului, si apoi cautarea pe rand a fiecarei subsecvente din sablon cuprinsa intre '*' in cuvant. Daca acestea apar ca subsecvente in cuvant, in ordinea in care apar in si in sablon si nu se intersecteaza cu bucatile de care am stability ca vor corespunde capetelor sablonului, atunci cuvantul se potriveste cu sablonul.

Solutia alternativa este data urmatoarea relatie de recurenta

Match[i][j] = 1 daca sufixul [j..m] din cuvantul c se potriveste cu sufixul [i..n] din sablonul s.

Match[n+1][m+1] = 1

Match[i][j] = Match[i+1][j+1] daca s[i] == '?' sau s[i] – literasi s[i] == c[j]
= 0, daca s[i] – literasi s[i] != c[j]
= Match[i+1][j] ‖ Match[i+1][j+1] ‖ … Match[i+1][m+1], daca s[i] =≠ '*', unde ‖ este 'sau' logic. -> se pot tine 'sau'-uri partiale pe fiecare linie a matricei Match pentru a obtine valoarea in O(1).

Pentru a tine cont de posibiliatea eliminarii unei litere, putem sa aplicamalgoritmul, pentru sablonurile obtinute prin eliminarea fiecarei litere. Complexitatea finala va fi O(nr. Cuvinte * N^3) cu ambele abordari. Pentru algoritmul greedy, nu putem folosi KMP din cauza caracterelor ?, deci vom folosi algoritmul de potrivire brut. Aceasta solutie poate fi dificila de implementat dar poate obtine chiar 100 puncte deoarece algoritmul brut de potrivire se comporta foarte bine in practica.

Soluție 100 puncte (student - Mihai Nițu):

Putem sa imbunatatim, spre exemplu, solutia prin recurenta, pentru a numai fi nevoie să o recalculam de fiecare data cand eliminam o litera. Astfel, vom calcula aceeasi recurenta de 2 ori, Match1 si Match2. Diferenta dintre ele este ca Match1 va verifica potrivirea intre un sufix al sablonului si un sufix al cuvantului iar Match2 va verifica potrivirea intre un prefix al sablonului si un prefix al cuvantului. Putem, apoi, sa verificam pentru fiecare pozitie din sabloni, pe care exista o litera, daca ok2[i-1][j] = 1 si ok1[i+1][j+1] = 1, pentru oricare j de la 0 la m. Daca cel putin una se evalueaza la 1, atunci avem o potrivire. Complexitate: O(nr. Cuvinte * N^2).