Livada – soluţie (livada.cpp)

Evident, cele două cerinţe se rezolvă în paralel, pentru fiecare rând.

Pentru rezolvarea primei cerinţe am folosit un contor **rsm** şi apoi, pentru fiecare rând, am verificat dacă el are soi majoritar sau nu folosind următorul algoritm:

- considerăm prima valoare de pe rândul respectiv ca fiind posibilul soi majoritar (variabila **sm**) şi iniţializăm un contor **cnt** (care numără de câte ori se găseşte posibilul soi majoritar de pe acel rând cu 1;
- parcurgem restul rândului şi ori de câte ori găsim un soi egal cu **sm** creştem valoarea lui **cnt** cu 1 (creşte probabilitatea ca **sm** să fie soi majoritar) şi ori de câte ori găsim un soi diferit scădem valoarea lui **cnt** cu 1 (scade probabilitatea ca **sm** să fie soi majoritar) . Dacă la un moment dat **cnt** devine 0, atunci reiniţializăm soiul majoritar cu soiul curent, iar **cnt** devine 1.
- după ce terminăm de parcurs rândul curent verificăm dacă valoarea lui **cnt** este 0 sau nu. În caz afirmativ înseamnă că nu avem soi majoritar pe rândul respectiv. Altfel, dacă valoarea lui **cnt** este diferită de 0, înseamnă că în **sm** avem cel mai bun candidat pentru soiul majoritar şi verificăm dacă el este într-adevăr soi majoritar, reparcurgând rândul respectiv. Acest lucru este obligatoriu, pentru că în cazul rândului 1,3,1,2,3,3,3 contorul **cnt** va fi 3, iar soiul majoritar este într-adevăr, soiul 3. În schimb, în cazul rândului 1,2,1,2,3,3,3 contorul va fi tot 3, fără ca soiul 3 să fie majoritar!
- dacă rândul curent are soi majoritar, creştem valoarea contorului **rsm** cu 1.

Pentru rezolvarea celei de-a doua cerinţe am folosit un contor **max** care va păstra maximul lungimilor celor mai lungi secvenţe ce au proprietatea cerută pe fiecare rând. Algoritmul folosit este unul clasic, în care se folosesc doi indecşi (j şi k) cu care se parcurge fiecare rând, element cu element. Dacă valoarea lui v[k] este egală cu valoarea lui v[j], atunci se creşte valoarea lui k. În momentul în care se termină o secvenţa formată din copaci de acelaşi soi ($v[j] \neq v[k]$), se verifică dacă lungimea ultimei secvenţe este mai mare decât valoarea maximă obţinută până în acel moment (variabila **rmax**) sau nu.

Fiecare dintre cele două cerinţe a fost rezolvată folosind algoritmi cu complexitatea O(n), deci complexitatea soluţiei este O(m*n). Observaţi că această comlexitate nu depinde de valoarea lui p!

Există şi alte posibilităţi de rezolvare ale acestei probleme, care pot obţine punctaje parţiale sau chiar 100 de puncte:

1. fişierul "livada_v1.pas" conţine o variantă de 100 de puncte bazată pe vectori de contorizare, alocaţi eficient (din precizările din enunţul problemei se deduce uşor că un vector de contorizare poate avea cel mult 250.000 de elemente)
2. fişierul "livada_v2.cpp" conţine o soluţie bazată pe sortarea unui rând (folosind funcţia predefinită qsort) şi verificare faptului ca elementul din mijlocul rândului este soi majoritar sau nu (se observă că dacă soiurile sunt sortate crescător, atunci soi majoritar nu poate fi decât cel aflat pe poziţia din mijloc);
3. se pot da soluţii de peste 50-70 de puncte folosind un algoritm asemănător sortării prin numărare în care să se numere de câte ori apare fiecare soi pe un rând.