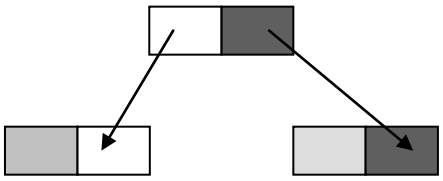
Culori – Soluţie (Berinde Radu, Dăneţ Tiberiu)

Problema ar fi evidentă fără limitare de memorie (am ţine arborele bucăţilor). Trebuie să găsim un mod eficient de a reţine informaţiile astfel încât să fie posibilă aflarea culorii unei bucăţi de ţeavă care a fost tăiată. Pentru asta, vom ţine două culori pentru fiecare bucată de ţeavă. Una este culoarea ei reală, iar una este o culoare „de moştenire". Când o ţeavă este tăiată, ea are două valori care vor fi reţinute în cele două valori "de moştenire" ale celor două bucăţi noi.



Astfel când trebuie să unim două bucăţi putem reconstrui tatăl complet pentru că ştim cele două valori ale lui. Aceste valori pot fi ţinute în doi vectori – **C1** şi **C2**, ambii de mărime **N**. Mai avem nevoie de un vector **Len[i]** care ne spune lungimea blocului care începe la poziţia **i**. Astfel memoria totală este de **N*(1+1+4) = 6*N = 3 Megabytes**.

Bec – Soluţie (Dăneţ Tiberiu, Berinde Radu)

Problema se rezolvă cu back. O implementare naivă ar avea complexitatea $O(2^N*M)$. Aceasta poate fi îmbunătăţită la $O(2^N)$ prin folosirea unor măşti de biţi pentru fiecare linie (o linie poate fi identificată prin 3 întregi de 32 de biţi). Numărarea biţilor dintr-o mască se poate face în O(1) prin precalcularea unui vector de mărime 2^{16}. A doua îmbunătăţire majoră este folosirea unei euristici: liniile vor fi sortate iniţial crescător după numărul de becuri aprinse de pe fiecare linie şi la fiecare nivel din back se estimează aria maximă care ar putea fi obţinută în continuare.
Evident, la început dacă **M > N**, liniile se interschimbă cu coloanele.
Programul comisiei rulează în mai puţin de 18 minute pe cele 10 teste propuse.

```
void compute (int line, int selected, unsigned int mask1, unsigned int mask2,
unsigned int mask3)
{
      bit = countbits( mask1, mask2, mask3 );

      if ( selected * bit > Area )
            Area = selected * bit;

      if ( line == lines || (selected + lines - line) * bit <= Area )
            return;

      compute( line + 1, selected, mask1, mask2, mask3 );
      compute( line + 1, selected + 1, mask1 & v[line][0],
               mask2 & v[line][1], mask3 & v[line][2] );
}
```