

Descriere soluție - **problema palindrom**:

Se precalculează elementele matricei *ajunge*, după relația

$ajunge[i][j]$ = numărul minim de transformări ce pot fi efectuate pornind de la cifra *i* pentru a obține cifra *j*.

Având aceasta matrice calculată, primul subpunct este trivial, rămâne de văzut pentru fiecare pereche de cifre din șirul inițial (*i*, *N-i+1*), cu *i* de la 1 la *N/2* (unde *N* = lungimea șirului) care este cea mai mare cifră care se poate obține prin transformări repetate, atât din *A[i]*, cât și din *A[N-i+1]* (am notat cu *A* șirul inițial de lungime *N*).

Pentru al doilea subpunct vom folosi o recurență începând de la jumătatea șirului spre început, $D[i][j]$ = numărul minim de aplicări pentru care șirul cuprins între pozițiile *i* și *N-i+1* este palindrom și dă restul *j* la împărțirea prin *K*. Dacă pe pozițiile (*i*, *N-i+1*) punem cifra *q*, atunci restul obținut la pasul anterior *S* devine $(S + q * (10^{(N-i)} + 10^{(i-1)}) \% K$. Astfel se deduce relația de recurență:

```
if(ajunge[ A[i] ][p] >= 0 && ajunge[ A[N-i+1] ][p] >= 0) {
    //se poate ajunge la cifra p
    //vechiul rest este j
    int nou_rest = (j + p * (put_zece[N-i] + put_zece[i-1])) % K;
    D[i][nou_rest] = min(D[i][nou_rest], D[i+1][j] + ajunge[cif1][p]
                        + ajunge[cif2][p]);
}
```

unde *i* ia valori de la *N/2* la 1, *j* ia valori de la 1 la *K* și *p* ia valori de la 1 la 9. Astfel vom avea în $D[1][0]$ numărul minim de aplicări.

Pentru a reconstitui palindromul maxim divizibil cu *K*, repetăm procedeul plecând de la început spre mijloc (pentru a ne asigura că se obține cel mai mare palindrom), reținând poziția pe care ne aflăm și restul corespunzător secvenței de palindrom rămase.

Complexitatea algoritmului: $O(N * K)$.

Autori: Student Robert Hasna
 Student Vlad Ionescu
 Student Cosmin Tutunaru
 Student Vlad Duță
 Student Dragoș Oprică
 Student Alexandru Cazacu