Example: https://codeforces.com/contest/750/problem/E

With this trick you can use an automata/graph as a node of your segment tree, and merge two nodes using the floyd warshall algorithm, the main idea is:

To merge the segments [l,mid) and [mid,r) into a segment [l,r)
I can look at all paths:
-> I to **J** in [l,mid)
-> J to **K** in [mid,r)
and find a new path from **I** to **K** in [l,r), this is similar to the floyd warshall algorithm, so i can maintain a matrix with the minimum cost from **I** to **J**

```
struct nd{
 int mat[5][5];
 nd(int c){
   FOR(i,5){
     FOR(j,5){
       mat[i][j]=INT_MAX;
     }
     mat[i][i]=0;
   }
...
```

merge two matrices to get a new matrix

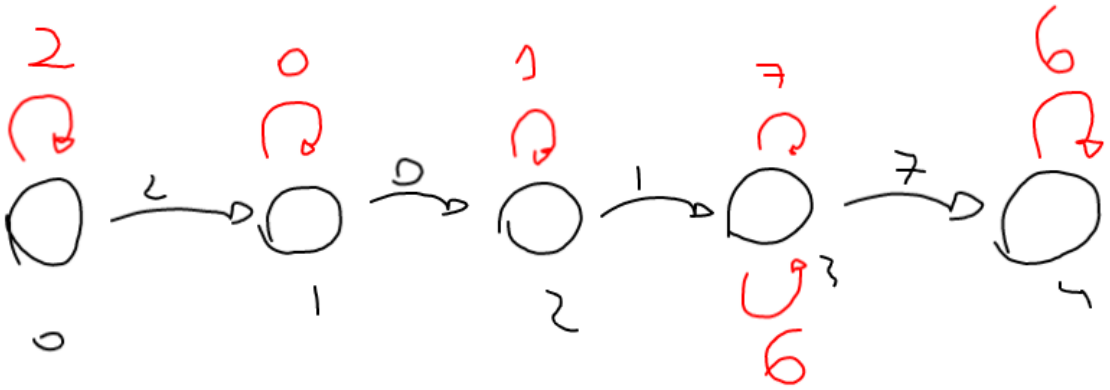
```
nd merge(nd a,nd b){
     nd NOVO;
     for(int i=0;i<5;i++){
       for(int j=i;j<5;j++){
         for(int k=i;k<=j;k++){
           NOVO.mat[i][j]=min(NOVO.mat[i][j],a.mat[i][k]+b.mat[k][j]);
         }
       }
     }
     return NOVO;
   }
```

And it makes a lot of sense, because all paths [l,r) can be decomposed into a path [l,mid) and a path [mid,r)

In this problem, we are given a string (with digits from ‘0’ to ‘9’) and queries

Each query is a range from L to R, in which we have to find the minimum number of deletions such that “2017” occurs as a subsequence in the substring[l,r] and “2016” does not occur.

We can model the cost with a graph (Nondeterministic Finite Automata)



The edges with red color means that it has a cost of 1, and black color have a cost of 0 (we also have implicit edges from **X** to **X** with cost 0 for all other digits),

1. The red edge means that we are deleting this digit with a cost of 1,
2. The black edge means we are getting this digit and trying to form a string with it.
3. The implicit edges means we are doing nothing

A path in this automata means a string we are getting,
in the first case if we get a ‘2’ we can either try to form something or delete it
if we already formed a “2017” and get a ‘6’ we have to delete it
etc.

Now we have to find the minimum cost from 0 to 4.