# АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ТЪРСЕНЕ

Ще разглеждаме зададеното число като низ и ще го означим с *s*.

Първата подзадача може да се реши като последователно търсим дали подниза, съставен от първите *k* символа на *s*, се среща в останалата част на низа. Задачата решаваме за *k* = 1, 2, … до момента, в който такъв подниз не може да бъде открит. При решаването могат да се използват стандартните функции за търсене на подниз. В най-лошия случай такова решение е със сложност $O(n^3)$.

Решение със същата сложност може да се намери като търсенето извършваме за $k = \left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor - 1, \left\lfloor \frac{n}{2} \right\rfloor - 2$, ... до откриване на първата стойност на *k*, за която има подниз с исканото свойство.

Описаните решения могат да се ускорят, като за търсената максимална дължина на маркираната част се приложи двоично търсене. Така би се получило решение със сложност $O(n^2 \log n)$, което решава втората подзадача.

Втората подзадача може да се реши и чрез използване на следната функция: За низа *s* с *bl*[*i*] означаваме най-дългия подниз на *s*, който започва от позиция *i* и съвпада с префикс на *s* (приемаме, че *bl*[0] = 0). Нека номерацията в *s* да започва от 0. Ако *cmp*($p_1$, $p_2$) е функция, която сравнява два подниза на *s*, започващи от позиции $p_1$ и $p_2$ и намира дължината на тяхната съвпадаща част в началото на двата подниза, стойностите на *bl*[*i*] могат да бъдат намерени по следния начин:

```
n=s.size();
bl[0]=0;
for(int i=1; i<n; i++)
   bl[i]=cmp(0,i);
```

Тогава търсената максимална дължина *ans* и позиция *pos* могат да се намерят така:

```
ans=0, pos=0;
for(int i=1; i<n; i++)
{
    z = min(bl[i],i);
    if(z>ans) {ans=z; pos=i;}
}
pos++;
```

Описаното решение е със сложност $O(n^2)$ и решава втората подзадача. За да може да решим и третата подзадача, ще ускорим пресмятането на стойностите на *bl*[*i*]. За всяка позиция *i* стойностите на *bl*[*i*], които не са равни на 0, определят дясна граница $i + bl[i] - 1$ на подниз, който започва от позиция *i* и съвпада с префикс на *s* (такъв подниз ще наричаме блок). *i*-тата позиция на низа може да се покрива от няколко блока. Най-голямата дължина на такъв блок ще означим с *r*[*i*], т.е. $r[i] = \max_{0<j\le i} \{j + bl[j] - 1\}$. Стойността на $r[i]$ може да се получи за повече от една стойност на *j*. Една от тези стойности означаваме с $l[i]$, т.е. $l[i]$ и $r[i]$ са ляв и десен край на един и същи блок. При определяне на стойностите на $bl[i]$ са нужни само последните пресметнати

стойности на $l[i]$ и $r[i]$, поради което е достатъчно да се използват две променливи *r* и *l*, а не масиви.

Стойностите на $bl[i]$ може да намерим със следния алгоритъм:

```
l=0, r=0;
bl[0]=0;
for(int i=1; i<n; i++)
{
   bl[i]=0;
   if(i>r)
   {
      bl[i]=cmp(0,i);
      if (bl[i]>0)
      {
         l=i;
         r=i+bl[i]-1;
      }
   }
   else
   {
      int k=i-l;
      if (bl[k]<r-i+1) bl[i]=bl[k];
      else
      {
          bl[i]=r-i+1;
          l=i;
          int q=cmp(r-i+1,r+1);
          if (q>0) {bl[i]+=q; r=i+bl[i]-1;}
       }
   }
```

Така сложността на пресмятането на $bl[i]$, а оттам и на целия алгоритъм става $O(n)$.

*Автор: Младен Манев*