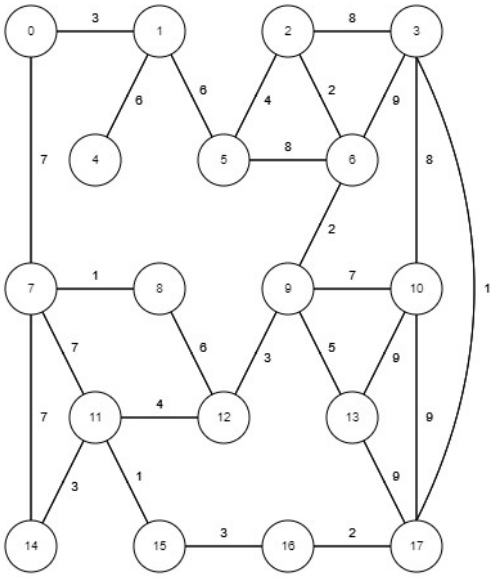
Конструкција и анализа алгоритама – фебруар 2018 (испит)

1. Израчунати брзу Фуријеову трансформацију полинома $P(x)=2+10x+23x^2+12x^3$

Решење: (47,-21-2w,3,-21+2w)

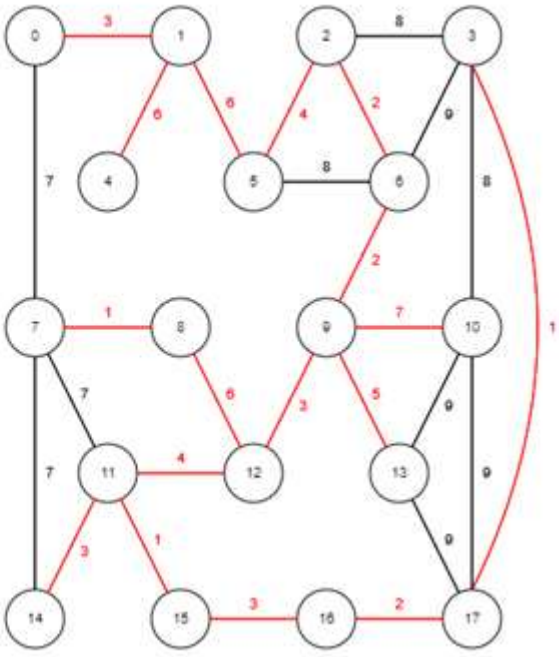
2. За граф дат матрицом тежина, конструисати обухватно стабло минималне цене (МЦСТ) и одредити цену тог стабла.



Решење

Прим (полазни чвор 0)

чвор	марк	цена	пут
0	T	0	-1
1	T	3	0
2	T	4	5
3	T	1	17
4	T	6	1
5	T	6	1
6	T	2	2
7	T	1	8
8	T	6	12
9	T	2	6
10	T	7	9
11	T	4	12
12	T	3	9
13	T	5	9
14	T	3	11
15	T	1	11
16	T	3	15



Цена: 3+6+6+4+2+2+7+5+3+6+4+1+3+1+3+2+1=59

3. Дат је природан број $N<10^9$ и N новчића положених на сто. Играмо игру окретања новчића тако да другог дана окренемо сваки други новчић, трећег дана окренемо сваки трећи новчић и тако редом све до N-тог дана када окренемо само последњи новчић.

а) Конструисати алгоритам временске сложености $O(\log N)$ који ће одредити колико је новчића остало у почетном положају након примене описаног алгоритма.

б) Образложити временску и просторну сложеност конструисаног алгоритма.

Решење:

1. Почнимо од чињенице да је први новчић све време остао у почетном положају.
2. За све остале, постављамо питање: да ли су били окренути паран или непаран број пута? Остају у почетном положају уколико су окренути паран број пута.

Решење онда гласи: изузимајући дељење са један, у почетном положају су новчићи на оном месту које има паран број делилаца.

На пример, од првих 10 бројева, то су само 4 (дељив са 2 и са 4) и 9 (са 3 и са 9).

Заједно са првим новчићем, укупно решење је 3 новчића. Ова три броја (1, 4 и 9) су све потпуни квадрати.

Од 90 новчића, 9 би остало у истом положају (најближи квадрат је 81).

4. **Заправо, у питању је математичко правило: укључујући дељивост са један, сматрамо да потпуни квадрати имају непаран број делилаца.** Решење овог задатка за било који број новчића је: $\left[|\sqrt{\boldsymbol{N}|}\right]$

Временска сложеност: алгоритам за тражење целог дела квадратног корена (бинарна претрага, вавилонски метод, Њутн,...) захтева ***O(log N)*** времена ако ***N*** има произвољан број битова. За вежбу остављамо проблем у ком је број ***N*** задат са ограничењем на број битова.

Просторна сложеност: O(1) јер не користимо додатни простор за чување броја делилаца квадрата мањих од ***N***

4. Ако је дат алгоритам за множење две ***n*x*n*** доње троугаоне матрице чије време извршавања је O(*T(n)*), доказати да постоји алгоритам за множење две произвољне ***n*x*n*** матрице чије време извршавања је O($T(n)+n^2$). (Може се претпоставити да је *T(cn)=O(T(n))* за сваку константу *c*)

Решење

Нека су A и B две произвољне квадратне матрице реда n .

Свака од њих се мозе представити као збир по једне горње и по доње троугаоне матрице (T_A, B_A, T_B, B_B, редом):

$A= T_A + B_A$

$B= T_B + B_B$

Даље је: $AB=(T_A + B_A) (T_B + B_B)=T_AT_B + B_AB_B + B_AT_B + T_AB_B$

Ако се употреби алгоритам из формулације задатака могуће је израчунати производ:

$$\begin{bmatrix} B_A & 0 \\ T_A & B_A \end{bmatrix} \times \begin{bmatrix} B_B & 0 \\ T_B & B_B \end{bmatrix} = \begin{bmatrix} B_AB_B & 0 \\ B_AT_B + T_AB_B & B_AB_B \end{bmatrix}$$

Као резултат добија се матрица која садржи блокове: B_AB_B , $B_AT_B + T_AB_B$

Ови блокови учествују у израчунавању производа A × B.

Даље, блок , $T_AT_B= [(T_AT_B)^T]^T=[T_B^T T_A^T]^T$ се добија применом црне кутије на транспоноване матрице T_A,T_B

На тај начин се проблем израчунавања производа две произвољне матрице своди на проблем израцунавања производа квадратне доње троугаоне матрице квадратном доње троугаоном матрицом.

Укупно време извршавања : $O(T(2n)+3*n^2)=O(T(n)+n^2)$

5. Међусобно различити елементи неког скупа ***A*** су смештени у бинарно стабло претраге висине ***h***. Конструисати алгоритам сложености ***O(h)*** који за дати елемент ***x*** скупа ***A*** одређује претходни по величини елемент скупа ***A***.

Решење

Појам претходника и алгоритам за налажење претходника је симетричан у односу на алгоритам следбеника чије решење је детаљни изложено на вежбама, односно дати као решење колоквијума.
Претпоставимо да је бинарно стабло претраге грађено тако да је кључ сваког десног сина већи од кључа оца.
Постоје 2 карактеристична случаја за чвор v чији кључ име вредност елемњнта ***x*** из скупа ***A***:

1. чвор v има лево подстабло
=> Следбеник се тражи као највећи у левом подстаблу (кључ најдешњег чвора у левом подстаблу)

2. чвор v нема левог сина
=> Следбеник се тражи тако што се тражи кључ најнижег претка ***w*** тако да леви син од ***w*** је предак чвора ***v***.

Ова операција се најефикасније изводи ако се за сваки чвор чува и поље отац са показивачем на оца чвора. Дакле, довољно је кренути навише од чвора ***v*** према корену све док се не нађе чвор који је леви син свог оца. Ако се деси да не постоји такав предак, онда је ***x*** најмањи елемент скупа и нема свог претходника.

У обе ситуације, следбеник се налази без упоређивања кључева захваљујући принципу распореда стабала.

Сложеност: У оба случаја, сложеност алгоритма је O(h).
Операције налажења максималног кључа у BST стаблу, као и максималног претходника навише имају време извршавања сразмерно висини стабла *h*, тј. *O(h)*, јер у свакој од ситуација прате путању кроз стабло у само једном смеру (навише или надоле).

```
Prethodnik(x)
 begin
      if x.levi != NULL then
          return Maksimum(x.levi)
      y = x.otac;
      while y != NULL and x = y.levi do
            x = y;
           y = y.p;
      return y
end
```