CEZAR – Soluţie

O implementare posibilă utilizează metoda GREEDY, O(n*(n-k)) sau O((n-k)*log(n))

Se elimină succesiv, dintre frunzele existente la un moment dat, frunza de cost minim. Toate nodurile au costul iniţial 1. La eliminarea unei frunze, se incrementează cu 1 costul tatălui acesteia. Validitatea metodei rezultă din observaţia că, la eliminarea unei frunze oarecare, tatăl acesteia poate deveni frunză la rândul lui, dar cu un cost strict mai mare decât al frunzei eliminate.

Se poate reţine arborele cu ajutorul listelor de adiacenţă (liniare sau organizate ca arbori de căutare), iar frunzele se pot memora într-un minheap de costuri, structură care se actualizează în timp logaritmic.

```
const fi='cezar.in';fo='cezar.out';
type pnod=^nod;
     nod=record nod:integer; next:pnod end;
     vect=array[1..10000]of integer;
var f:text;
    p:array[1..10000]of pnod;
    h:array[1..10001]of integer;
    c:^vect;
    n,k,lh:integer;s:longint;

procedure remove(var p:pnod; a:integer);
var q,r:pnod;
begin
  q:=p;
  if p^.nod=a then begin
      p:=p^.next;
      dispose(q)
  end
  else begin
    while q^.next^.nod<>a do q:=q^.next;
    r:=q^.next;
    q^.next:=q^.next^.next;
    dispose(r)
  end
end;

procedure citire;
var i,x,y:integer;q:pnod;
begin
   assign(f,fi);reset(f);
   readln(f,n,k);
   for i:=1 to n-1 do begin
      readln(f,x,y);
        new(q);q^.nod:=x;q^.next:=p[y];p[y]:=q;
        new(q);q^.nod:=y;q^.next:=p[x];p[x]:=q;
   end;
   new(c);
   for i:=1 to n do begin c^[i]:=1;h[i]:=n+1 end;
   close(f);
   for i:=1 to n do
       if p[i]^.next=nil then begin inc(lh);h[lh]:=i end;
   c^[n+1]:=maxint;h[n+1]:=n+1
end;
```

```
procedure desfrunzire;
var ii,i,j,pmin,aux:integer;min:longint;
begin
  s:=0;
  for ii:=n-1 downto k+1 do begin
    pmin:=h[1];s:=s+c^[pmin];
    j:=p[pmin]^.nod;
    c^[j]:=c^[j]+c^[pmin];
    remove(p[j],pmin);
    if p[j]^.next=nil then h[1]:=j
    else begin h[1]:=h[lh];h[lh]:=n+1;dec(lh) end;
    i:=1;
    while 2*i<=lh do begin
      if c^[h[2*i]]<c^[h[2*i+1]] then j:=2*i else j:=2*i+1;
        if c^[h[i]]>c^[h[j]] then begin
         aux:=h[i];h[i]:=h[j];h[j]:=aux;i:=j
      end
      else i:=n
    end
  end
end;

begin
  citire;
  desfrunzire;
  assign(f,fo);rewrite(f);
  writeln(f,s);
  close(f)
end.



#include <fstream.h>
struct NOD{int nod;NOD* next;};
NOD *dp[10000];
long s; int *c,n,k,h[10001],lh;

void add(NOD*& p, int a)
{NOD* q;
 q=new(NOD);q->next=p;
 q->nod=a;p=q;
}

void remove(NOD*& p, int a)
{NOD *q,*r;
 if (p->nod==a){
     q=p;p=p->next;
     delete q;
 }
 else {
    q=p;
    while (q->next->nod!=a) q=q->next;
    r=q->next;q->next=q->next->next;
    delete r;
 }
}

void citire()
{int i,x,y;
 ifstream f("cezar.in");
 f>>n>>k;
 c=new int[10000];
```

```
 for (i=0;i<n;i++){
   *(c+i)=1;dp[i]=0;h[i]=n;
 }
 for (i=0;i<n-1;i++){
   f>>x>>y;
   add(dp[x-1],y-1);add(dp[y-1],x-1);
 }
 for (i=0;i<n;i++)
   if (!dp[i]->next)h[++lh]=i;
 *(c+n)=20000;h[n]=n;
 f.close();
}

void desfrunzire()
{ int ii,i,j,pmin;long min;
  s=0;
  for (ii=n-1;ii>=k+1;ii--){
    pmin=h[1];s+=*(c+pmin);
    j=dp[pmin]->nod;
    *(c+j)+=*(c+pmin);
    remove(dp[j],pmin);
    if(!dp[j]->next)h[1]=j;
    else {h[1]=h[lh];h[lh]=n;lh--;}
    i=1;
    while (2*i<=lh){
      if (*(c+h[2*i])<*(c+h[2*i+1])) j=2*i; else j=2*i+1;
      if (*(c+h[i])>*(c+h[j]))
      {int aux=h[i];h[i]=h[j];h[j]=aux;i=j;}
      else i=n;
    }
  }
}
void main()
{ citire();
  desfrunzire();
  ofstream f("cezar.out");
  f<<s<<'\n';
  f.close();
}
```