{ altura(a) +k -1 <= N; k>=1}

```
func kobtenir_suma_nivells ( a: arbre(int), k:nat) ret v: vector[1..N] d'int;
 var   v1,v2: vector[1..N] d'int fvar;
 Si es_buit(a) llavors v:=zeros(k);
 sino
   v1:=  kobtenir_suma_nivells ( fe(a), k+1 );
   v2:=  kobtenir_suma_nivells ( fd(a), k+1 );
   v:= suma(v1,v2,k+1);
   v[k]=arrel(a);
 fSi
```

{per tota posició $m \in \{k..N\}$ tenim v[m] =“suma dels element d'a a nivell m-k+1”}

{ altura(a) <= N }

```
func obtenir_suma_nivells ( a: arbre(int) ) ret v: vector[1..N] d'int;
  v:=kobtenir_suma_nivells ( a,1);
```

{per tota posiciót $m \in \{1..N\}$ tenim v[m] =“suma dels element d'a a nivell m”}

{ altura(a) +k -1 <= N; k>=1}

acció kobtenir_suma_nivells (ent a: arbre(int), ent k:nat, e/s v: vector[1..N] d'int)
Si es_buit(a) **llavors** ;
sino
kobtenir_suma_nivells (fe(a), k+1, v);
kobtenir_suma_nivells (fd(a), k+1, v);
v[k]+=arrel(a);
fSi

{per tota posició a $m \in \{k..N\}$ afegim a v[m] la suma dels element d'a a nivell m-k+1}

{ altura(a) <= N }

func obtenir_suma_nivells (a: arbre(int)) **ret** v: vector[1..N] d'int;
v := zeros(1);
kobtenir_suma_nivells (a,1,v);

{per tota posiciót $m \in \{1..N\}$ tenim v[m] =“suma dels element d'a a nivell m”}