1. evitarea apelului recursiv - Prolog (RS)

```
r([],0).
r([H|T],S) :- r(T,S1), aux(S1,H,S).

aux(S1,H,S) :-
        S1>=2, !,
        S = S1+H.

aux(S1,_,S) :-
        S = S1+1.
```

3. se generează lista submulțimilor de sumă S

Ideea algoritmului:

- un predicat ce construiește submulțimile valide (nedet.)
- un predicat ce returnează o listă cu toate submulțimile valide

Modelele matematice:

sublm $(e_1 \dots e_n, s) =$

1. (e_1), $s = e_1$
2. sublm $(e_2 \dots e_n, s)$
3. $e_1 \oplus$ sublm $(e_2 \dots e_n, s - e_1)$, dacă $n > 1$ și $s - e_1 > 0$

main $(e, s) =$ $\bigcup$ sublm (e, s)

```
% sublm(L - lista de elemente, S - nr. intreg, R - lista)
% R -> o submultime valida (a carei elem. au suma S)
% model de flux: (i,i,o) -> nedeterminist
```

```
subm([H|-], H, [H]). % se incheie apelul recursiv si se
% incepe construirea submultimii valide

subm([-|T], S, R):-
    % se trece mai departe, neadaugand elem. curent in submultime
    subm(T, S, R).

subm([H|T], S, R):-
    H =< S,
    % se adauga elem. curent la submultime
    S1 is S - H,
    subm(T, S1, R1),
    R = [H|R1].

% main(L - lista, S - Nr, R - lista de liste)
% reprezinta lista de rezultate, stocata in R
% model de flux: (i, i, o) -> determinist
main(L, S, R):-
    findall(R1, subm(L, S, R1), R).
```

5. Nr. de sublisteste pt. care cel mai mare element numeric de la coada nivel este nr. par.

Ideea algoritmului:

1. o functie ce determina maximul (dintre atomii numerici) ai unei liste neliniare
2. o functie ce nr. sublistelor valide

Modelele matematice:

$$maxim(l_1 \dots l_n, m) = \begin{cases} m, & n = 0 \\ maxim(l_2 \dots l_n, l_1), & l_1 \text{ e nr. si } m = \phi \\ maxim(l_2 \dots l_n, l_1), & l_1 \text{ e nr., } m \neq \phi \text{ si } l_1 > m \\ maxim(l_2 \dots l_n, m), & l_1 \text{ e atom nenumeric} \\ & \text{SAU } l_1 \text{ e nr., } m \neq 0 \text{ si } l_1 \leq m \\ maxim(l_2 \dots l_n, maxim(l_1, m)), & \text{altfel} \end{cases}$$

$$suc(l) = \begin{cases} 0, & l \text{ e atom} \\ 1 + \sum_{i=1}^{n} suc(l_i), & l \text{ e lista si } maxim(l, \phi) \neq \phi \\ & \text{si } maxim(l, \phi) \text{ e par} \\ \sum_{i=1}^{n} suc(l_i), & \text{altfel} \end{cases}$$

```
; maxim (l - lista, m - nr. sau φ)
; returnează maximul dintre atomii numerici ai unei liste
(defun maxim (l m)
    (cond
        ((null l) m)
        ((and (numberp (car l)) (null m)) (maxim (cdr l)
                                                 (car l)))
        ((and (numberp (car l)) (not (null m)))
         (> (car l) m)) (maxim (cdr l) (car l)))
        ((and (numberp (car l)) (not (null m)))
         (maxim (cdr l) m))
        ((atom (car l)) (maxim (cdr l) m))
        (t (maxim (cdr l) (maxim (car l) m)))
    )
)
```

```
; sublista (e - lista)
; returnează nr. de sublista pt. care cel mai mare atom
; numeric este par

(defun sublista (e)
    (cond
        ((atom e) 0)
        ((and (listp e) (not (numberp (maxim e nil))))
            (evenp (maxim e nil)))
            (+ 1 (apply #'+ (mapcar #'sublista e))))
        (+ (apply #'+ (mapcar #'sublista e)))
    )
)
```