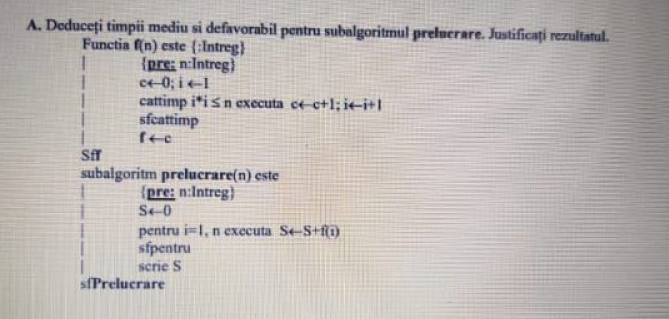
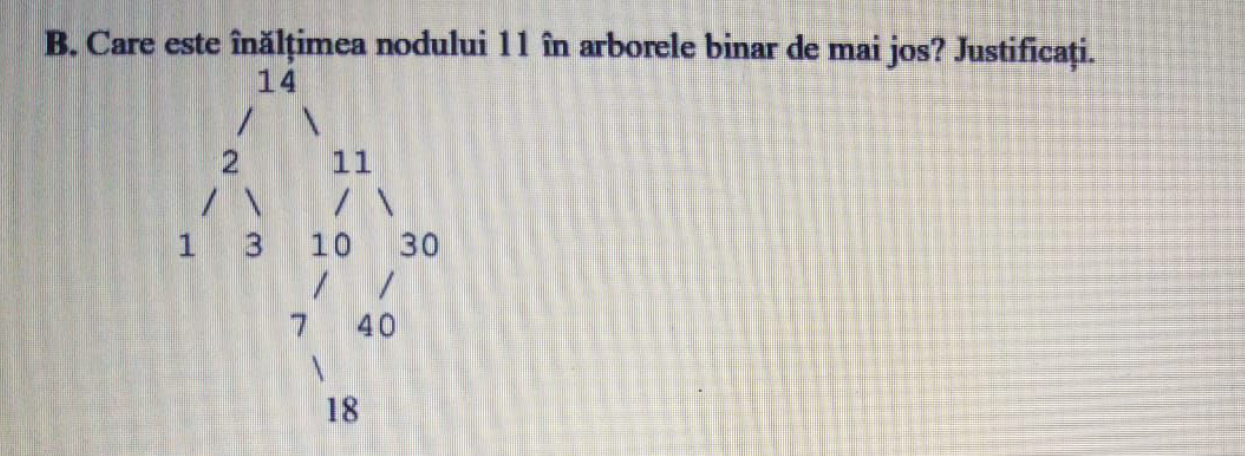
Model 4

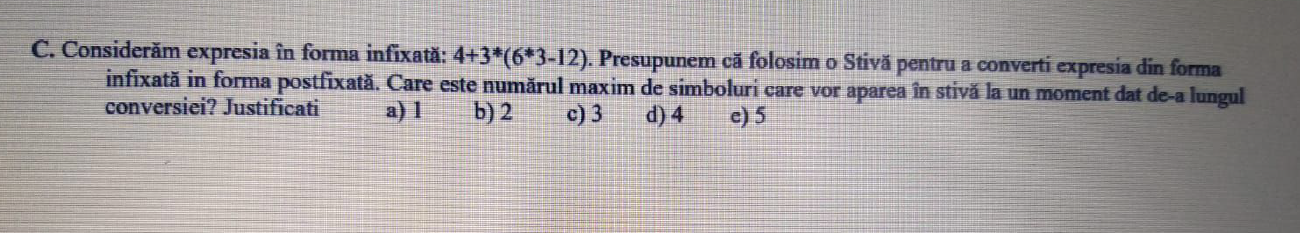


Functia f : se realizeaza in F(n) = sqrt(n) pasi => Theta(sqrt(n))

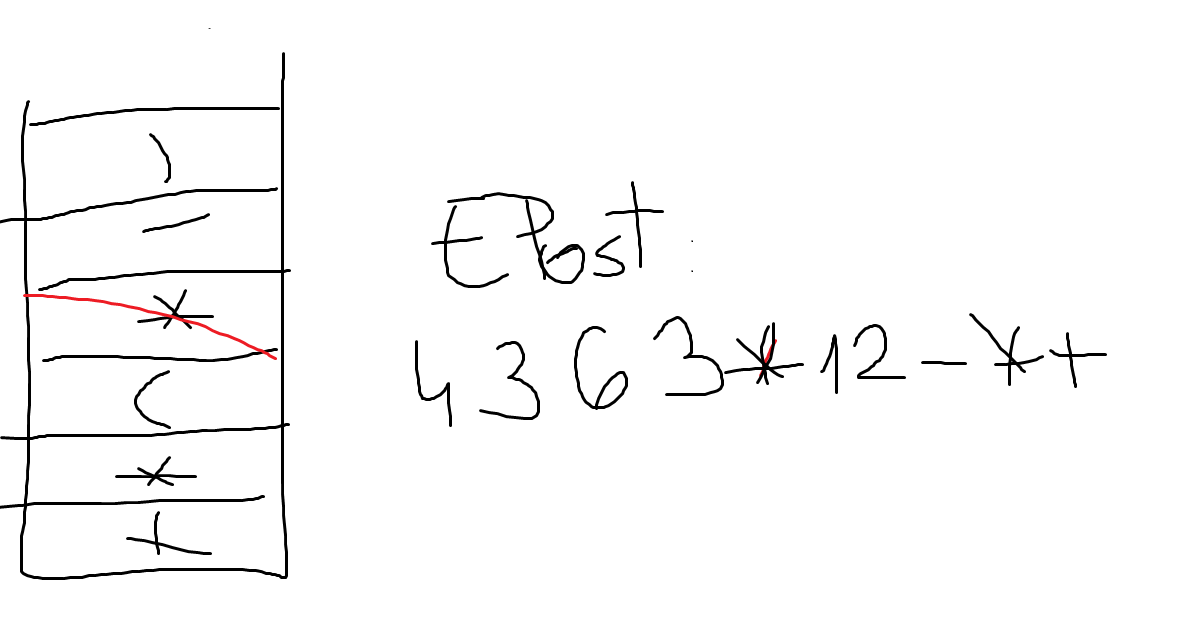
T(n) = Σ F(i) = Σ sqrt(i) = sqrt(1) + sqrt(2) + … + sqrt(n) ∈ O(n\*sqrt(n))



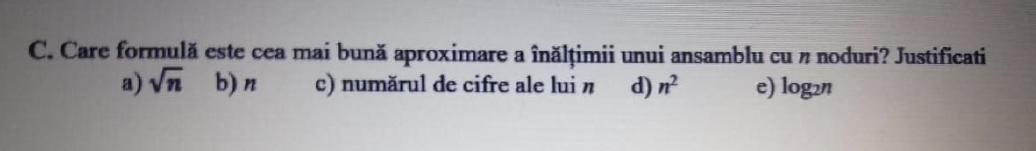
Inaltimea unui nod este lungimea celui mai lung lant de la nodul respectiv la o frunza. (Inaltimea subarborelui care are nodul respectiv ca radacina). Deci, inaltimea nodului 11 este 3. (Avem lantul 11 -> 10 -> 7 -> 18)

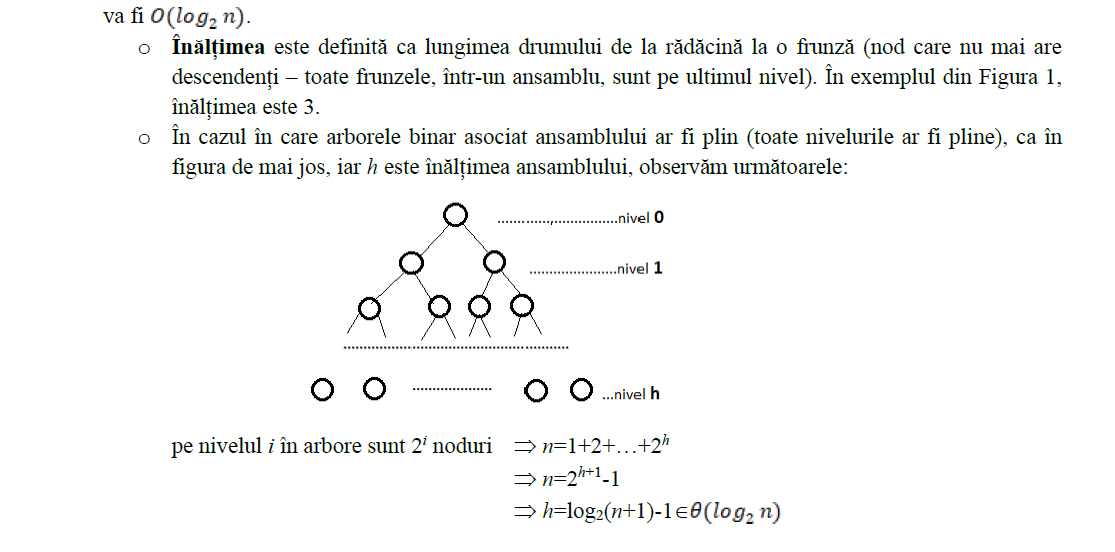


Vom folosi o stiva in care adaugam operatorii si o coada in care vom avea forma finala EPost.

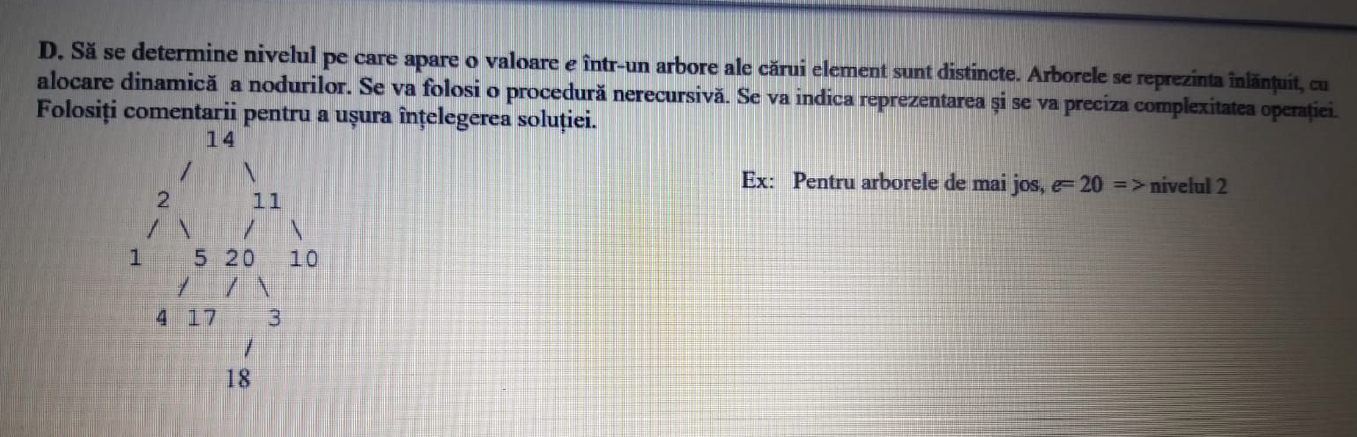
4 + 3 \* (6 \* 3 -12) – Forma infixata

d) 4 – vor exista 4 simboluri (dupa ce se adauga – si pana cand se ajunge la paranteza inchisa)





e) log2n –



Subalgoritm nivel(ab, e)

{Complexitate: O(n)}

Pre: ab este un arbore binar

Post: se afiseaza nivelul pe care se afla elementul e

{coada va continue adrese de noduri}

Creeaza(C)

Daca ab.rad != NIL atunci

{arborele nu e vid}

Adauga(C, ab.rad)

{se adauga radacina in coada}

Niv <- 0

SfDaca

{vom retine primul element de pe urmatorul nivel, pentru a stii cand sa incrementam variabila nivel}

primulDePeNivel <- -1

CatTimp !vida(C) executa

Sterge(C, p)

{se sterge nodul din coada}

Daca primulDePeNivel = [p].e atunci

{daca am ajuns la primul elem de pe urm nivel, atunci crestem nivelul si resetam variabila primulDePeNivel la valoarea -1}

Niv <- niv + 1

primulDePeNivel <- -1

SfDaca

Daca primulDePeNivel = -1 atunci

{daca nu am gasit inca primul element de pe urmatorul nivel, incercam sa il actualizam, verificand daca nodul actual are descendenti}

Daca [p].st != NIL atunci

primulDePeNivel <- [[p].st].e

altfel Daca [p].dr != NIL atunci

primulDePeNivel <- [[p].dr].e

SfDaca

SfDaca

Daca [p].e = e atunci

{daca nodul actual este egal cu cel cautat, returnam nivelul pe care se afla}

Nivel <- niv

SfDaca

Daca [p].st != NIL atunci

{exista legatura stanga}

Adauga(C, [p].st)

{se adauga descendentul stang}

SfDaca

Daca [p].dr != NIL atunci

{exista legatura dreapta}

Adauga(C, [p].dr)

{se adauga descendentul drept}

SfDaca

SfCatTimp

SfSubalgoritm