Arbori AVL

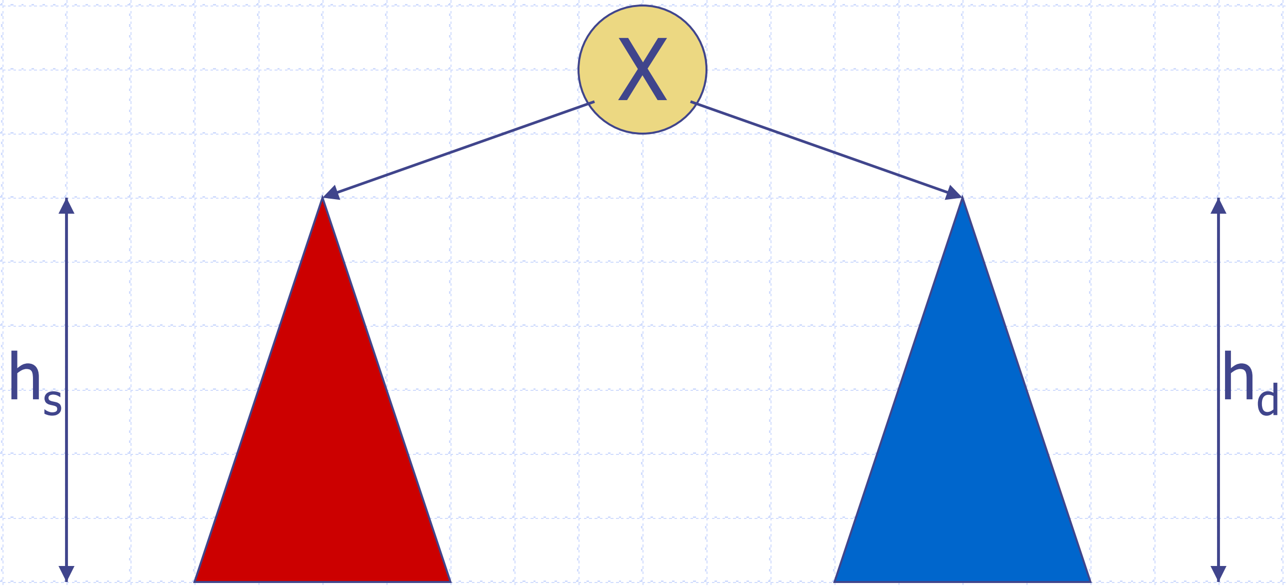
- Arborii AVL sunt arbori binari de cautare, care au in plus o proprietate de echilibru stabilita de **A**delson-**V**elskii si **L**andis, de unde si denumirea de arbori **AVL**.

- Proprietatea de echilibru e valabila ***pentru orice nod*** al arborelui si spune ca: “inaltimea subarborelui stang al nodului difera de inaltimea subarborelui drept al nodului prin cel mult o unitate”

- Cu alte cuvinte, pentru orice nod, cei doi subarbori au inaltimile sau egale, sau, daca nu, ele difera prin maxim o unitate in favoarea unuia sau altuia dintre subarbori

Arbori AVL

- Formal, acest lucru se traduce in felul urmator:



- $| h_s - h_d | \leq 1$, oricare ar fi nodul X apartinand arborelui

Arbori AVL

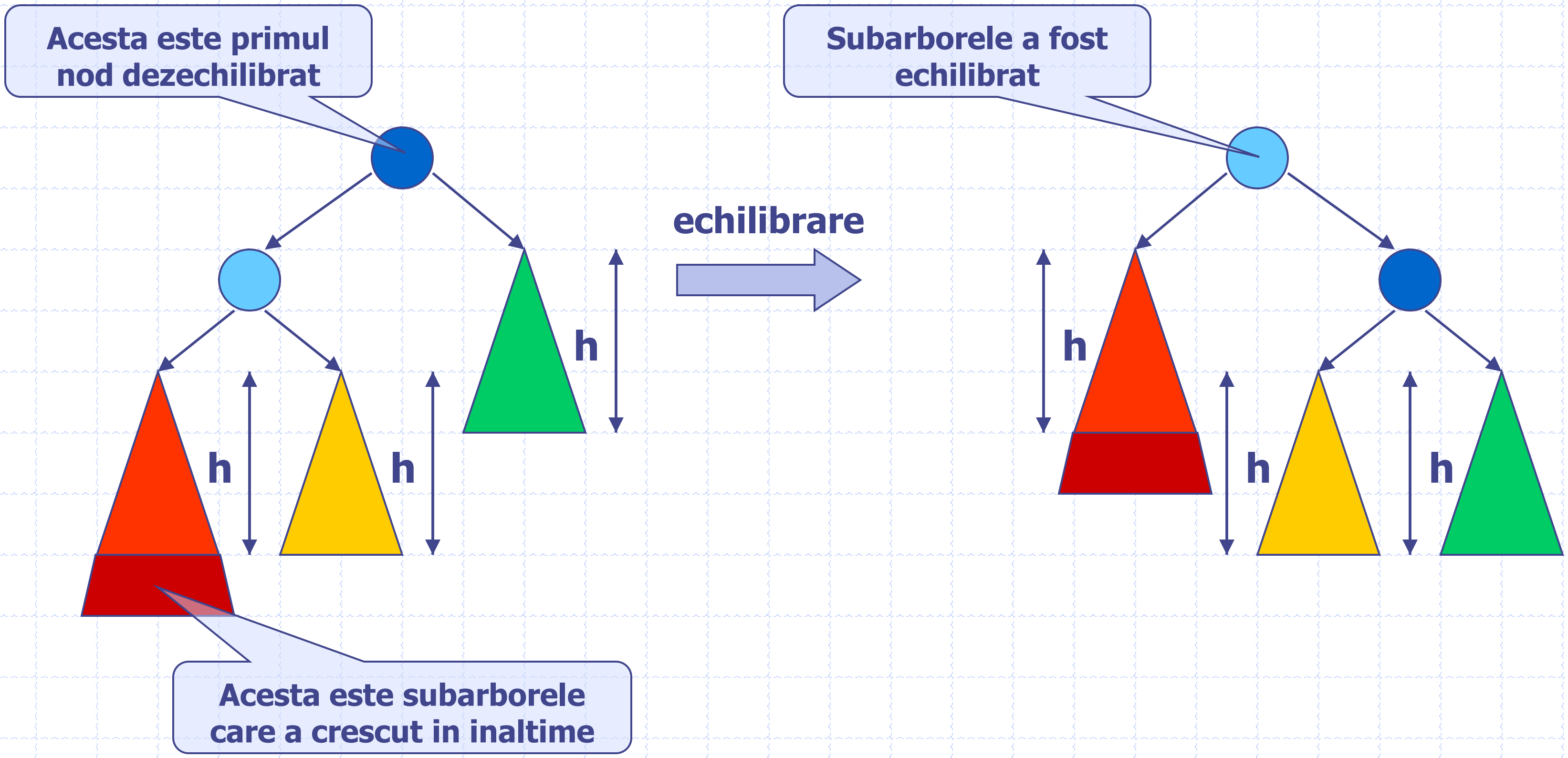
- Practic, arborii AVL se comporta la fel ca arborii binari de cautare simpli, mai putin in cazul operatiilor de insertie si suprimare de chei
- O insertie intr-un arbore binar ordonat poate duce la dezechilibrarea anumitor noduri, dezechilibrare manifestata prin nerespectarea formulei de pe slide-ul anterior pentru respectivele noduri
- Daca in cazul arborilor binari de cautare simpli aceste situatii nu deranjeaza (acestia nefiind suficient de competenti pentru a le rezolva), in cazul arborilor AVL ele trebuie rezolvate pentru a asigura faptul ca si dupa o insertie sau o suprimare, arborele ramane un arbore AVL

Arbori AVL

- Vom discuta in continuare numai despre cazurile care apar la insertia cheilor, urmand sa tratam la final suprimarea cheilor folosindu-ne de aceleasi cazuri
- Principial, o cheie se insereaza intr-o prima faza, ca si intr-un arbore binar ordonat obisnuit, adica se porneste de la radacina si se urmeaza fiul stang sau fiul drept, in functie de relatia dintre cheia de inserat si cheia nodurilor prin care se trece, pana se ajunge la un fiu nul, unde se realizeaza insertia propriu-zisa
- In acest moment se parcurge drumul invers (care este unic) si se cauta pe acest drum ***primul nod care nu este echilibrat***, adica primul nod ai carui subarbori difera ca inaltime prin 2 unitati
- Acest nod trebuie echilibrat si el se va afla intotdeauna intr-unul din cele 4 cazuri prezentate in continuare

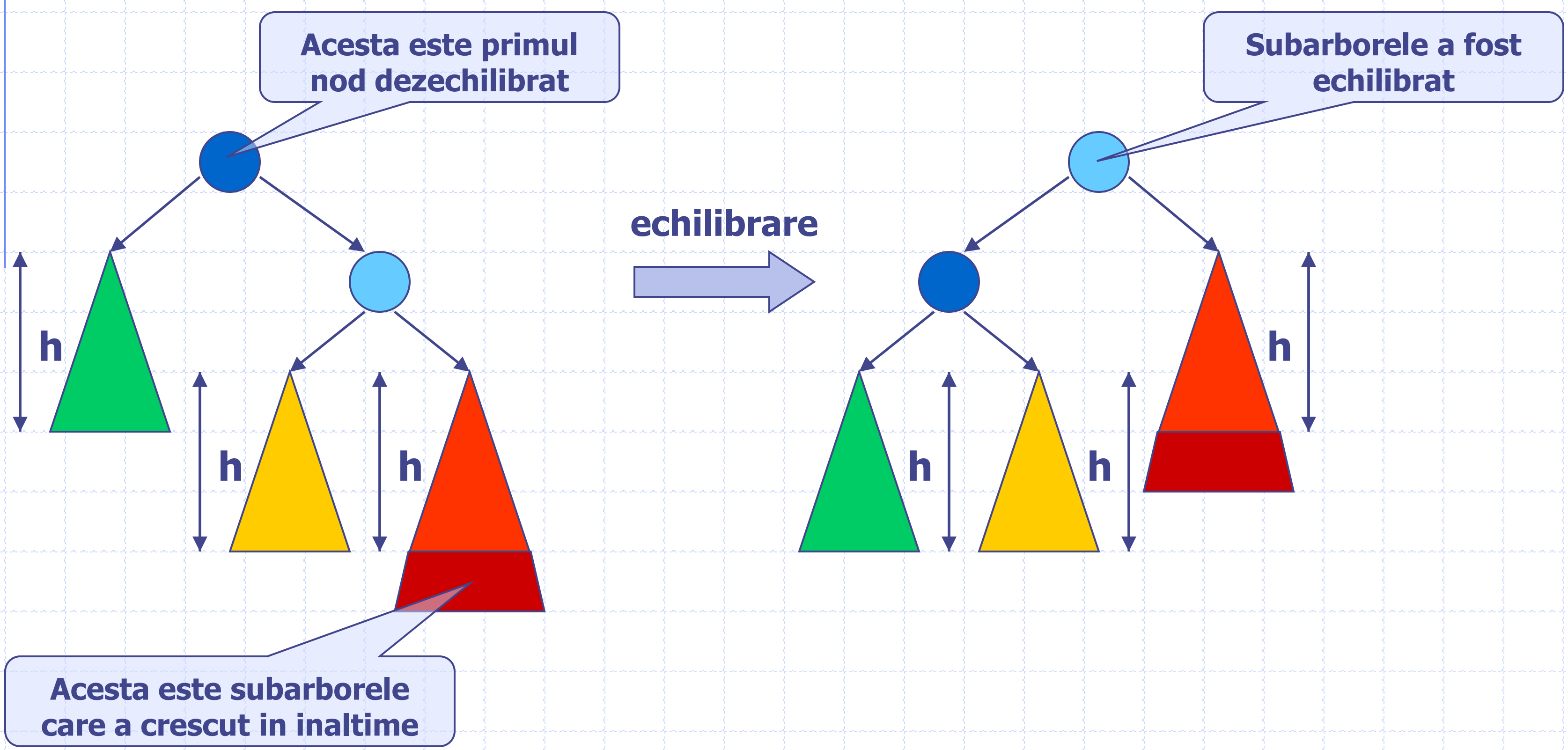
Arbori AVL

- Cazul 1-stanga



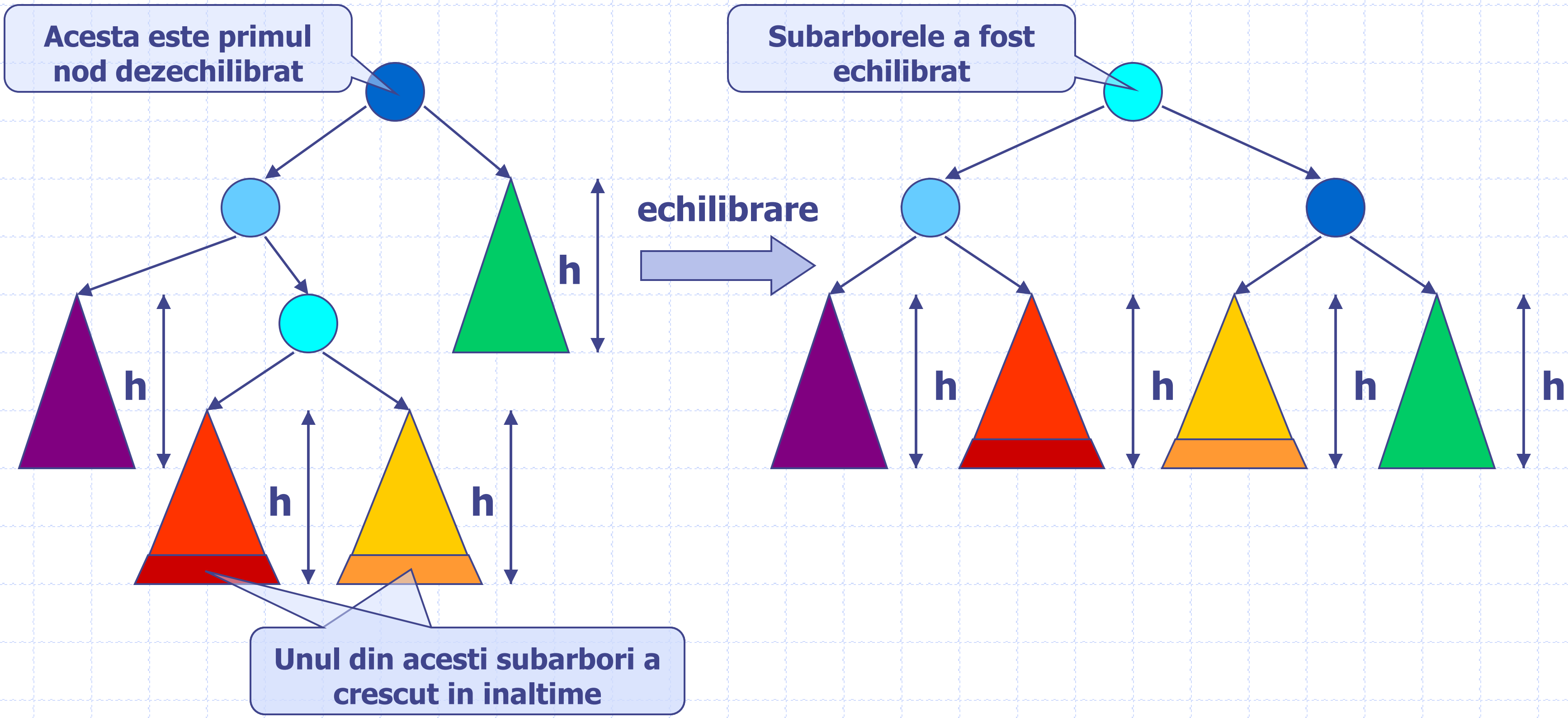
Arbori AVL

- Cazul 1-dreapta (simetric in oglinda fata de cazul 1-stanga)



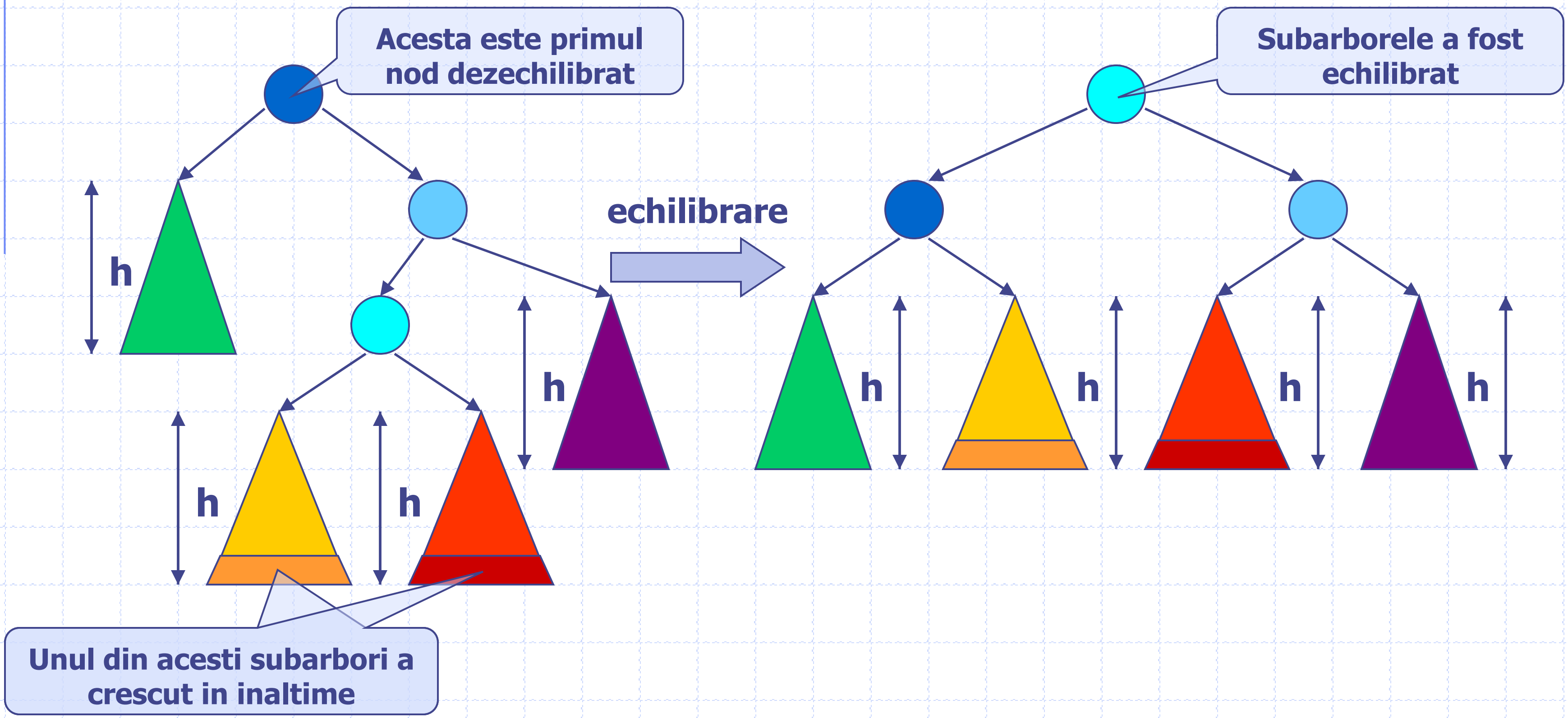
Arbori AVL

- Cazul 2-stanga



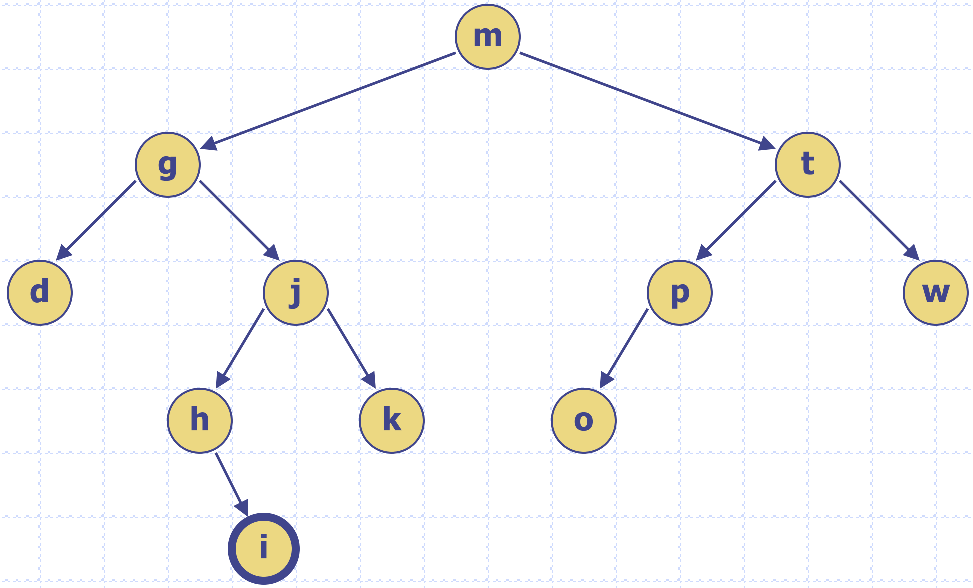
Arbori AVL

- Cazul 2-dreapta (simetric in oglinda fata de cazul 2-stanga)



Arbori AVL

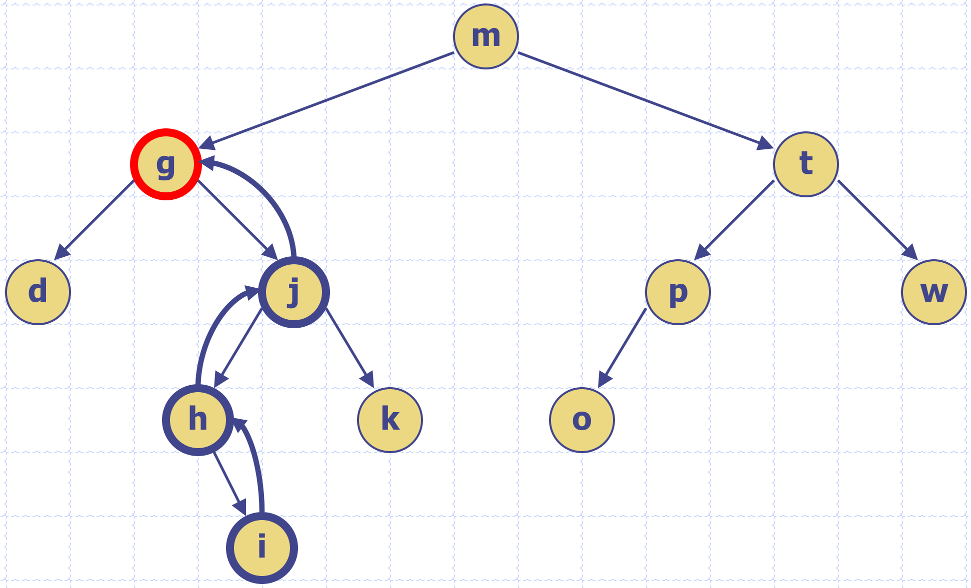
- Pentru exemplificare, vom considera o situatie in care o insertie intr-un arbore AVL necesita reechilibrarea arborelui



- Insertia cheii ‘i’ se face ca intr-un arbore binar obisnuit, in dreapta cheii ‘h’

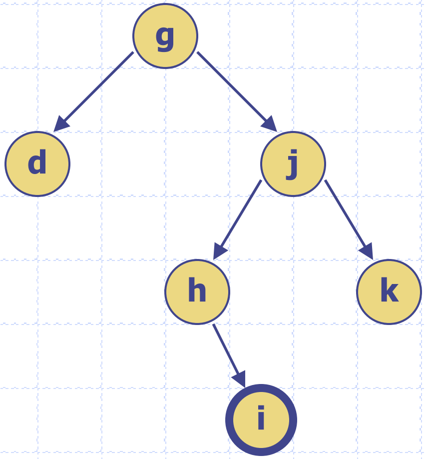
Arbori AVL

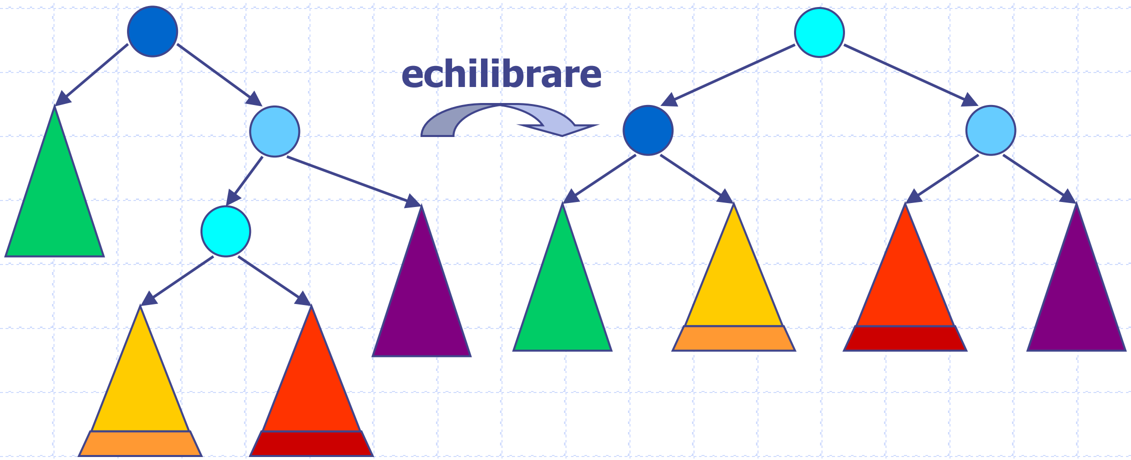
- Dupa insertie, se pleaca de la nodul inserat si se merge inapoi catre radacina, verificand ca toate nodurile intalnite sa respecte formula de echilibru AVL
- Primul nod dezechilibrat pe calea de cautare este nodul 'g'



Arbori AVL

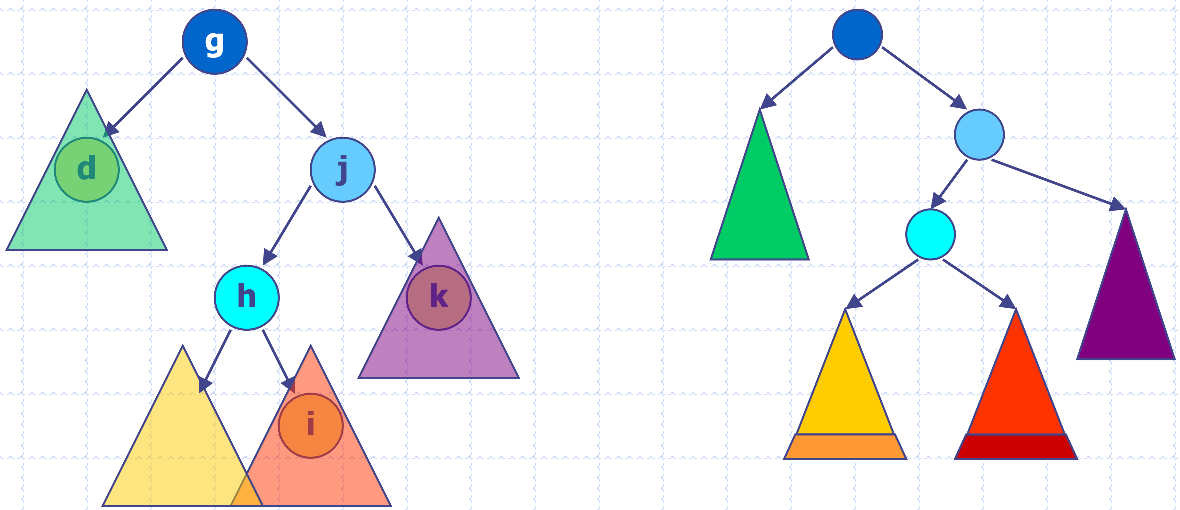
- Trebuie sa echilibram subarborele care are radacina 'g'
- Daca echilibram acest subarbore, intreg arborele va deveni un arbore AVL
- Pentru aceasta, consideram doar subarborele cu radacina 'g' si stabilim in care dintre cele 4 cazuri de echilibrare ne situam
- Se observa usor ca este vorba despre cazul 2-dreapta, deoarece cheia 'i' (cea care a dus la cresterea arborelui) se afla in subarborele **rosu** in raport cu cheia 'g' (daca s-ar fi aflat in subarborele **mov**, ne situam in cazul 1-dreapta)





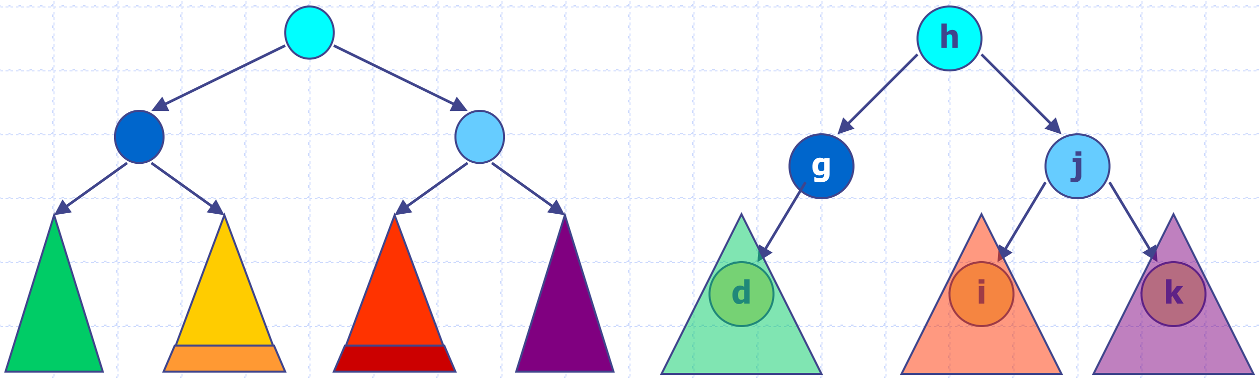
Arbori AVL

- In continuare, identificam pe arborele nostru concret, componentele modelului, folosind culori corespunzatoare



- Pentru echilibrare, nu trebuie decat sa echilibram modelul corespunzator cazului 2-dreapta si apoi sa reconstruim arborele concret, pornind de la model si de la componentele identificate

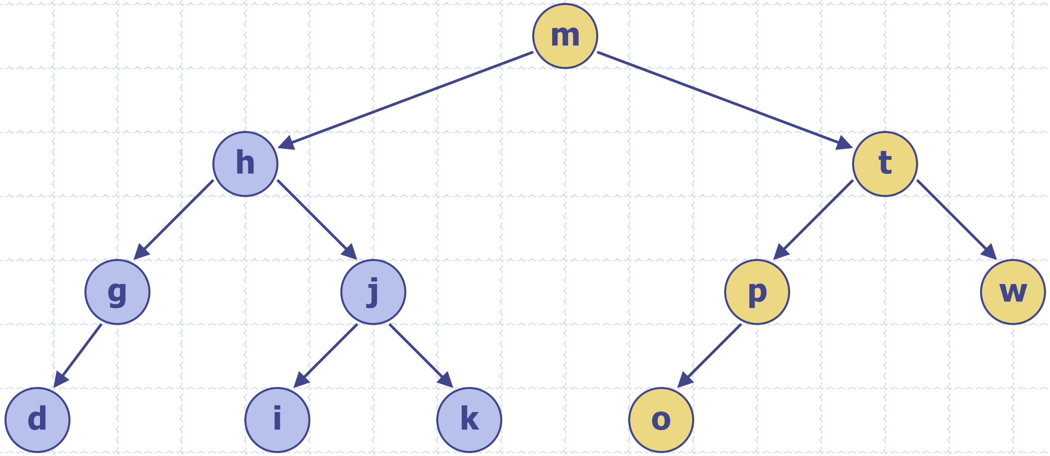
Arbori AVL



- Reechilibrarea se face relativ simplu:
  - Radacina arborelui echilibrat trebuie sa fie nodul albastru deschis, si pe slide-ul anterior am identificat acest nod ca fiind nodul 'h'
  - Idem pentru celelalte doua noduri cu nuante de albastru
  - Subarborele verde trebuie legat la stanga nodului albastru inchis, si pe slide-ul anterior am identificat subarborele verde ca fiind cel ce contine doar cheia 'd'
  - Idem pentru ceilalti subarbori colorati, cu mentiunea ca subarborele galben este subarborele vid, deci nu a mai fost reprezentat

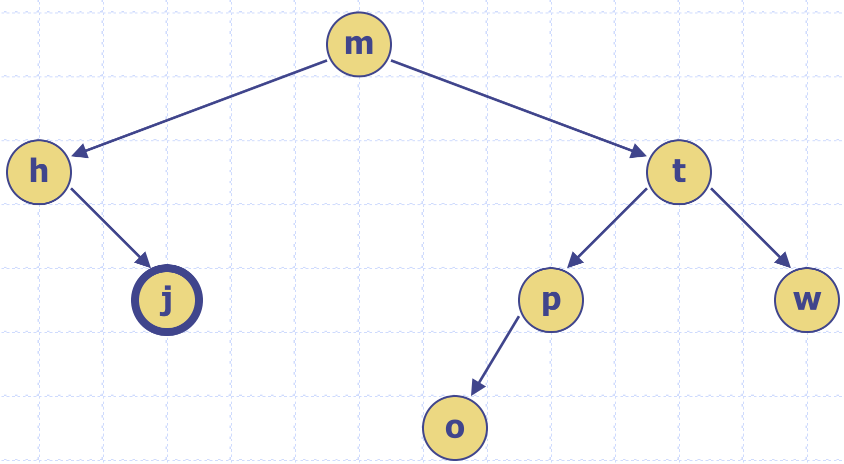
Arbori AVL

- Subarborele echilibrat este desenat mai jos, cu noduri albastre
- Legand acest subarbore de arborele initial, acesta devine echilibrat AVL la randul sau
- Practic, am inserat in arborele initial cheia ‘i’ care a provocat dezechilibrarea arborelui, dupa care am aplicat cazul de reechilibrare 2-dreapta asupra subarborelui corespunzator si am reobtinut un arbore AVL



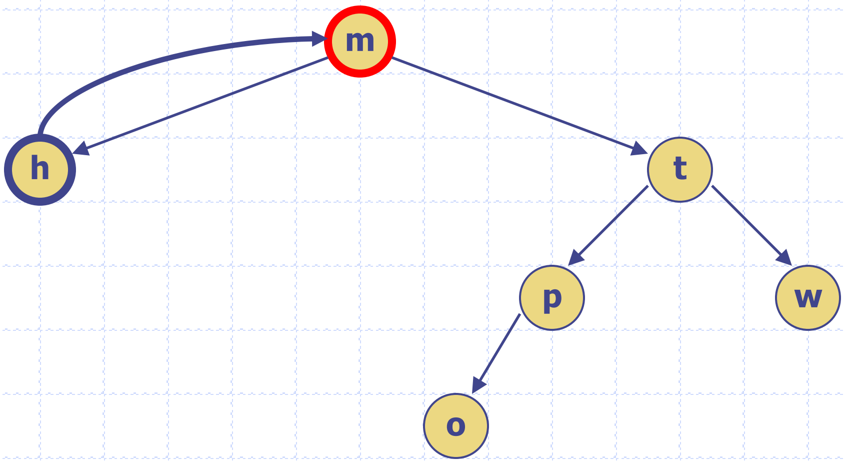
Arbori AVL

- Pentru a studia suprimarea cheilor din arbori AVL, vom apela la un artificiu care ne va permite sa facem uz tot de cele 4 cazuri de echilibrare de la insertia cheilor
- Presupunem ca dorim sa suprimam cheia 'j' din arborele de mai jos:



Arbori AVL

- Suprimand cheia ‘j’ cu tehnica de la arbori binari de cautare obisnuiti, ajungem la urmatorul arbore:

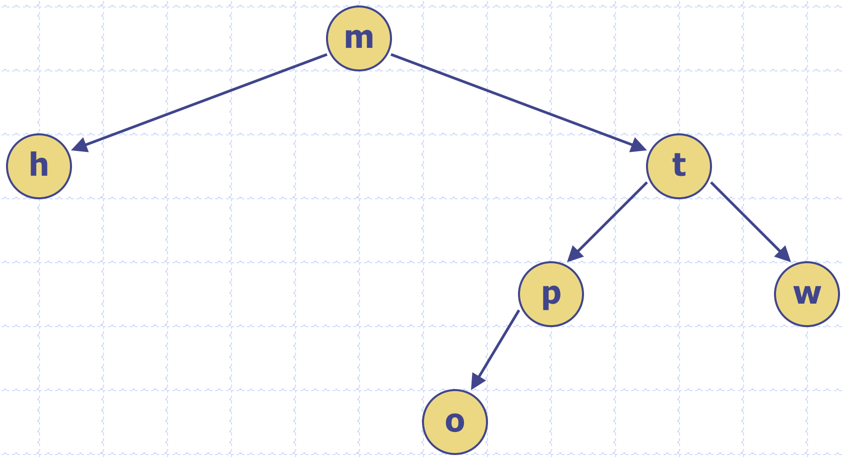


- Parcurgem drumul de la fosta locatie a cheii ‘j’ inspre radacina, inspectand toate nodurile intalnite, pentru a descoperi eventuale noduri care nu sunt echilibrate
- Primul nod care nu este echilibrat este chiar nodul ‘m’ (radacina)

Arbori AVL

- Practic, trebuie reechilibrat subarborele cu radacina ‘m’, adica intreg arborele
- Acest arbore a rezultat prin suprimarea cheii ‘j’ din subarborele stang al lui ‘m’
- Artificiul pe care-l utilizam aici este transformarea acestei suprimari intr-o insertie echivalenta care a produs un arbore dezechilibrat, si folosirea cazurilor de reechilibrare de la insertia cheilor in arbori AVL
- Suprimarea cheii ‘j’ din subarborele **stang** al lui ‘m’ este echivalenta cu situatia in care cheia ‘j’ nici nu exista de la bun inceput si a fost executata, in schimb, o insertie in subarborele **drept** al lui ‘m’ care a dus la dezechilibrarea nodului ‘m’

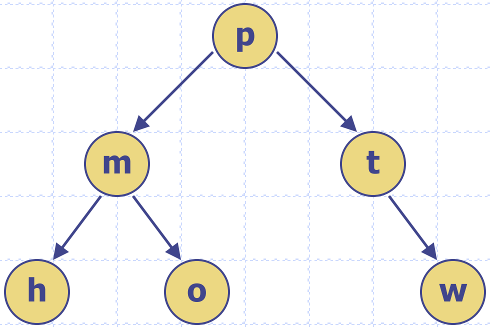
Arbori AVL



- Singura cheie care, inserata in subarborele drept al lui 'm' putea duce la dezechilibrarea nodului 'm' in situatia data este cheia 'o'
- Practic, daca am fi obtinut acest arbore prin insertia cheii 'o', am fi fost in cazul de reechilibrare 2-dreapta, adica tocmai cazul de reechilibrare pe care l-am studiat anterior in detaliu

Arbori AVL

- Prin aplicarea acestui caz de reechilibrare (fara a mai prezenta detaliile), obtinem un arbore echilibrat AVL



- Artificiul facut este echivalent cu suprimarea cheii 'j' din arborele de pe slide-ul 15 urmata de echilibrarea arborelui rezultat
- **Atentie!!!** Procesul de suprimare nu se opreste dupa echilibrarea primului nod dezechilibrat intalnit, asa cum era cazul la insertie, ci va continua cu parcurgerea drumului catre radacina si reechilibrarea tuturor nodurilor care necesita acest lucru, in aceeasi maniera
- In cazul nostru, acest lucru nu este necesar, deoarece nodul 'm' era chiar nodul radacina

Arbori AVL

- Arborii AVL reprezinta o alternativa putin costisitoare la arborii binari obisnuiti
- Cu pretul unor reechilibrari suplimentare si fara a modifica semnificativ performanta insertiei si suprimarii cheilor (celelalte operatii ramanand nemodificate), proprietatea de echilibru AVL a unui arbore binar ordonat duce la cautari mult mai rapide decat in cazul unui arbore binar ordonat obisnuit, datorita inaltimii mai mici
- S-a demonstrat ca un arbore echilibrat AVL va avea intotdeauna inaltimea cuprinsa intre **[$\log_2 N+1$]** si **[$1{,}43 \cdot \log_2 N+1$]**, unde N reprezinta numarul de chei din arbore si [x] este partea intreaga a lui x
- Spre comparatie, un arbore binar ordonat perfect echilibrat va avea intotdeauna inaltimea egala cu **[$\log_2 N+1$]** dar a-l mentine perfect echilibrat este mult mai costisitor (ca timp) decat in cazul arborilor AVL
- De asemenea, un arbore binar ordonat obisnuit va avea inaltimea cuprinsa intre **[$\log_2 N+1$]** si **N**, deci poate ajunge la inaltimi mult mai mari decat un arbore AVL cu aceleasi chei