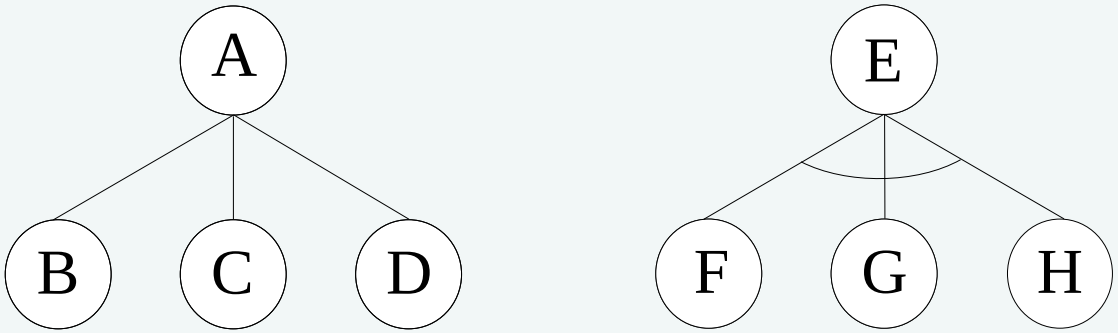
3. Metody řešení úloh rozkladem na podproblémy (AND/OR grafy)

Uzly nyní znamenají problémy/podproblémy (ne stavy !!!).

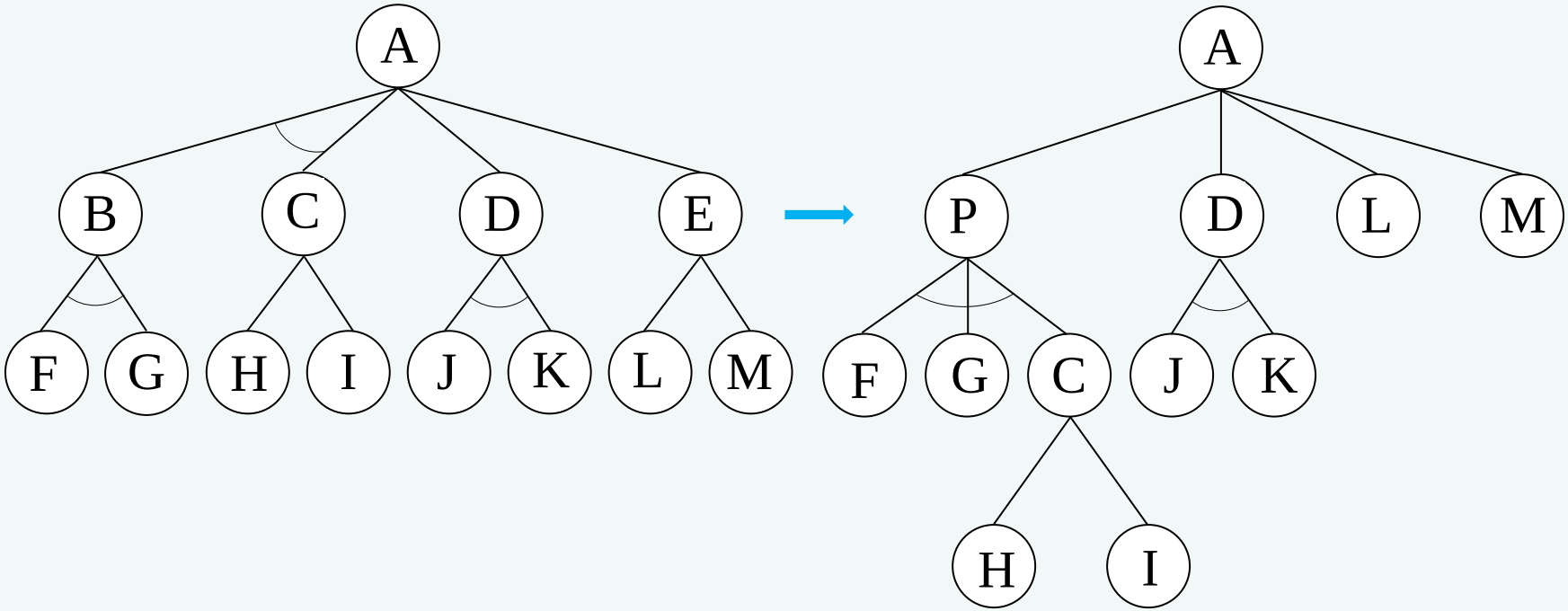
Máme dvě možnosti rozkladu problému:



Problém A je řešitelný, je-li řešitelný alespoň jeden z podproblémů B, C, D.

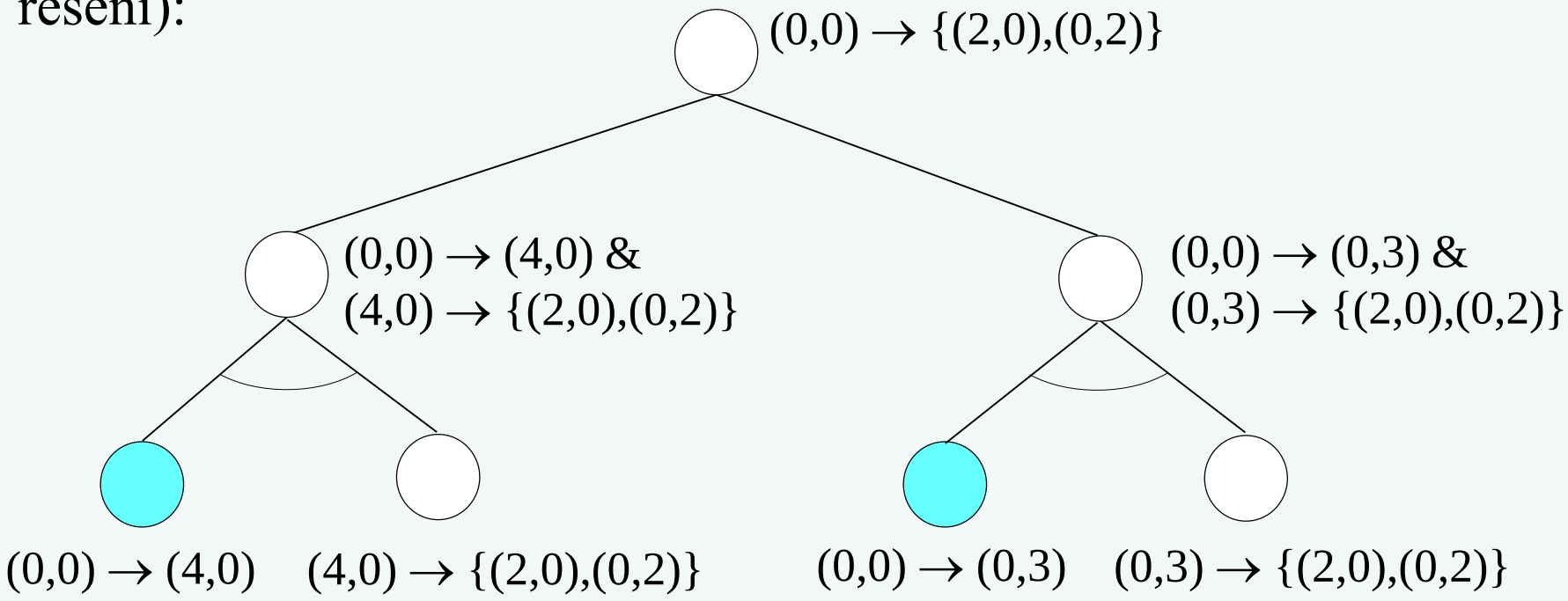
Problém E je řešitelný, jsou-li řešitelné všechny jeho podproblémy F, G a H.

Obecný AND/OR graf lze jednoduchými úpravami převést na graf, ve kterém jsou v každé vrstvě buď pouze AND nebo pouze OR uzly:

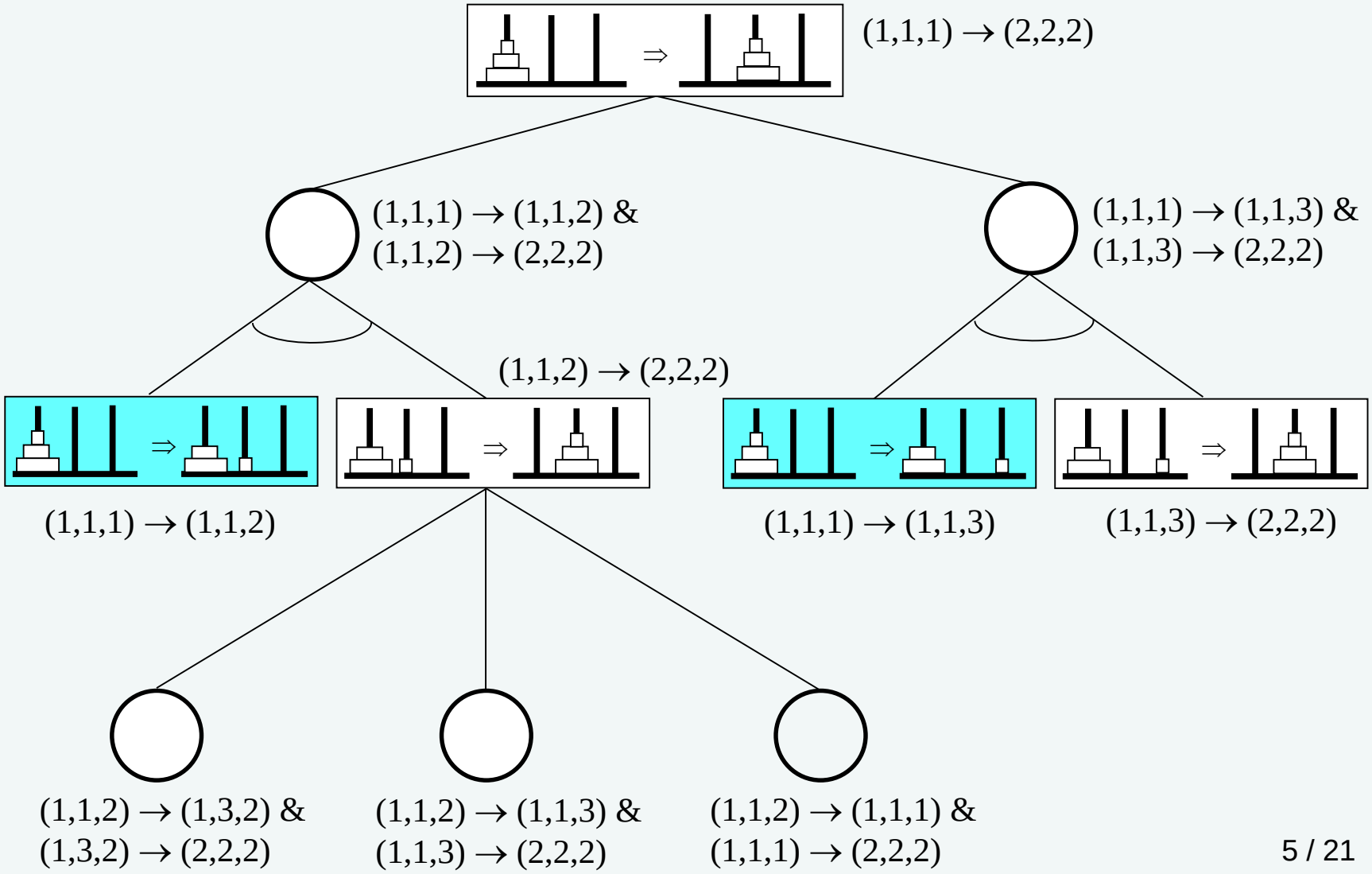


Každou úlohu je možné převést ze stavového prostoru na rozklad podproblémů. Pak každý OR uzel představuje konjunkci dvou podproblémů, z nichž jeden má vždy triviální řešení (použití některého operátoru).

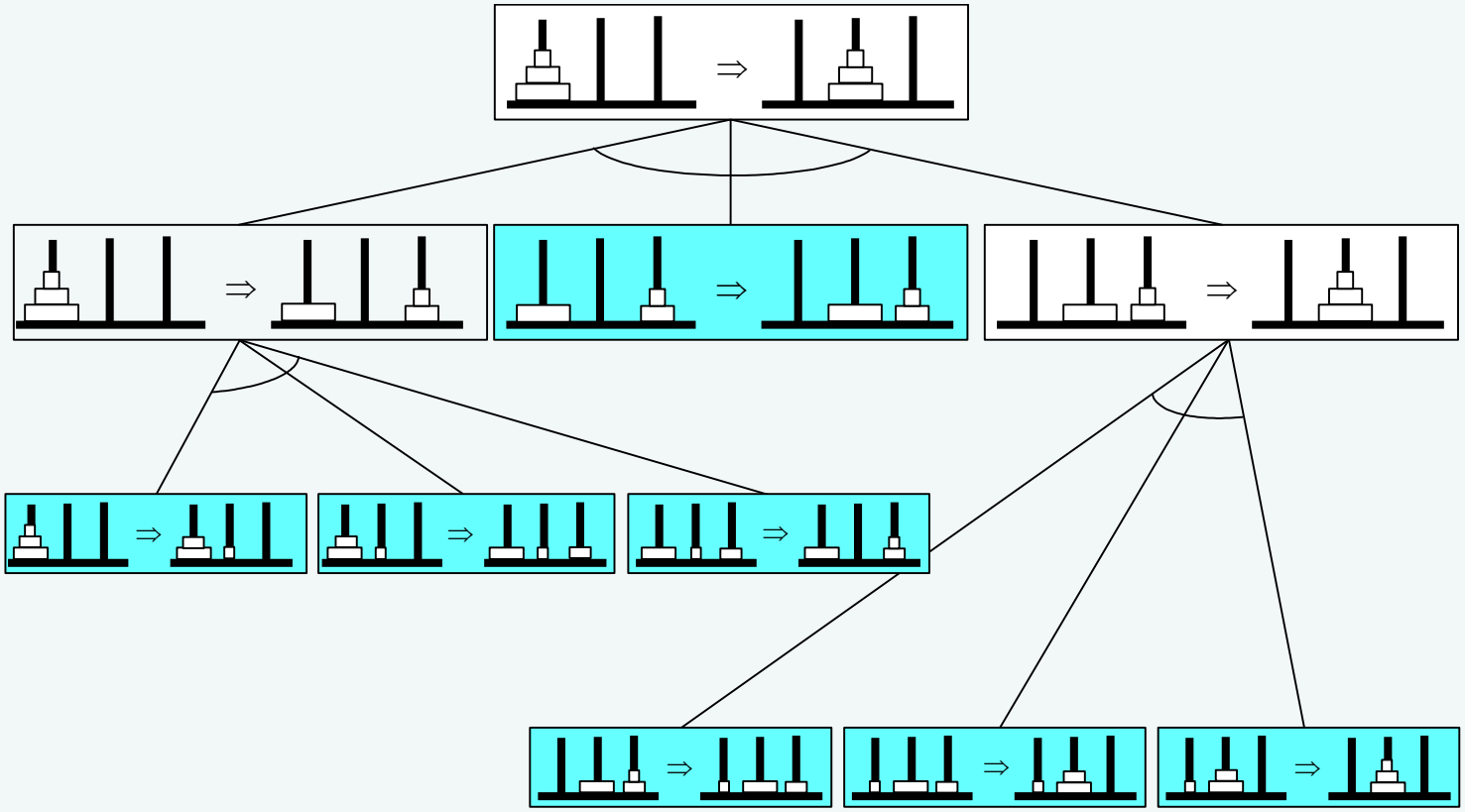
Například úloha dvou džbánů (barevně jsou označená triviální řešení):



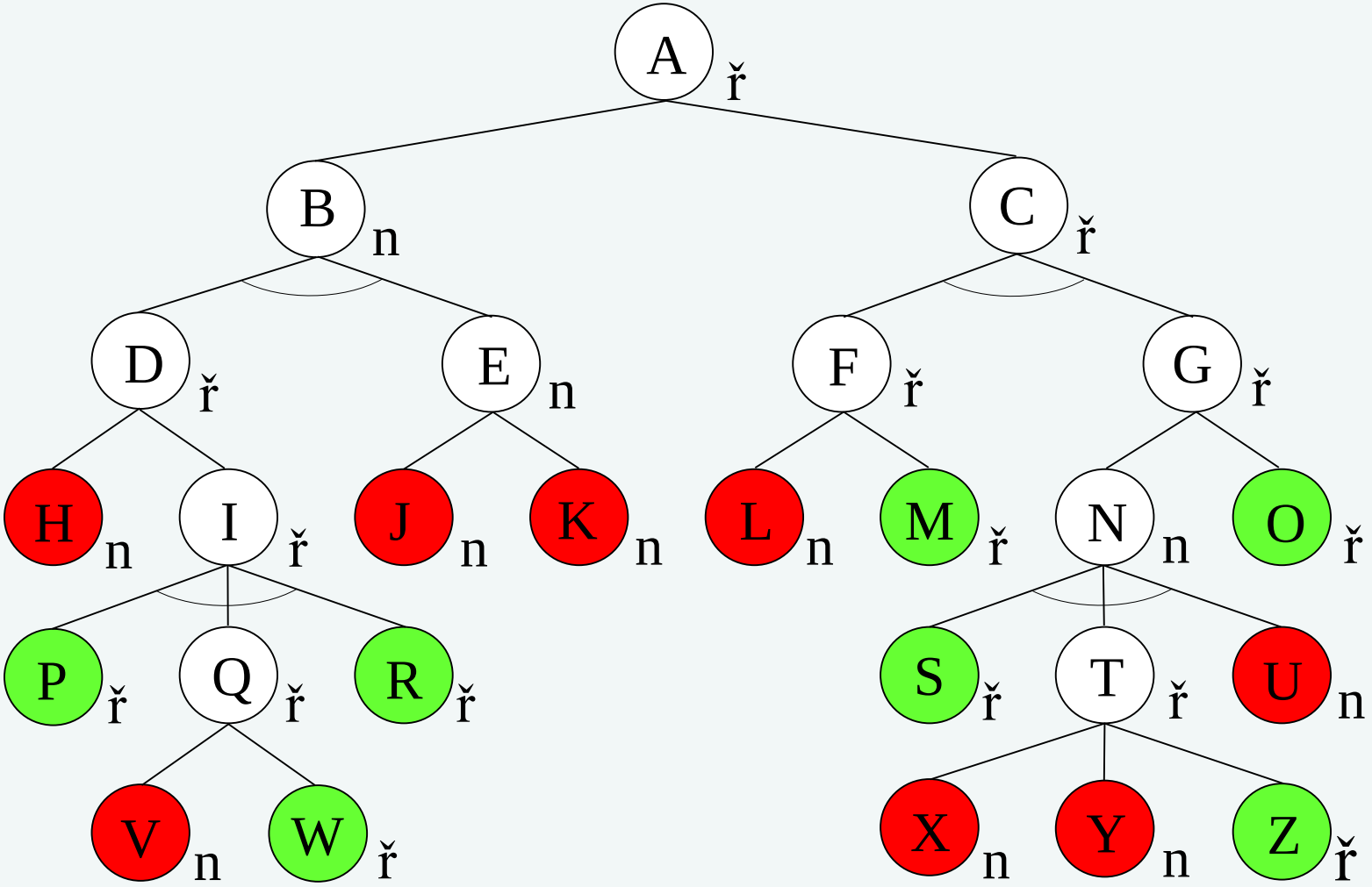
nebo úloha Hanojských věží (stavy jsou dány pozicemi jednotlivých kroužků od největšího k nejmenšímu na tyčkách zleva):



Některé úlohy je výhodné řešit pomocí rekurze, tj. „čistých“ AND grafů, například právě úloha Hanojských věží z předchozího snímku:



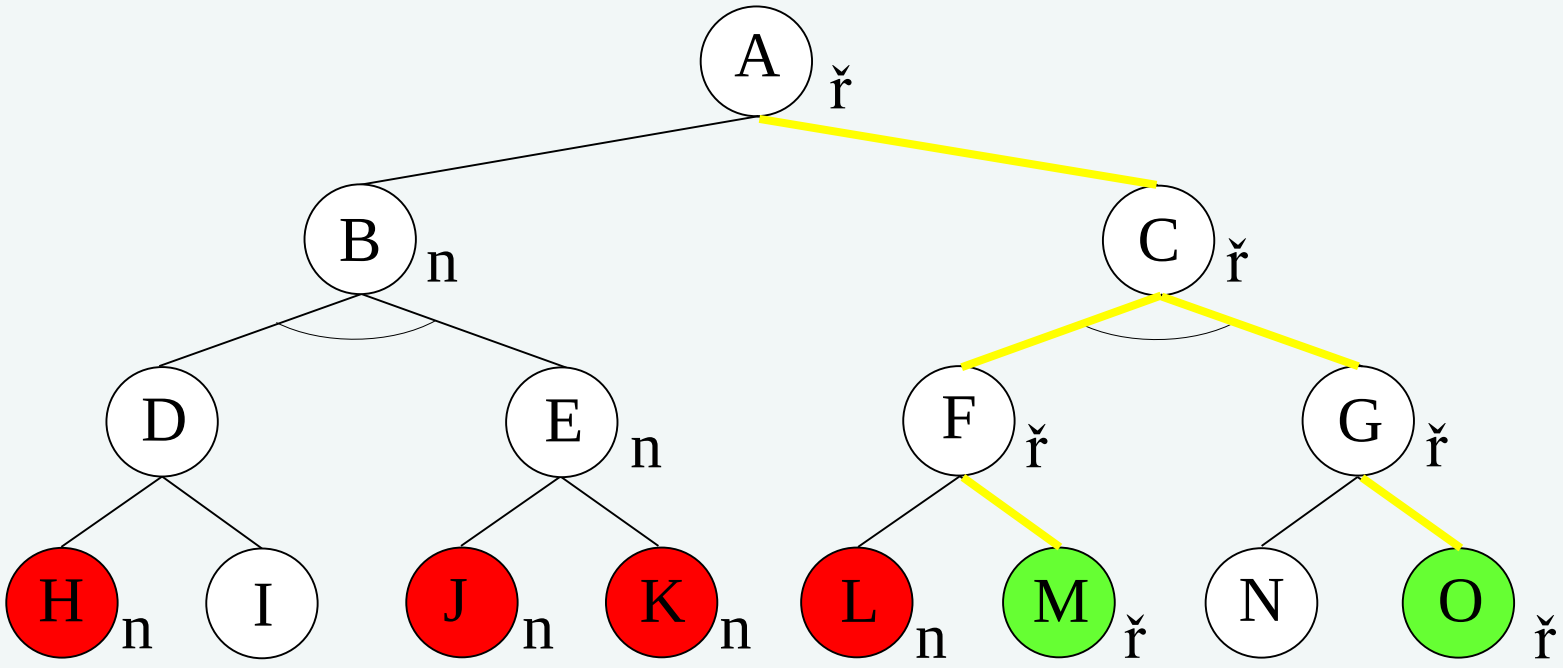
Demonstrační příklad řešení problému pomocí AND/OR grafu:



Slepé AO (AND/OR) algoritmy - BFS, DFS

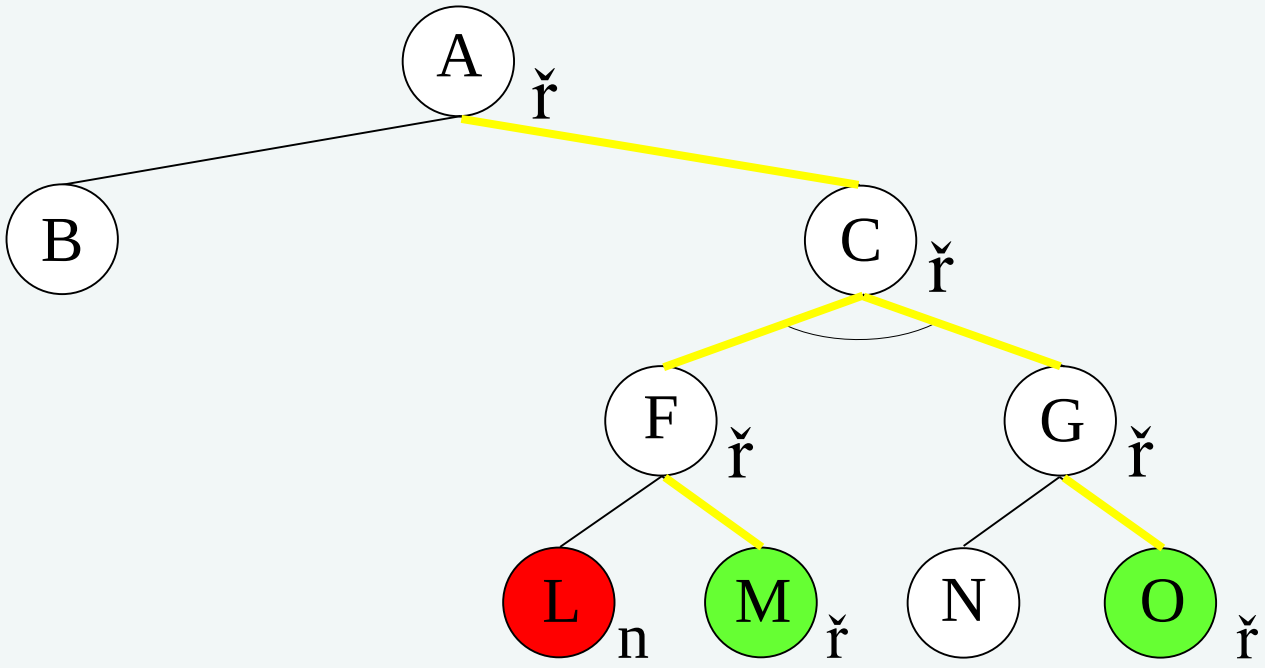
1. Sestrojte prázdný **seznam** OPEN (frontu pro BFS, zásobník pro DFS) a prázdný **graf/strom** G a do obou uložte počáteční uzel (problém), kterým nesmí být elementárně řešitelný nebo neřešitelný problém.
2. Vyjměte uzel z OPEN a označte jej jako uzel *X*.
3. Expandujte uzel *X* (rozložte *X* na podproblémy) a všechny jeho následníky připojte ke grafu G.
   a) Pro všechny řešitelné následníky uzlu *X* přeneste informaci o jejich řešitelnosti jejich předchůdcům. Je-li řešitelný počáteční problém, ukončete řešení jako úspěšné - vraťte relevantní část AND/OR grafu.
   b) Pro všechny neřešitelné následníky uzlu *X* přeneste informaci o jejich neřešitelnosti jejich předchůdcům. Není-li řešitelný počáteční problém, ukončete řešení jako neúspěšné.
   c) Všechny ostatní následníky uzlu *X* uložte do OPEN.
4. Odstraňte z OPEN všechny uzly, které mají vyřešené předchůdce.
5. Je-li seznam OPEN prázdný, ukončete řešení jako neúspěšné, jinak se vraťte na bod 2.

Příklad: AND /OR graf – BFS (animace)



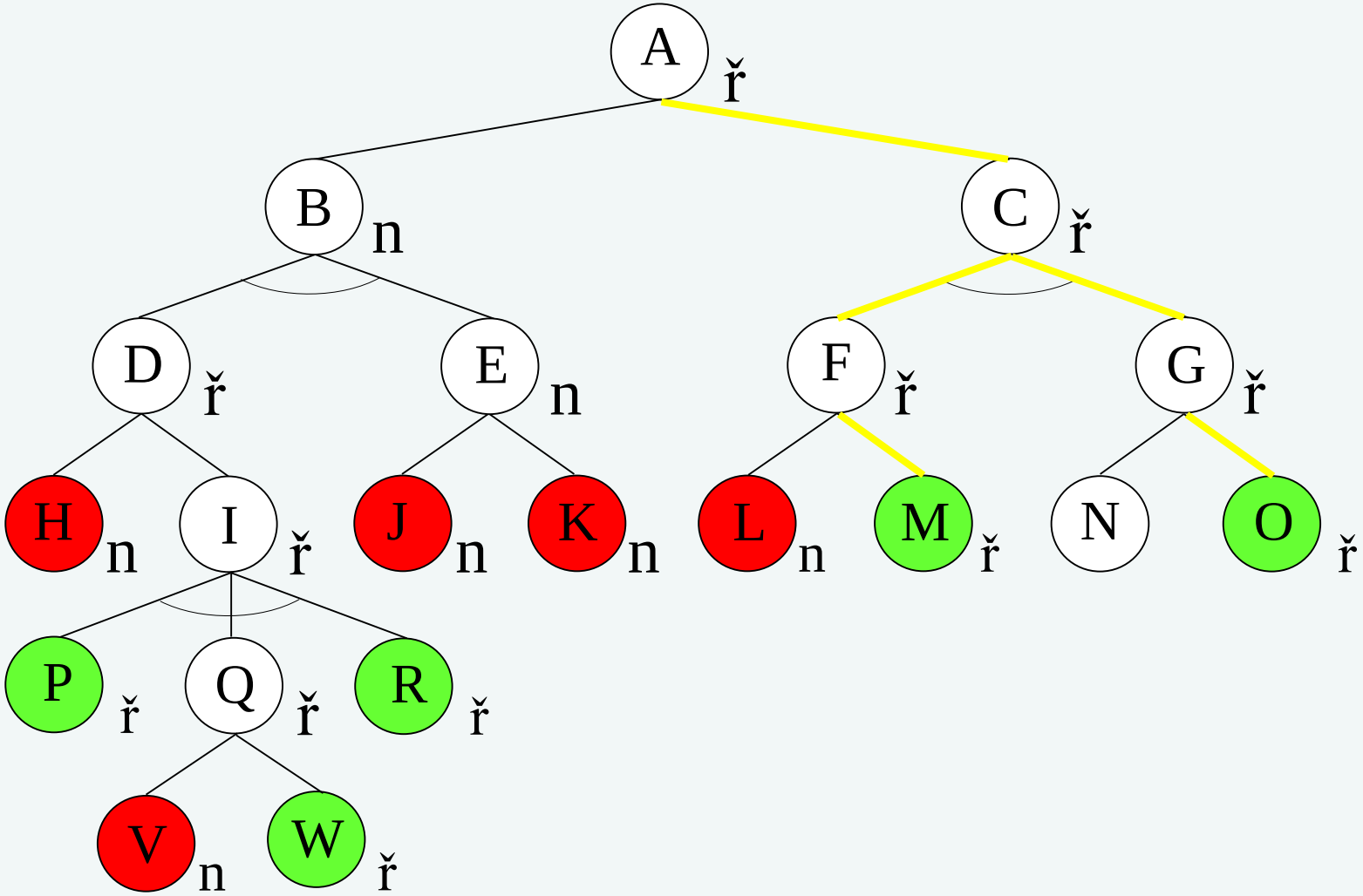
OPEN: A B C D E F G I N

Příklad: AND /OR graf – DFS (animace)



OPEN: A B C F G N

Příklad: AND /OR graf – DFS, generování následníků zprava, (animace)

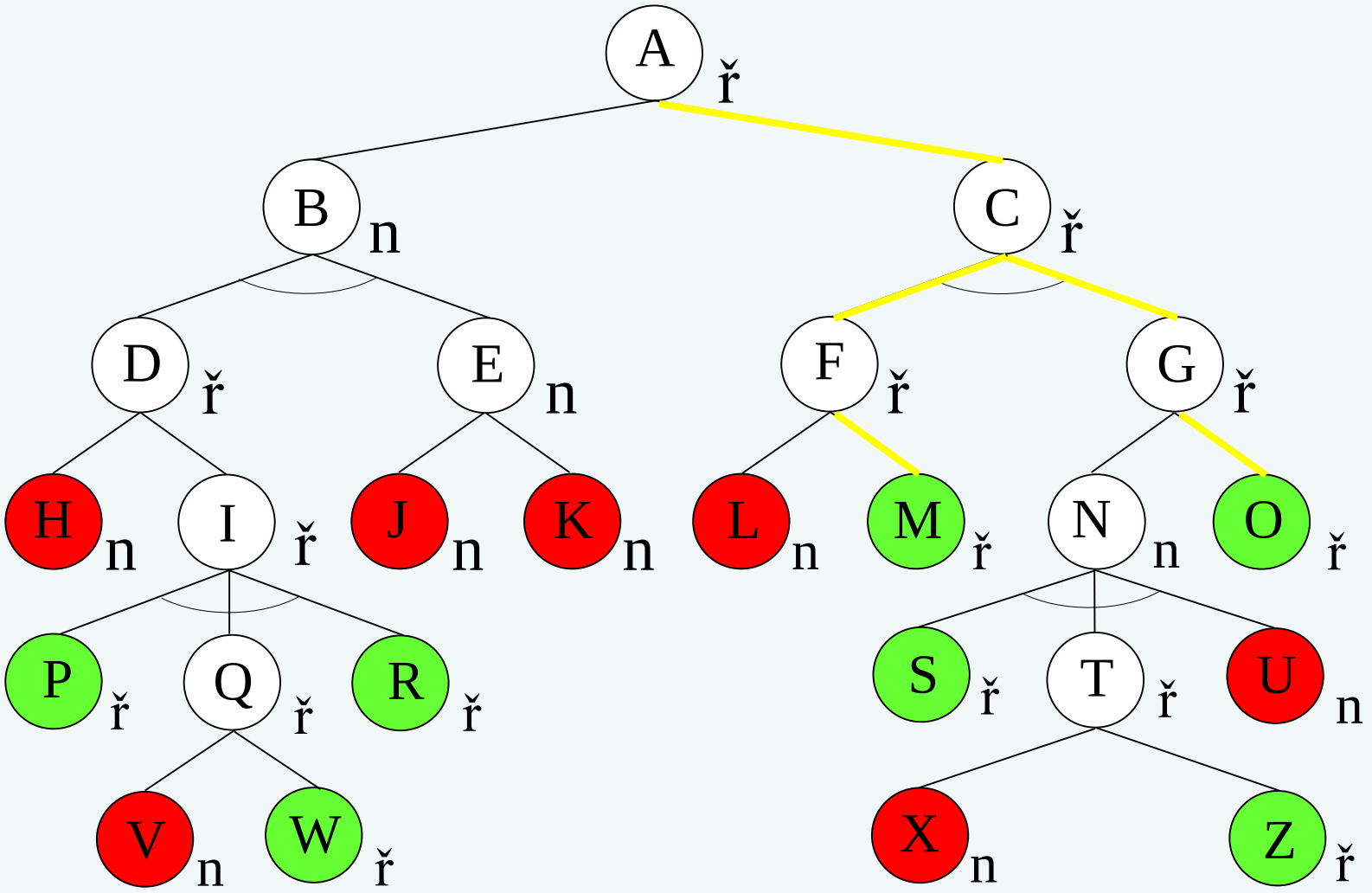


OPEN: A C B E D I Q G F N

Slepý AO algoritmus - Backtracking

1. Sestrojte **graf** G a **zásobník** OPEN a do obou uložte počáteční uzel (problém).
2. Je-li uzel na vršku OPEN řešitelný pak:
   a) Je-li tímto uzlem počáteční uzel (problém), ukončete řešení jako úspěšné (tj. vraťte relevantní část AND/OR grafu).
   b) Jinak přeneste informaci o řešitelnosti uzlu na jeho předchůdce a uzel z vršku zásobníku OPEN odstraňte.
3. Je-li uzel na vršku OPEN neřešitelný, pak:
   a) Je-li tímto uzlem počáteční uzel (problém), ukončete řešení jako neúspěšné.
   b) Jinak přeneste informaci o neřešitelnosti uzlu na jeho předchůdce a uzel z vršku zásobníku OPEN odstraňte.
4. Není-li uzel na vršku OPEN řešitelný/neřešitelný, pak generujte jeho prvního/dalšího následníka, uložte ho do OPEN a připojte ke grafu G.
5. Vraťte se na bod 2.

Příklad: AND /OR graf – Backtracking (animace)



OPEN: A B D H I P Q V WR E J K C F L M G N S T X Z U O

Úloha balančních vah (příklad klasického AND/OR grafu):

Jak bylo uvedeno v první přednášce, je cílem úlohy nalézt pomocí minimálního počtu vážení na balančních vahách minci s mírně odlišnou vahou od ostatních mincí. Dále uvažujme klasickou úlohu s 12-ti mincemi:

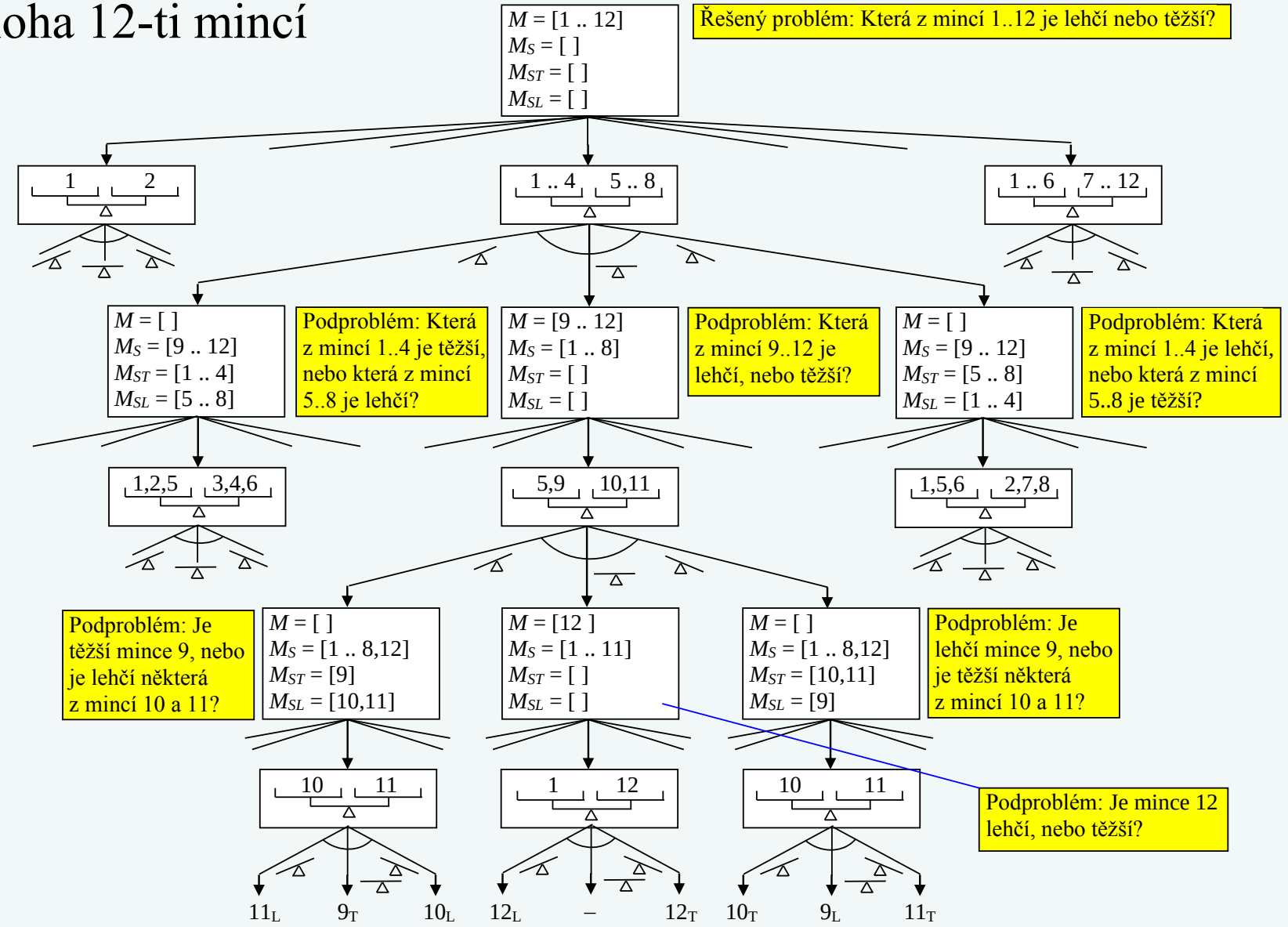
1. Ke každému vážení vybereme různé kombinace mincí (smysl mají pouze vážení, při kterých je na obou miskách jazýčkových vah stejný počet mincí). Jde tedy o problémy/uzly typu OR.
2. Na základě výsledku každého vážení musíme být schopni zpracovat výsledek tohoto vážení (levá miska je níže, pravá miska je níže, misky jsou ve vodorovné poloze). Jde tedy o problémy/uzly typu AND.

Připomeňme, že stavy úlohy lze reprezentovat čtveřicemi seznamů

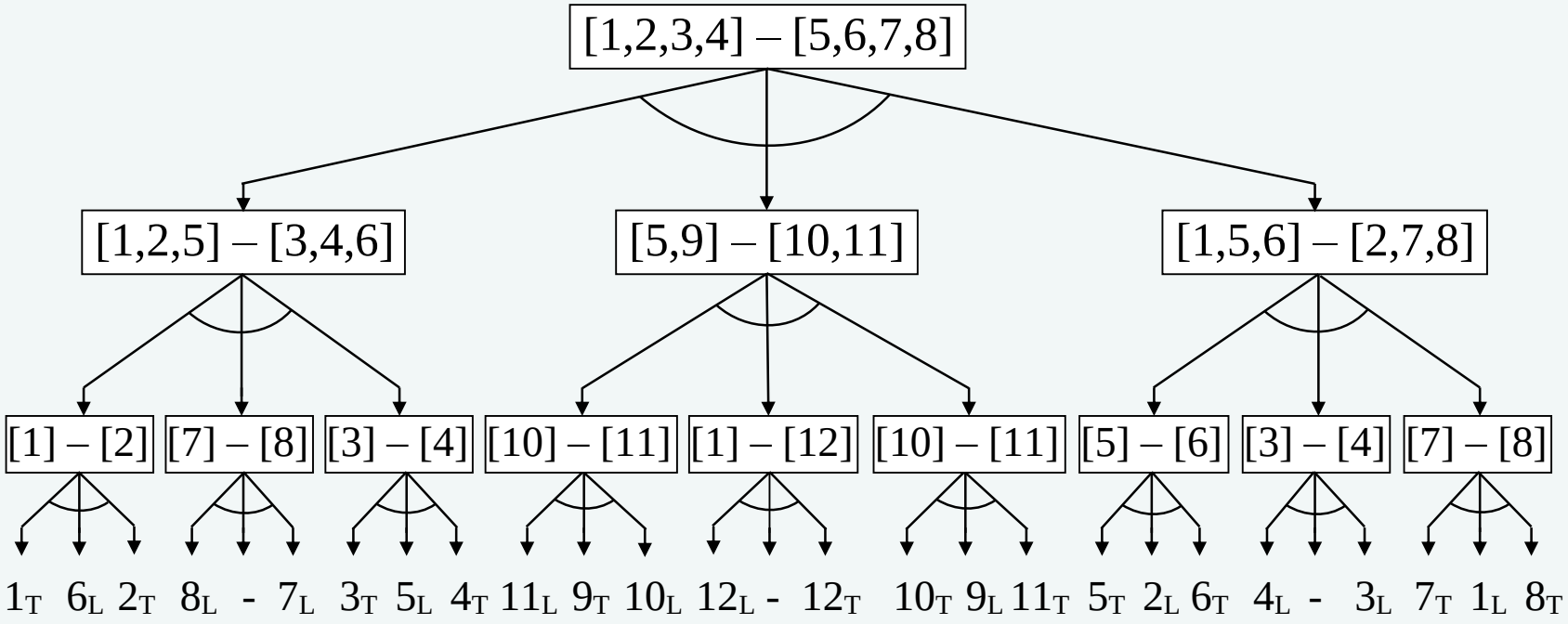
$s = (M, M_S, M_{SL}, M_{ST})$, kde značí

M množinu mincí s dosud nerozhodnutou váhou,
M_S množinu mincí se stejnými vahami,
M_{SL} množinu mincí, které jsou buď všechny stejné, nebo jedna z nich je lehčí,
M_{ST} množinu mincí, které jsou buď všechny stejné, nebo jedna z nich je těžší.

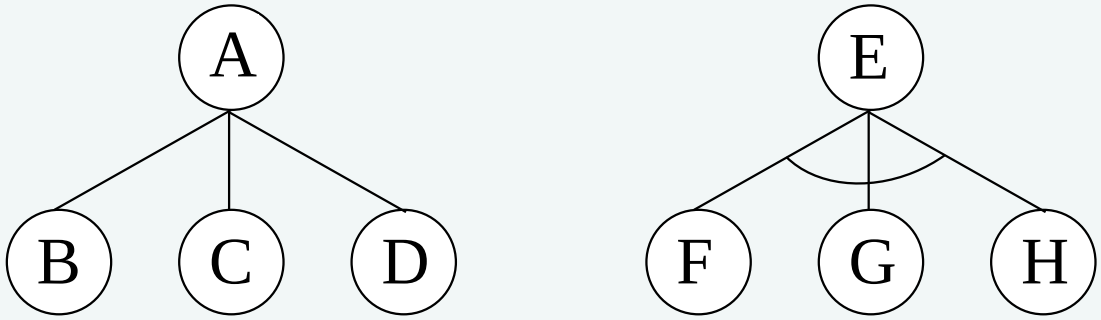
Úloha 12-ti mincí



Úloha 12-ti mincí – optimální řešení:



Informovaný AND/OR algoritmus – AO*



h(A) = min(h(B),h(C),h(D))

h(E) = h(F) + h(G) + h(H)

h(x) … odhad ceny řešení daného podproblému

Algoritmus vybírá k prohledávání vždy „nejnadějnější“ podstrom každého OR uzlu, počínaje kořenovým uzlem.

Informovaný AO* algoritmus

1. Sestrojte strukturu G pro reprezentaci AND/OR stromu a umístěte do ní počáteční uzel (označený jako INIT) s jeho ohodnocením. Stanovte hodnotu FUTILITY, která určuje maximální povolenou cenu řešení.
2. Je-li uzel INIT označen jako řešitelný (SOLVED), ukončete řešení jako úspěšné (řešení je dáno nejnadějnějším podstromem stromu G). Je-li hodnota uzlu INIT větší nebo rovna hodnotě FUTILITY, ukončete prohledávání jako neúspěšné. Jinak pokračujte.
3. Procházejte nejnadějnějším podstromem až narazíte na neexpandovaný uzel. Tento uzel označte NODE.
4. Expandujte uzel NODE. Jestliže NODE nemá žádného následníka, přiřaďte mu hodnotu FUTILITY (je ekvivalentní sdělení, že uzel není řešitelný (UNSOLVED)) a přejděte na bod 7. Jinak pokračujte.
5. Představují-li někteří bezprostřední následníci elementární úlohy, označte je jako SOLVED (tj. přiřaďte jim nulové ohodnocení), u zbývajících ohodnocení odhadněte (vypočítejte).

6. Připojte bezprostřední následníky uzlu NODE ke stromu G a přeneste informace o jejich ohodnocení směrem nahoru k uzlu INIT takto:
    - ohodnocení uzlu AND je rovno součtu ohodnocení všech jeho bezprostředních následníků.
    - ohodnocení uzlu OR je rovno nejmenšímu z ohodnocení jeho bezprostředních následníků.
7. V uzlech OR označte nejnadějnější podstromy (vždy hranu k nejlépe ohodnocenému bezprostřednímu následníku).
8. Vraťte se na bod 2.

AO* příklad (animace)