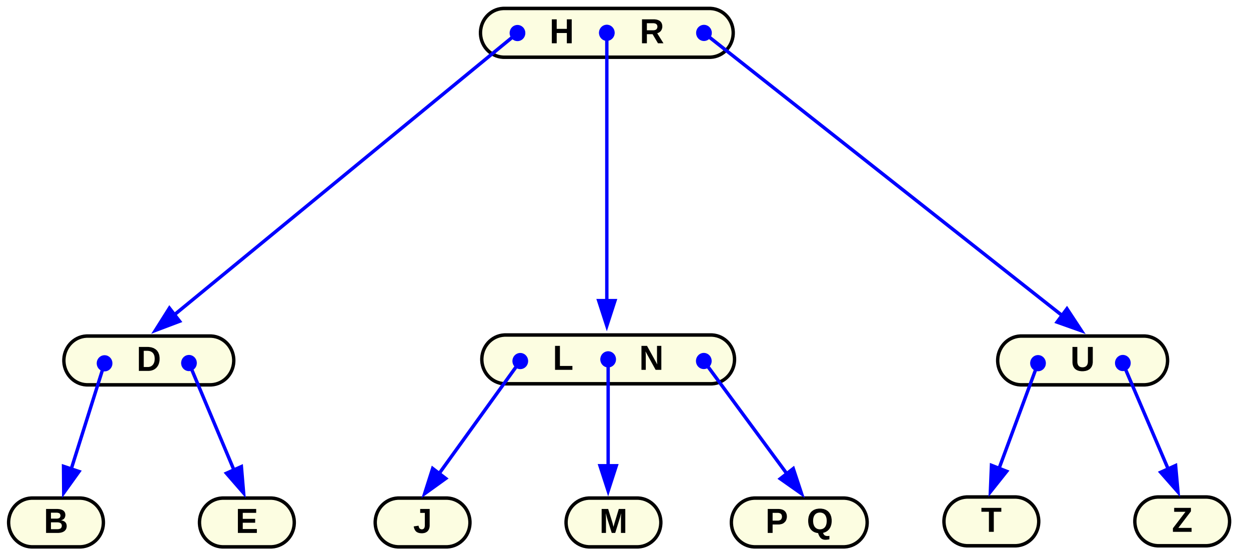
2-3 strom
(2-3 tree)

2-3 strom je **perfektne vyvážený** (usporiadaný) vyhľadávací strom, v ktorom je dĺžka cesty z koreňa do každého z listov rovnaká (na rozdiel napr. od AVL stromu, ktorý nie je perfektne vyvážený). **Štruktúra podporuje jednorozmerné intervalové vyhľadávanie.**
Pre vrcholy 2-3 stromu platí, že potenciálne disponujú buď **2** alebo **3** synmi/potomkami (tento typ stromu možno teda klasifikovať ako „maximálne“ trojcestný a „minimálne“ binárny).
Vo všeobecnosti môže byť kľúčom akýkoľvek dátový typ. V príkladoch bude použitý kľúč s jediným znakom.



Základné charakteristiky 2-3 stromu

a) Každý **list má rovnakú hĺbku** a obsahuje buď **1** alebo **2 záznamy** (**hĺbku** vrchola **v** v strome s koreňom **R** definujeme ako dĺžku cesty medzi **R** a **v**).

b) Každý interný vrchol (nie je listom) má buď **jeden záznam** a **dvoch synov** (nazývame ho **2-vrchol**) alebo **dva záznamy** a **troch synov** (**3-vrchol**)

Nech záznam **Z** obsahuje kľúč **K**.
Každému kľúču **K** z interného vrcholu **v** je fixne priradený jeho priľahlý ľavý „**Ľavý**(v_K)“ a pravý „**Pravý**(v_K)“ **syn** (podstrom), pričom pre dva kľúče **K1** a **K2** (**K1**<**K2**) z vrcholu **v** platí: **Pravý**(v_{K1}) = **Ľavý**(v_{K2}), **Pravý**(v_{K2}) ≠ **Levý**(v_{K1}).

c) Pre každý kľúč **K** z interného vrcholu platí, že je väčší než všetky kľúče z jeho ľavého priľahlého podstromu a menší než všetky kľúče z jeho pravého priľahlého podstromu.

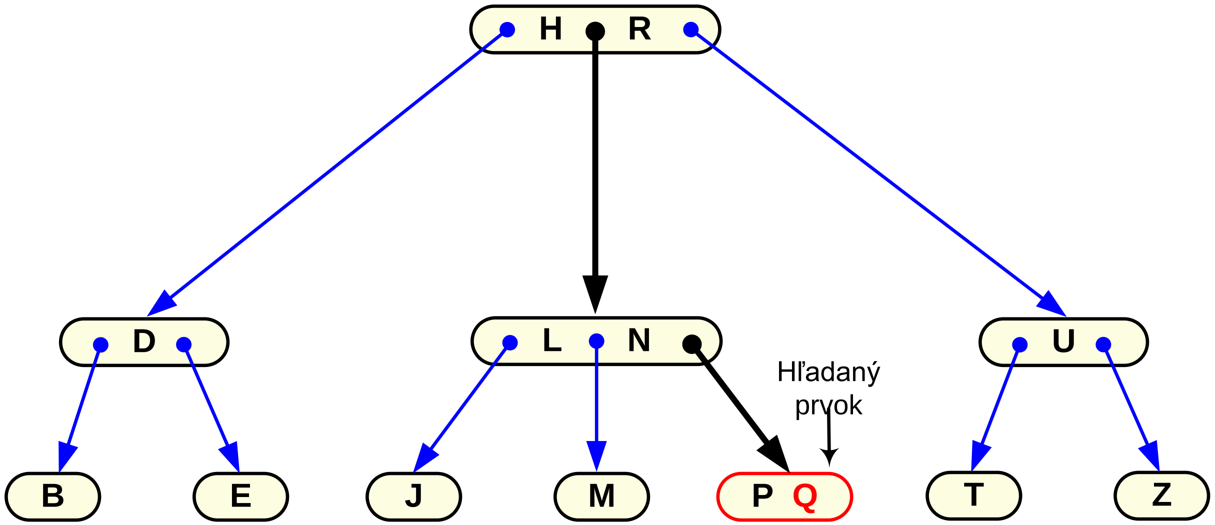
Ak 2-3 strom obsahuje iba **2-vrcholy**, tak počet záznamov **n** v ňom (totožný s počtom vrcholov) je **n** = $2^{h+1} - 1$ (výška **h** = $\lfloor \log_2 n \rfloor$).
Ak 2-3 strom obsahuje výhradne **3-vrcholy**, tak počet záznamov **n v ňom** (počet vrcholov = **n**/2) je **n** = $3^{h+1} - 1$ (výška **h** = $\lfloor \log_3 n \rfloor$).
Zložitosť operácií sa preto pohybuje medzi **O($\log_3$n) po O($\log_2$n).**

Operácia Nájdi - priemerná aj najhoršia zložitosť $O(\log_2 n)$:

Hľadáme kľúč **K.** Pri porovnávaní hľadaného kľúča **K** s kľúčmi aktuálne dosiahnutého vrchola (začíname v koreni stromu) **v** 2-3 strome, môžu nastať nasledujúce prípady:

a) Ak **v** je 3-vrchol (s kľúčmi **K1** a **K2** , **K1** < **K2**), tak:
   - ak **K** $\in$ { **K1** , **K2** } tak je operácia **Nájdi** úspešne ukončená,
   - ak **K** < **K1** tak hľadanie pokračuje vo vrchole **Ľavý**(v_{K1}), pokiaľ existuje,
   - ak **K** > **K1** $\wedge$ **K**<**K2** tak hľadanie pokračuje vo vrchole **Ľavý**(v_{K2}), pokiaľ existuje,
   - ak **K** > **K2** tak hľadanie pokračuje vo vrchole **Pravý**(v_{K2}), pokiaľ existuje.

b) Ak **v** je 2-vrchol, tak je aplikovaný rovnaký postup ako pri BVS.

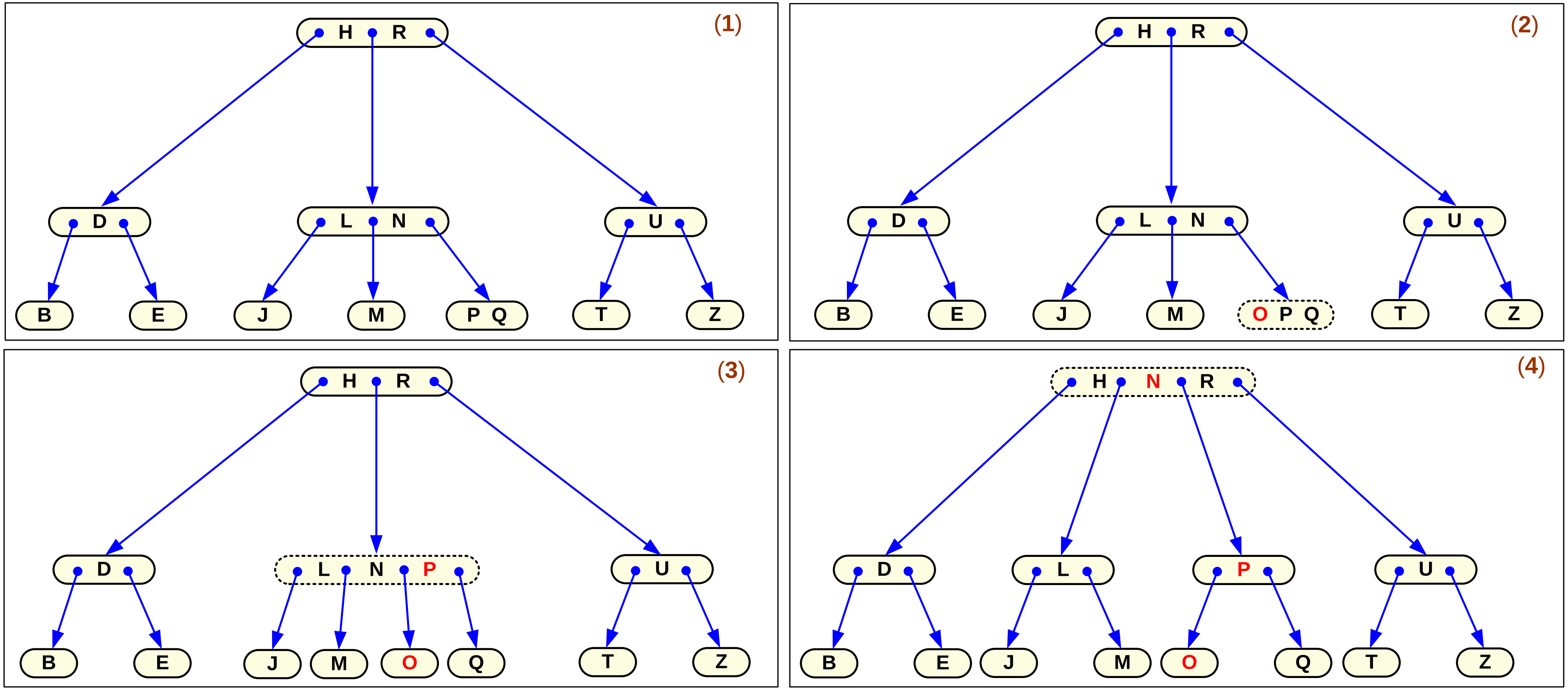


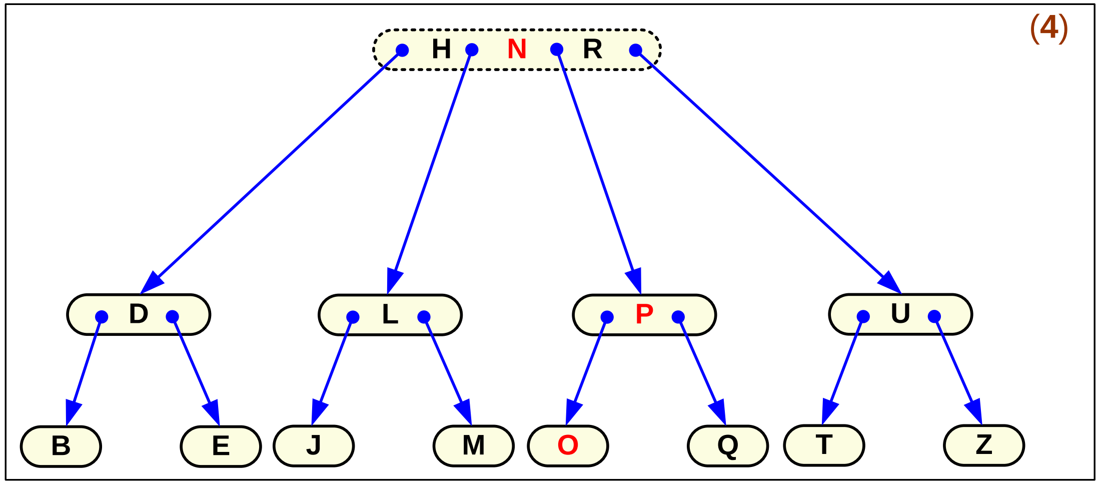
Operácia Vlož - priemerná aj najhoršia zložitosť $O(\log_2 n)$

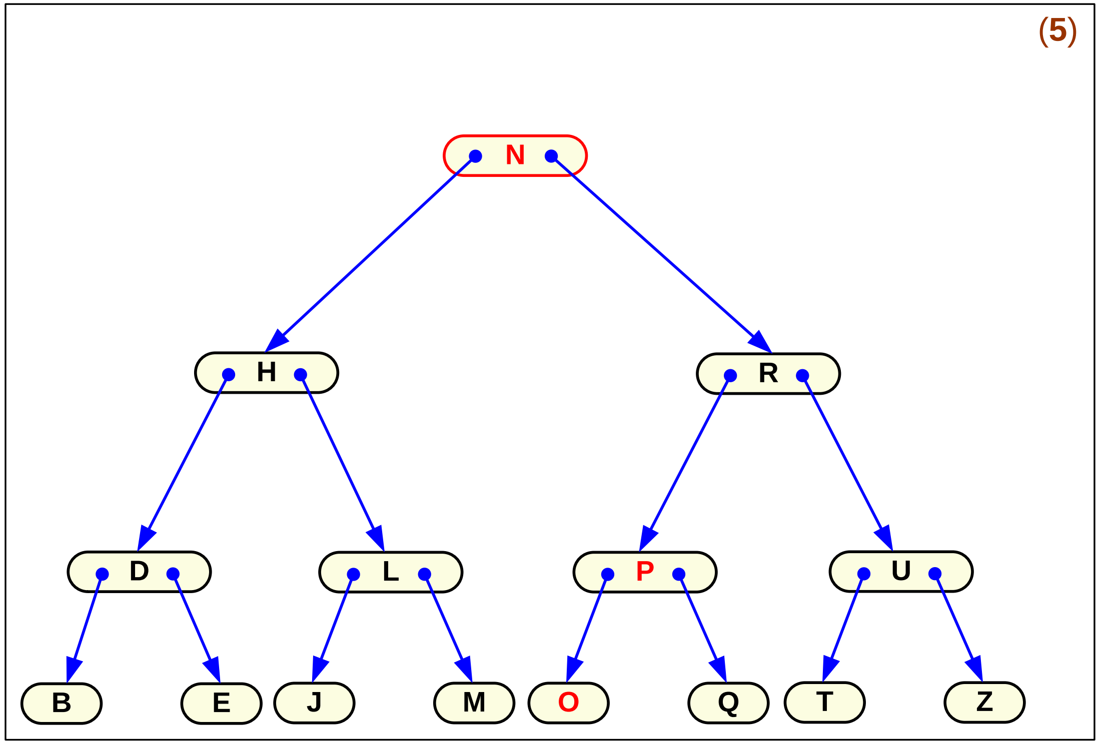
a) Nájdenie listu **L**, do ktorého patrí element **Prvok** s kľúčom **K**.
b) Ak **L** je 2-vrchol, tak doňho vložíme **Prvok**, koniec operácie,
c) Ak **L** je 3-vrchol (s kľúčmi **K1** a **K2** , **K1**<**K2**), rozdelíme ho na dva 2-vrcholy $\mathbf{L_{min}}$ a $\mathbf{L_{max}}$, ktoré budú mať kľúče $\mathbf{K_{min}}$ = **min**(**K1**, **K2**, **K**) a $\mathbf{K_{max}}$ = **max**(**K1**, **K2**, **K**), pričom $\mathbf{K_{stred}} \in \{\mathbf{K1},\mathbf{K2},\mathbf{K}\} - \{\mathbf{K_{min}}, \mathbf{K_{max}}\}$ môže byť následne potenciálne uložený do vrchola **Otec**(**L**), pokiaľ existuje.
d) Ak vrchol **Otec**(**L**) je 2-vrchol, tak ho zmeníme na 3-vrchol (pridaním kľúča $\mathbf{K_{stred}}$) a algoritmus končí. Ak **Otec**(**L**) je 3-vrcholom (došlo by k „pretečeniu“) vykoná (nastaví) sa: **L** = **Otec**(**L**) a **K** = $\mathbf{K_{stred}}$, návrat na bod c).

Uvedený postup sa uplatňuje postupne smerom ku koreňu stromu pokiaľ sa vo vrchole nenájde „voľné miesto“ pre presúvaný prvok alebo pokiaľ nedôjde k rozdeleniu koreňa - vtedy sa vytvorí nový koreň (2-vrchol) a výška stromu sa zvýši o jedna.

Príklad vloženia prvku s kľúčom O:




(4)
H
N
R
D
L
P
U
B
E
J
M
O
Q
T
Z


(5)
N
H
R
D
L
P
U
B
E
J
M
O
Q
T
Z

Operácia Odober - priemerná aj najhoršia zložitosť $O(\log_2 n)$

a) Ak je **Prvok** (s kľúčom **K**) v liste, tak ho z neho odoberieme. Inak ho nahradíme jeho „inorder-nasledovníkom“, ktorý odoberieme z príslušného listu.

b) Označme symbolom **v** vrchol, z ktorého bol aktuálne odobratý prvok. Ak je vo **v** práve jeden prvok, algoritmus končí. Ak vo **v** nie je žiaden prvok („podtečenie“), môžu nastať 3 prípady:

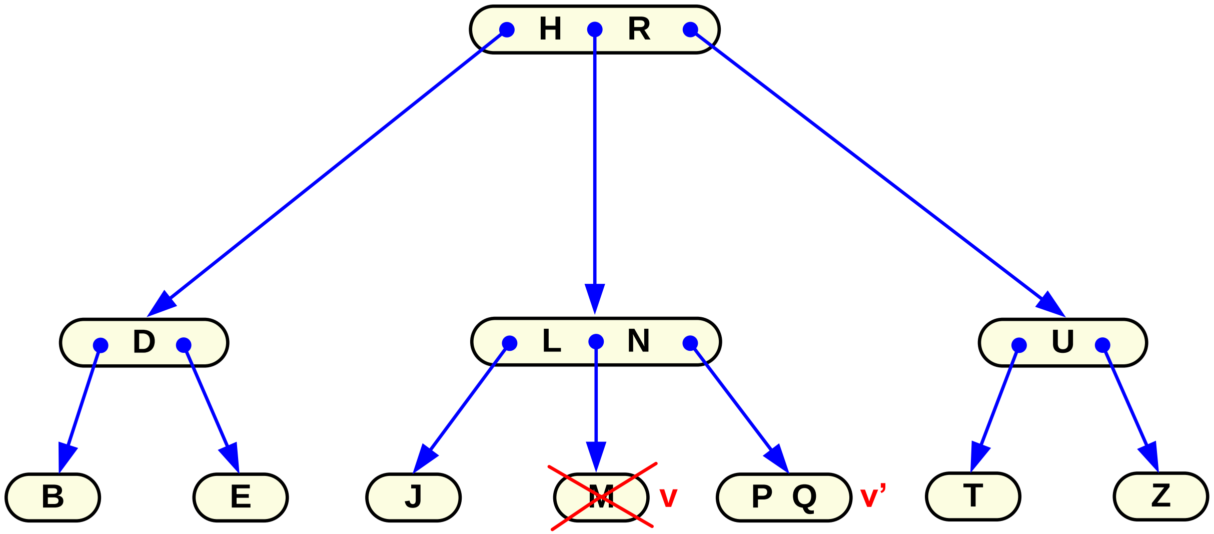
1. Vrchol **v** je **koreň**, ktorý je odstránený; ak koreň mal syna, tak sa tento stáva koreňom, inak sa strom stáva prázdnym.

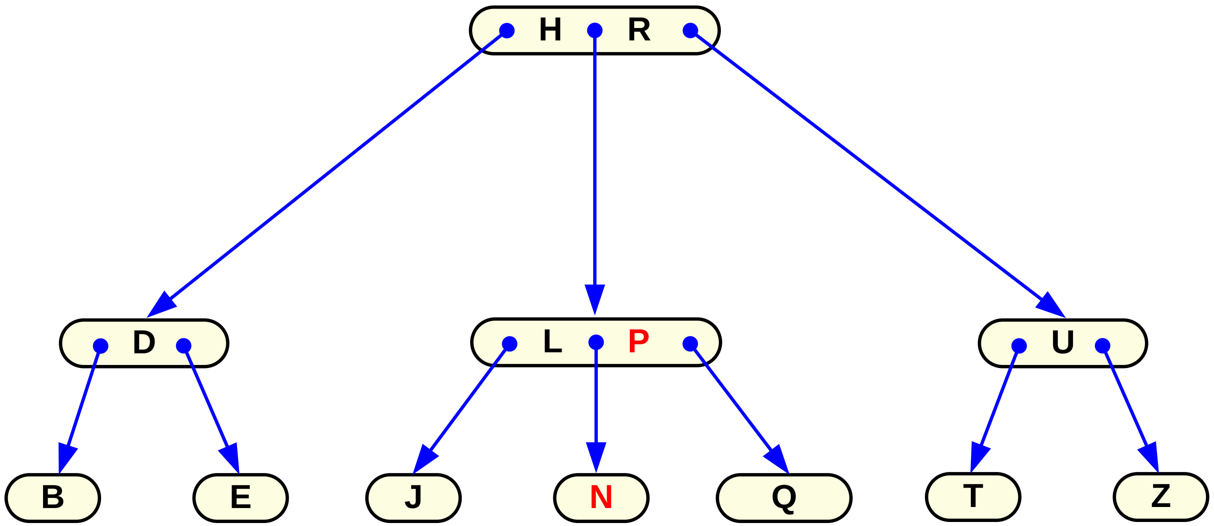
2. Ak má vrchol **v** brata **v’**, ktorý sa nachádza priamo po jeho „pravici“ alebo „ľavici“ a **obsahuje dva prvky**, potom označme ako $\mathbf{K_O}$ kľúč prvku v uzle **Otec**(**v**) ≡ **Otec**(**v’**), ktorý ich separuje. Prvok s kľúčom $\mathbf{K_O}$ presunieme do **v** a do prvku **Otec**(**v**) presunieme prvok z **v’**, ktorého kľúč je priľahlý k vrcholu **v**. Ak sú **v** a **v’** interné vrcholy (nelisty), potom sa presunie aj referencia na príslušného syna z **v’** do **v**. Vrcholy **v’** a **v** sa stanú 2-vrcholmi a pre tento prípad je algoritmus ukončený.

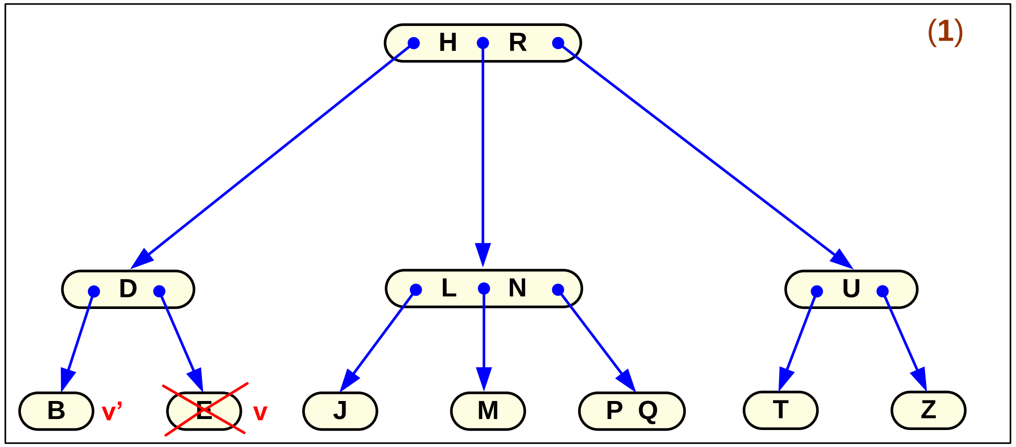
.

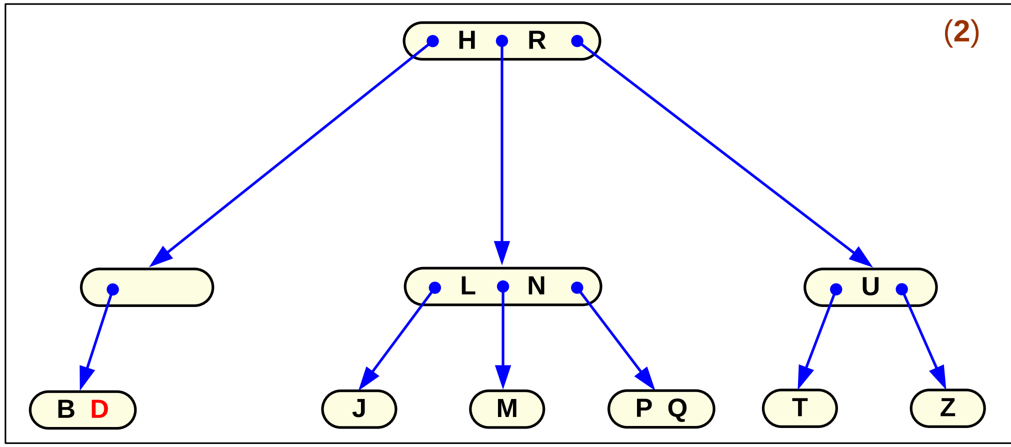
3. V tomto prípade vrchol **v** má brata **v’**, ktorý sa nachádza priamo po jeho „pravici“ alebo „ľavici“ a **obsahuje iba jeden prvok**. Nech $\mathbf{K_O}$ je kľúč prvku z vrcholu **Otec**(**v**) ≡ **Otec**(**v’**), ktorý separuje **v** a **v’**. Prvok s kľúčom $\mathbf{K_O}$ a prvok z **v’** sa zlúčia do nového 3-vrchola, ktorý nahradí **v’** a **v** (V pôvodnom vrchole **Otec**(**v**) ≡ **Otec**(**v’**) sa tým zníži o jedna počet prvkov i počet synov). Vykoná (nastaví) sa: **v** = **Otec**(**v**) a bod b) sa opakuje.

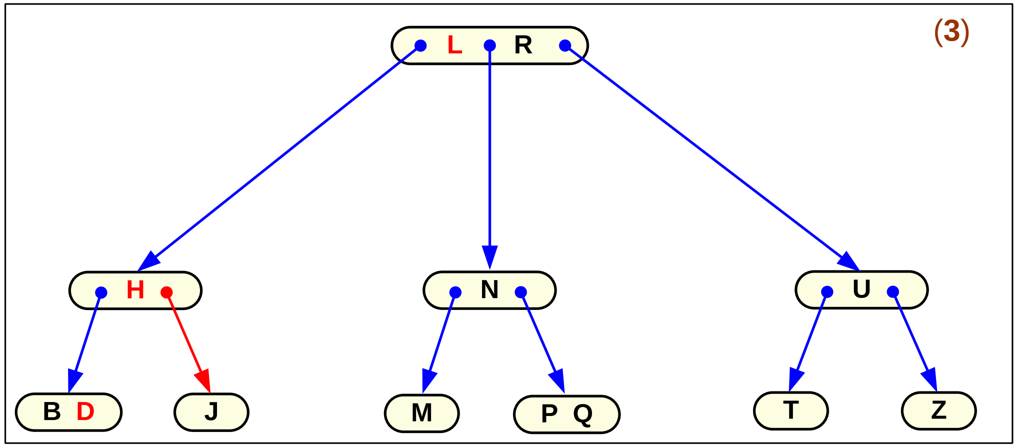
Ilustrácia odobratia prvku s kľúčom M











Ilustrácia odobratia prvku s kľúčom E