

Дървета за търсене



Двоично наредено дърво

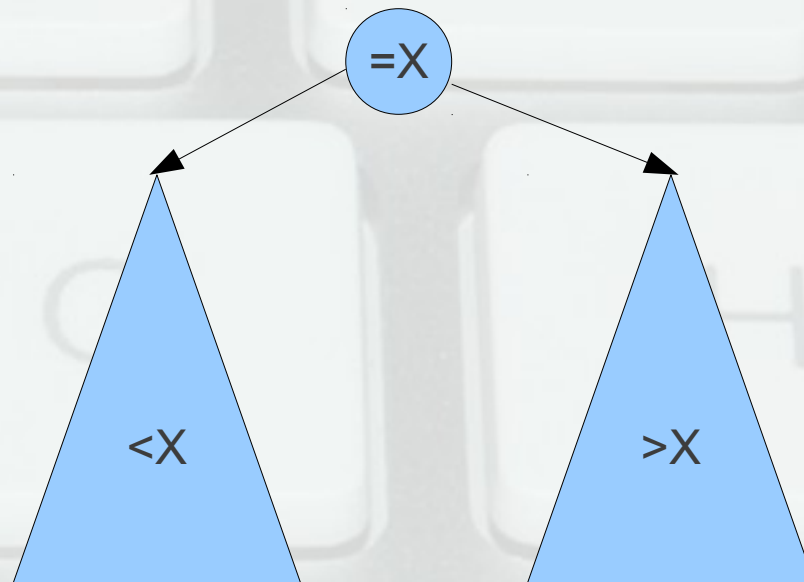
- Празното дърво е ДНД
- (X, L, R) е ДНД, ако:
 - X е по-голямо от всички върхове в L
 - X е по-малко от всички върхове в R
 - L и R са ДНД

Двоично наредено дърво

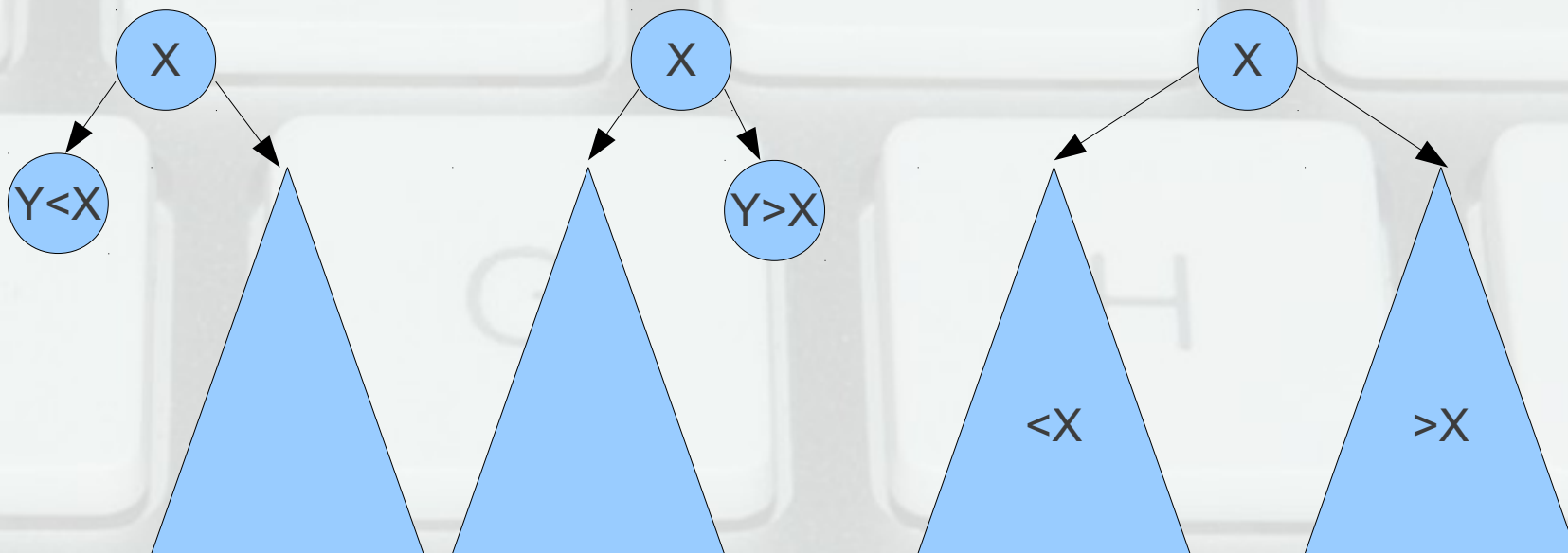
- Операции

- достъп до корена
- търсене на елемент
- включване на елемент
- изключване на елемент
- възходящо/низходящо обхождане

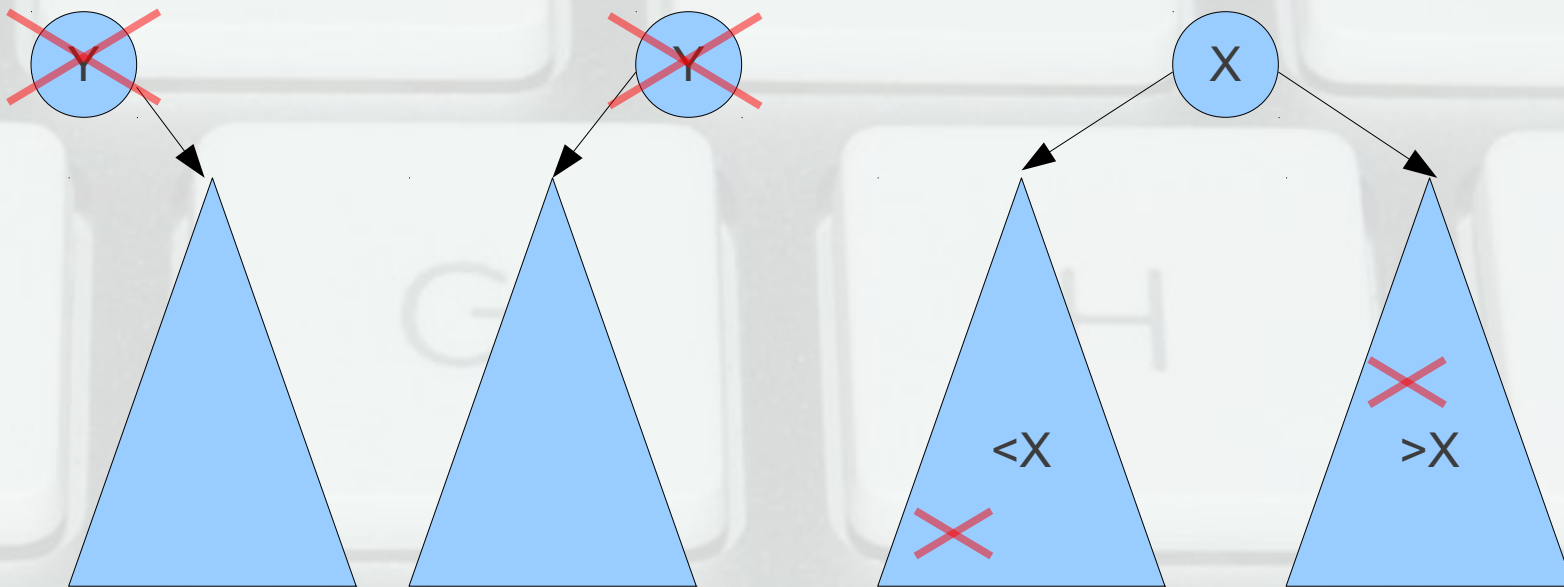
Търсене на елемент



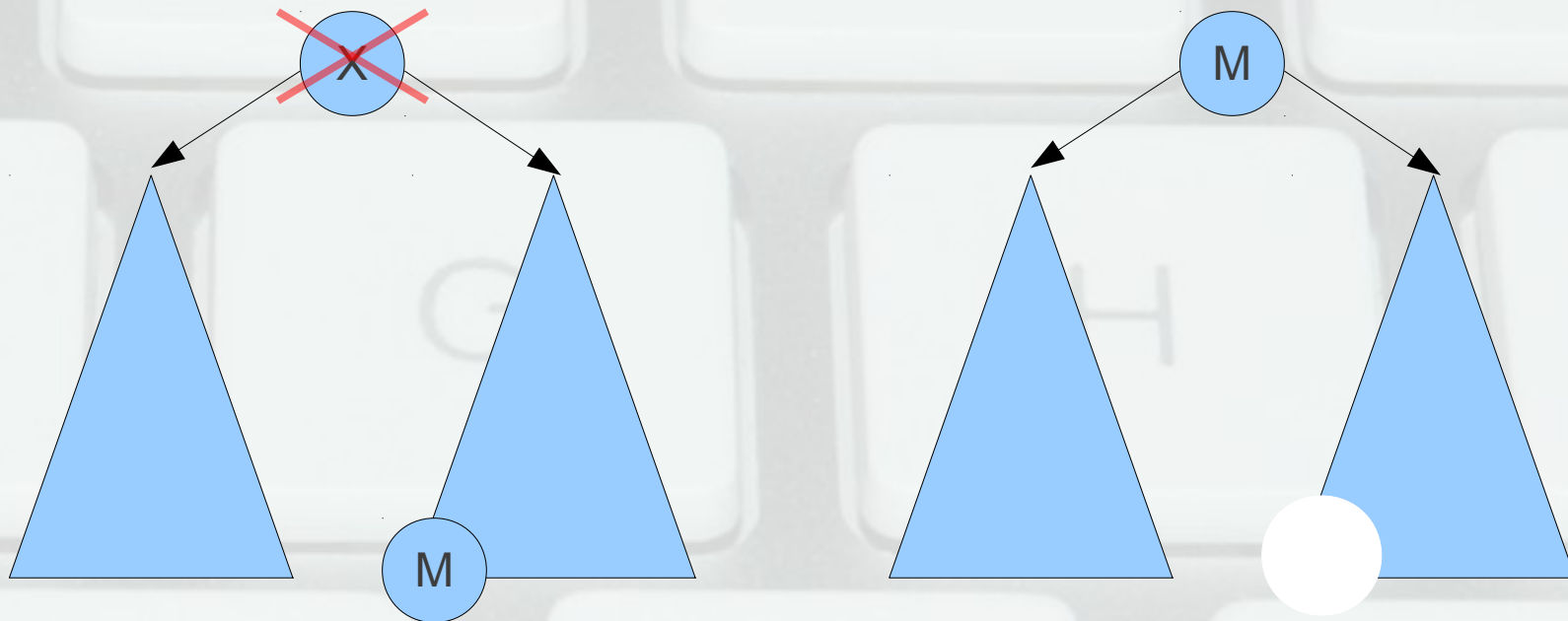
Включване на елемент



Изключване на елемент



Изключване на елемент



Балансирани дървета

- балансирана височина: $|h(L) - h(R)| \leq 1$
- идеално балансирани $|n(L) - n(R)| \leq 1$

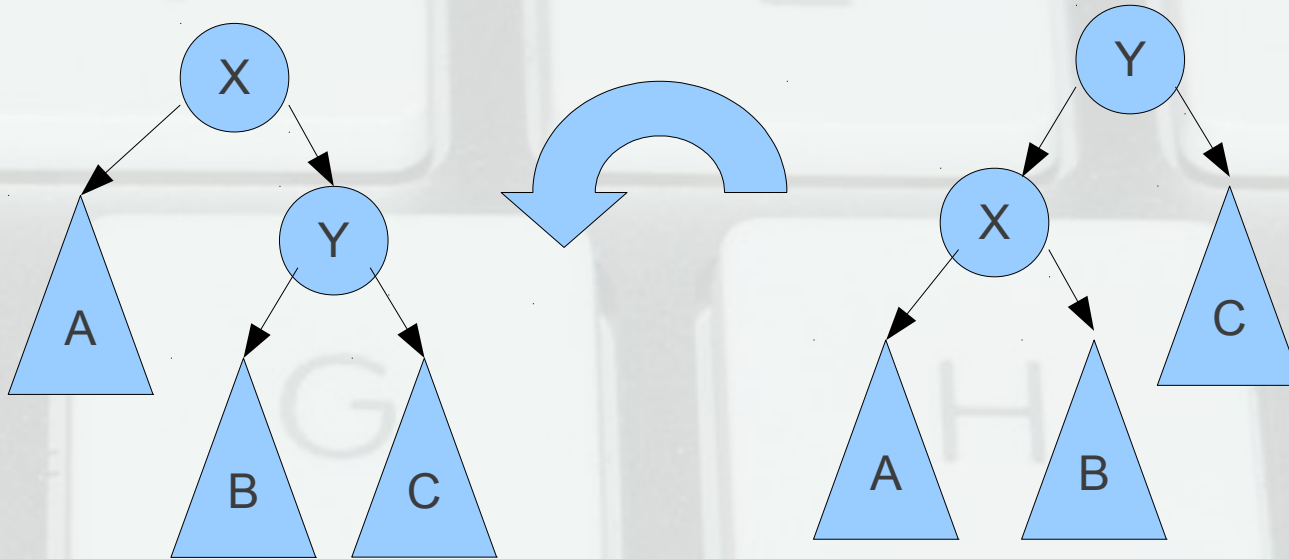
Генериране на иБД

- Нека L е сортиран списък
- $\text{generate}(L) = (\text{mid}(L), \text{generate}(\text{init}(L)), \text{generate}(\text{tail}(L)))$
- $\text{mid}(L)$ = “средният” елемент на L
- $\text{init}(L)$ = първата “половина” на L
- $\text{tail}(L)$ = втората “половина” на L

AVL дървета

- Двоично наредени дървета, които запазват баланса си
- Коефициент на балансиране
 $b(T) = h(R) - h(L)$

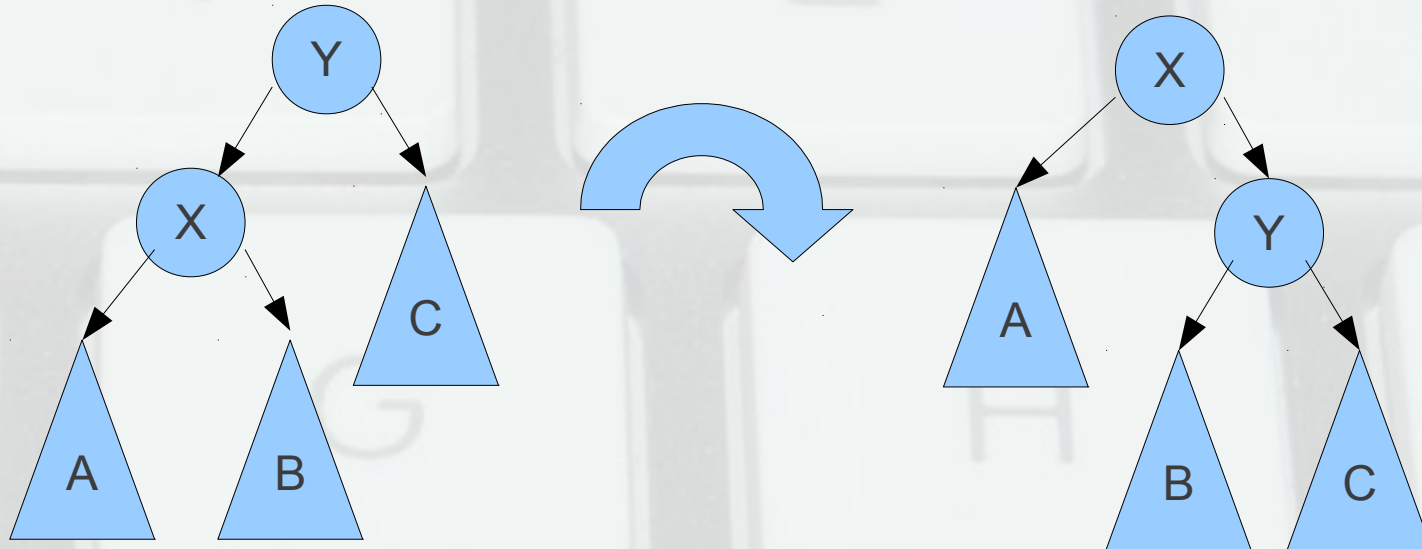
Завъртане наляво



```
if (h(B) >= h(C))  
    b'(X) = b(X) - 1;  
else  
    b'(X) = b(X) - (b(Y) + 1);
```

```
if (h(B) >= h(A))  
    b'(Y) = b(Y) - 1;  
else  
    b'(Y) = b(Y) + (b(X) - 1);
```

Завъртане надясно



if ($h(B) \geq h(C)$)

$b'(X) = b(X) + 1;$

else

$b'(X) = b(X) + (b(Y) + 1);$

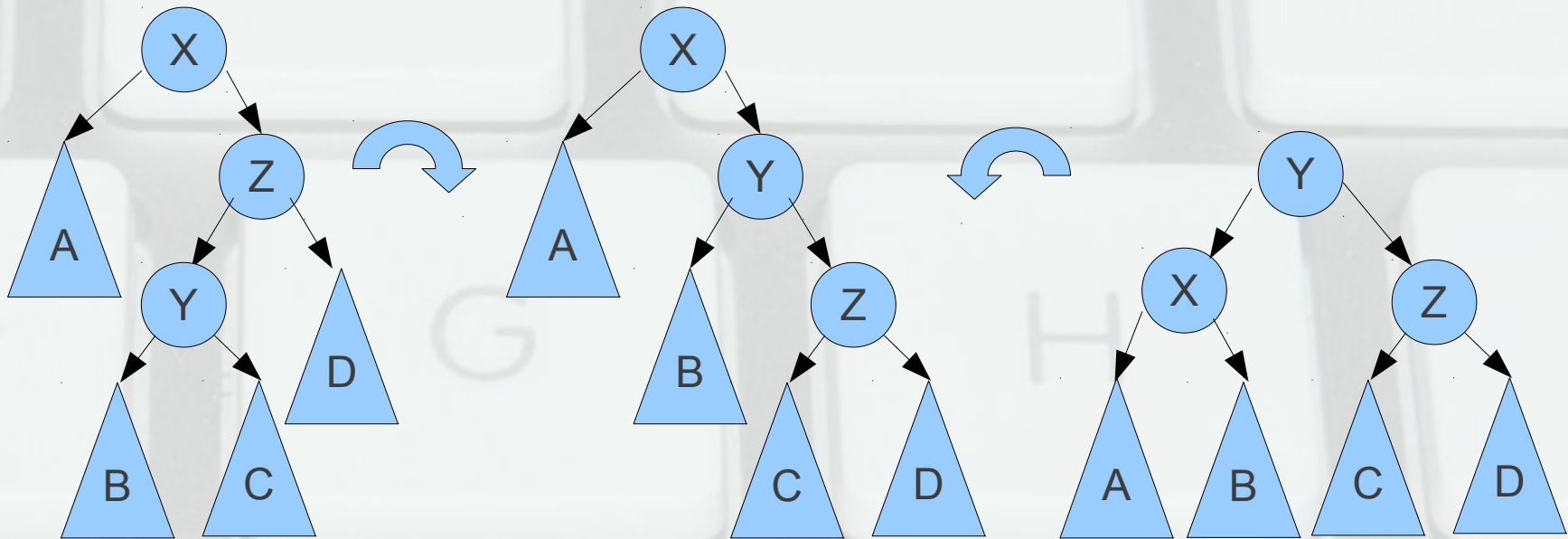
if ($h(B) \geq h(A)$)

$b'(Y) = b(Y) + 1;$

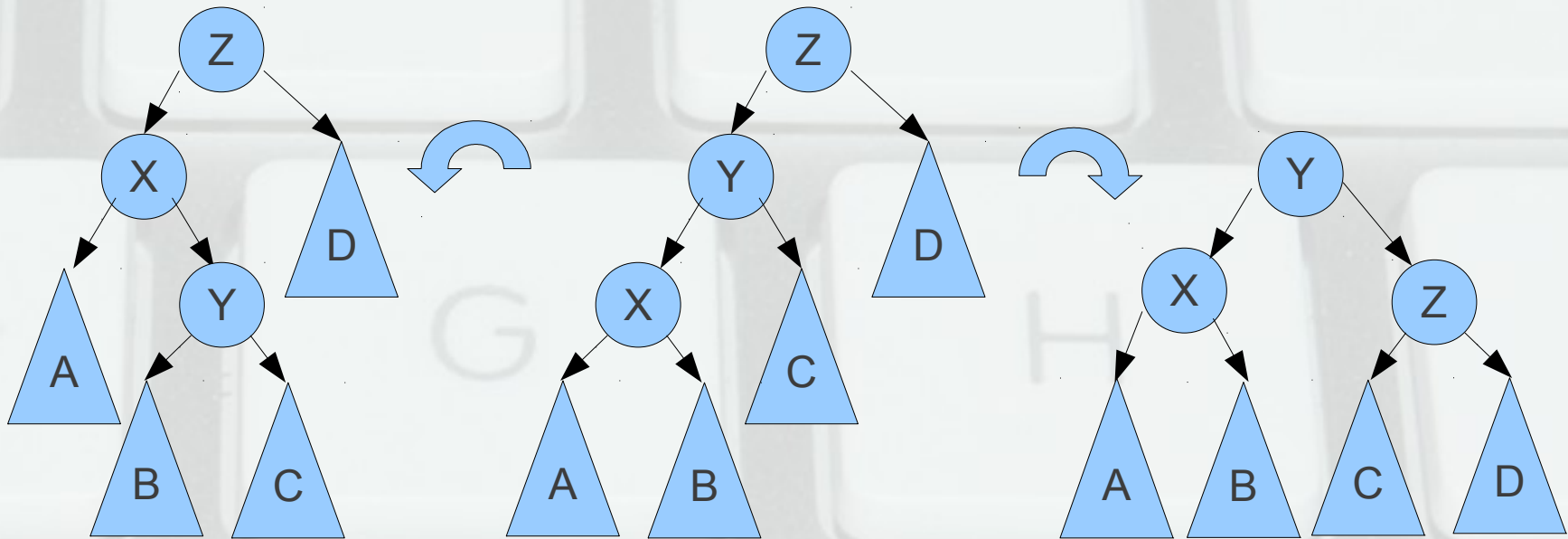
else

$b'(Y) = b(Y) - (b(X) - 1);$

Балансиране



Балансиране

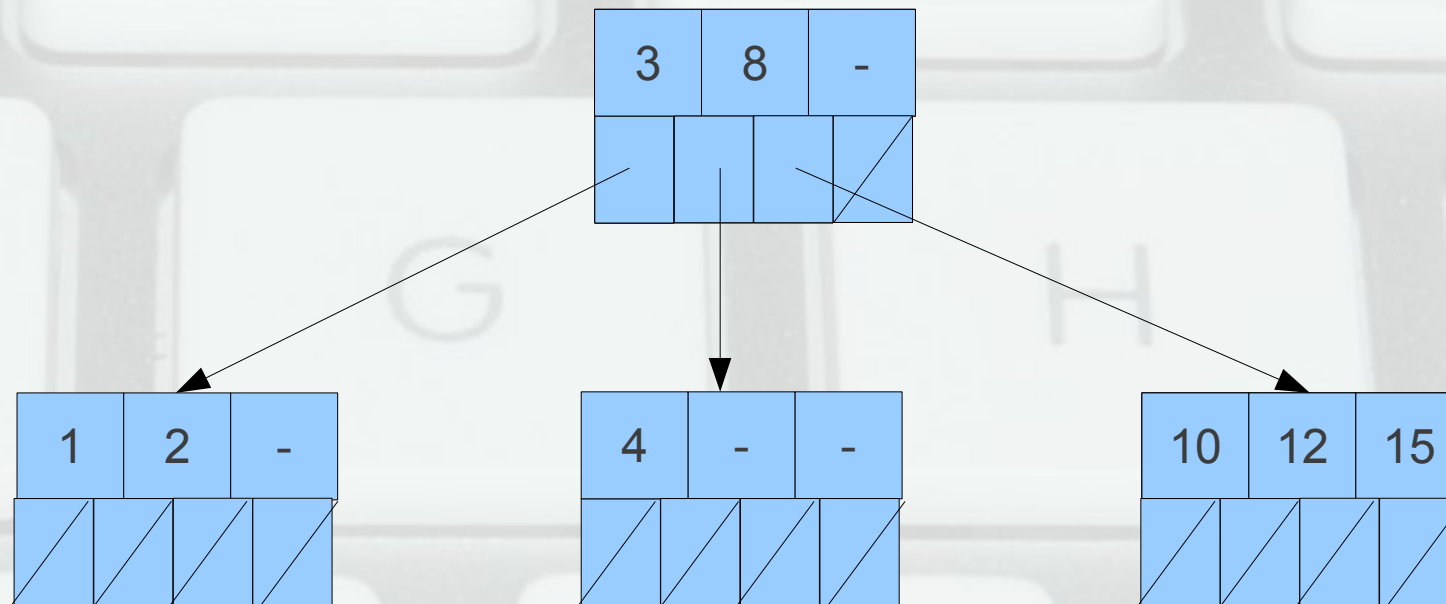


B-дървета

B-дърво от ред n е дърво, за което:

- всички листа са на еднаква височина
- коренът съдържа между 1 и $n-1$ ключа
- другите възли имат между $\left\lceil \frac{n-1}{2} \right\rceil$ и $n-1$ ключа
- всеки възел с m ключа има 0 или $m+1$ деца
- ключовете във всеки възел са строго растящи
- k_i е по-голям от ключовете в T_j за $j \leq i$
- k_i е по-малък от ключовете в T_j за $j > i$

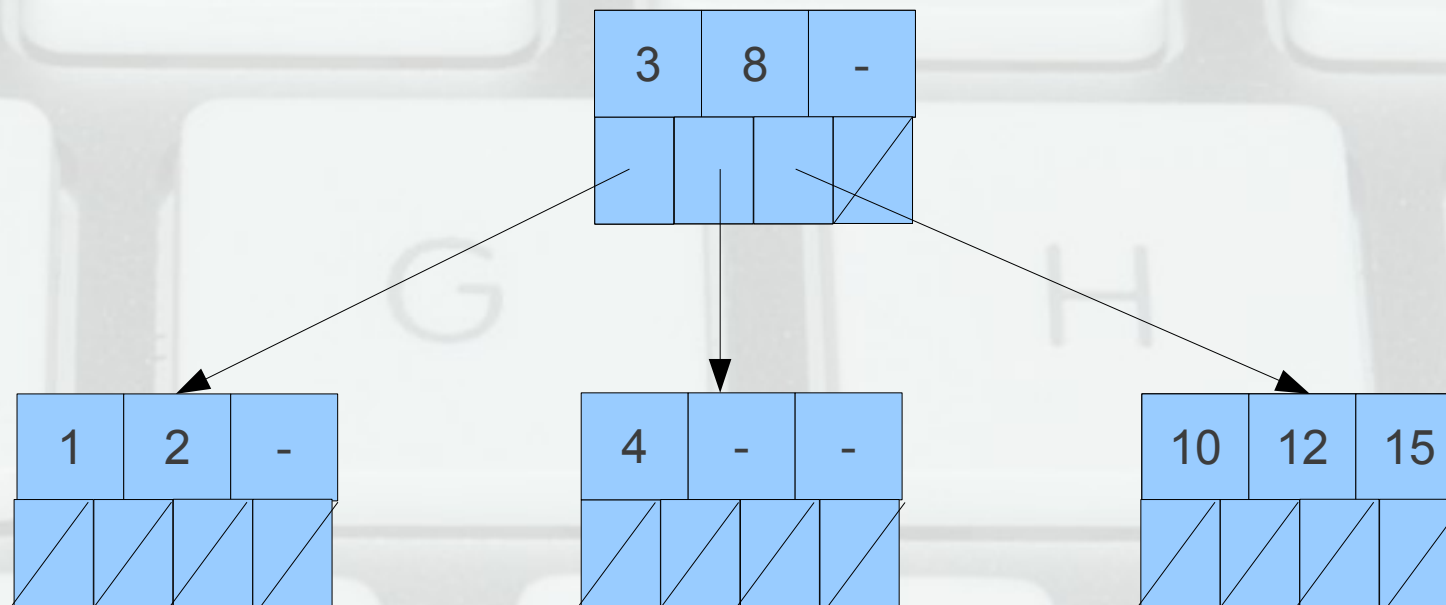
Пример за B-дърво от ред 4



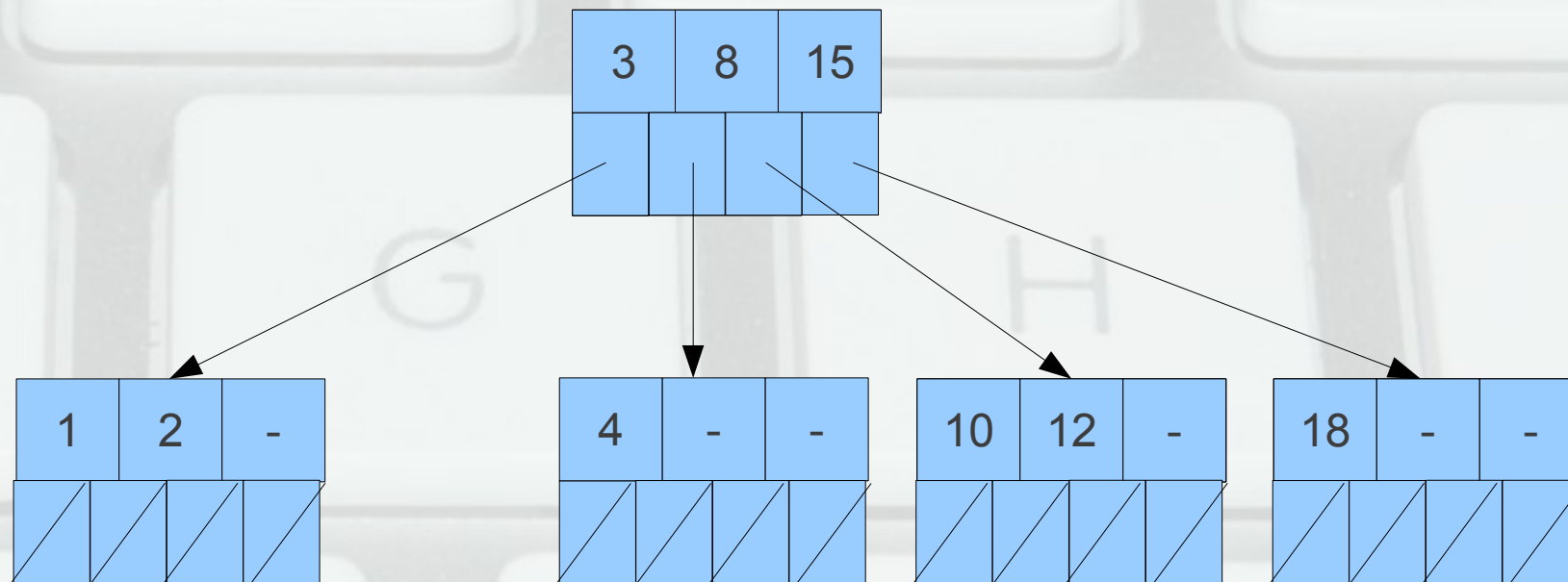
Добавяне на елемент

- Прави се опит да се добави в подходящо листо
- Ако опитваме да вмъкнем във възел, който вече има $n-1$ ключа, тогава:
 - разцепваме възела на два други възела с приблизително равен брой ключове
 - средният по големина ключ вмъкваме в родителя между двата нови възела
 - при нужда, създаваме нов корен

Пример за добавяне на елемент



Пример за добавяне на елемент



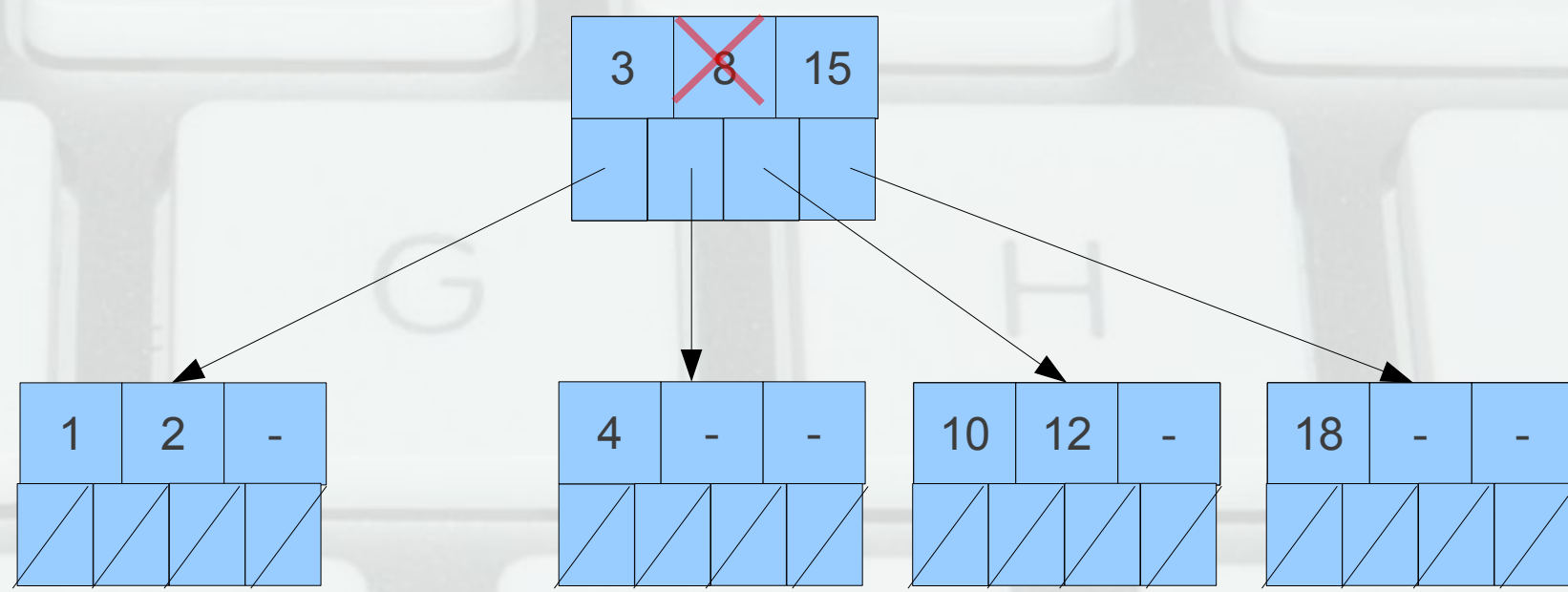
Изключване на елемент

- Намираме ключа K в дървото
- Ако е в листо, изтриваме го
- Ако е във вътрешен възел, заменяме го
 - с най-големия ключ $<K$, или
 - с най-малкия ключ $>K$

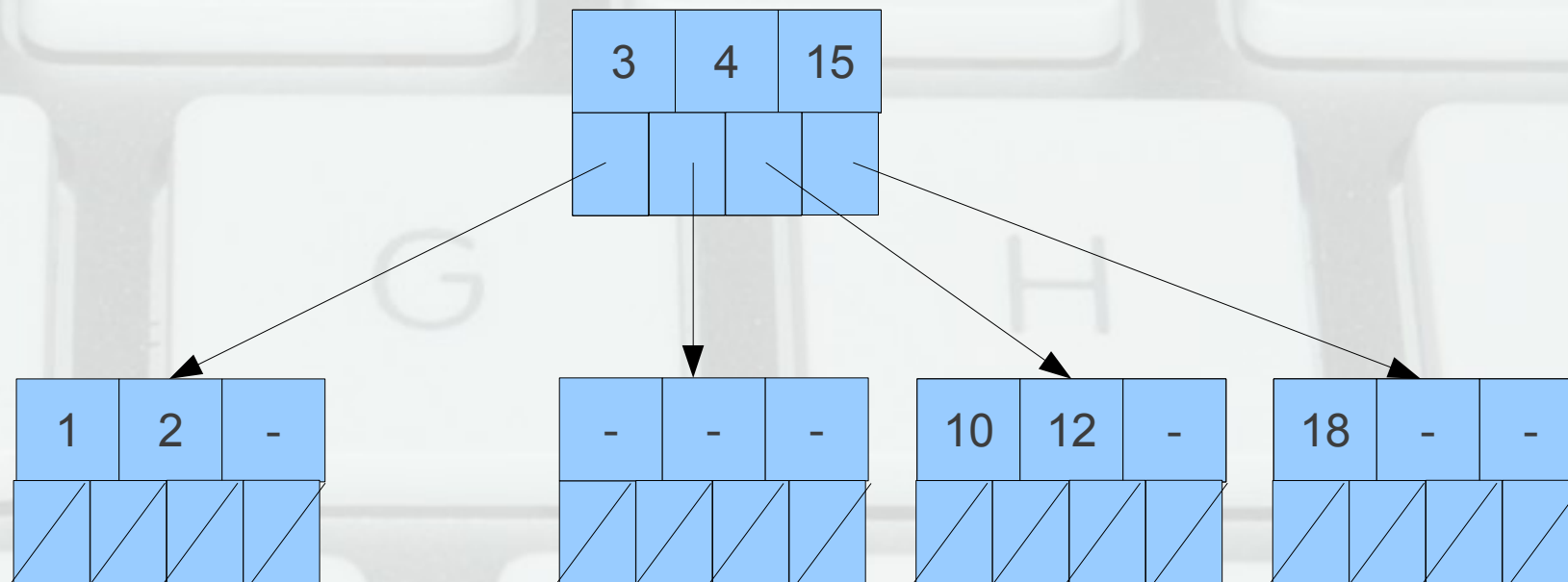
Изключване на елемент

- Ако броят ключове падне под минимума:
 - опит да се заеме ключ и поддърво от съсед с ключове над минимума
 - ако и двата съседа вече имат минимум ключове, се извършва сливане на възела с неговия съсед
 - в новия възел се добавя и ключа в родителя, който е разделял двата слети ключа
 - алгоритъмът се повтаря за родителя

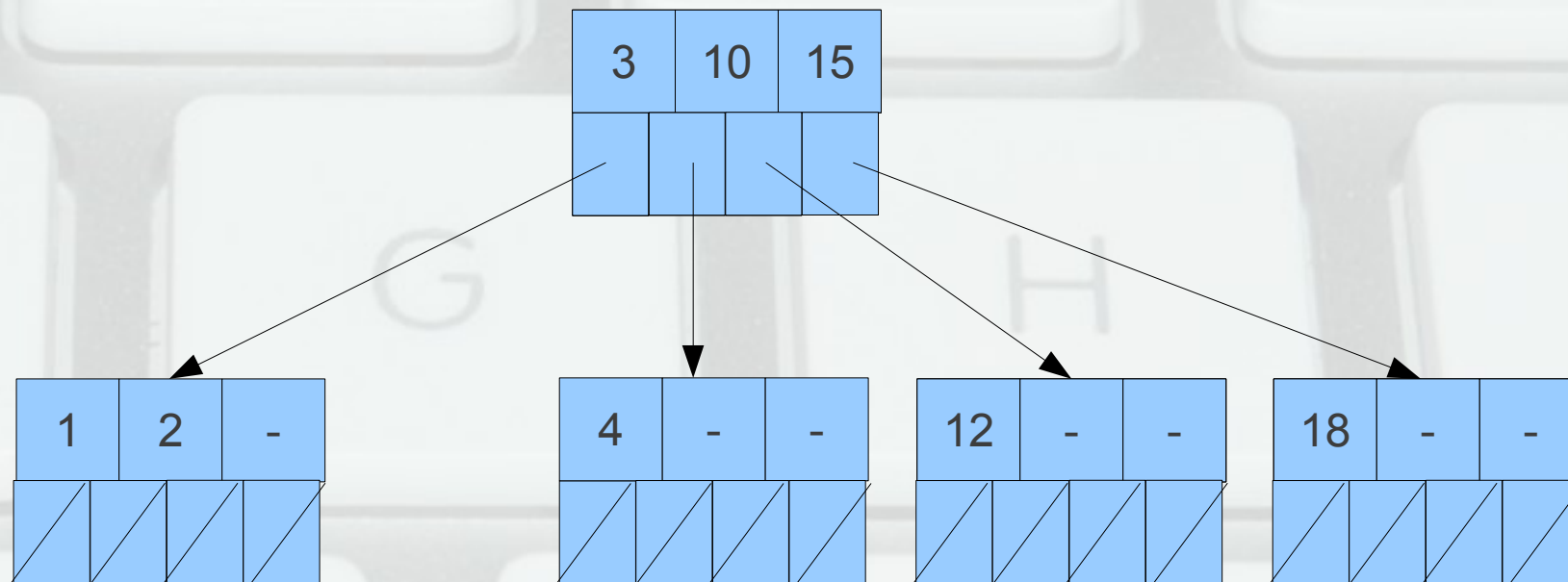
Примери за изтриване на елемент



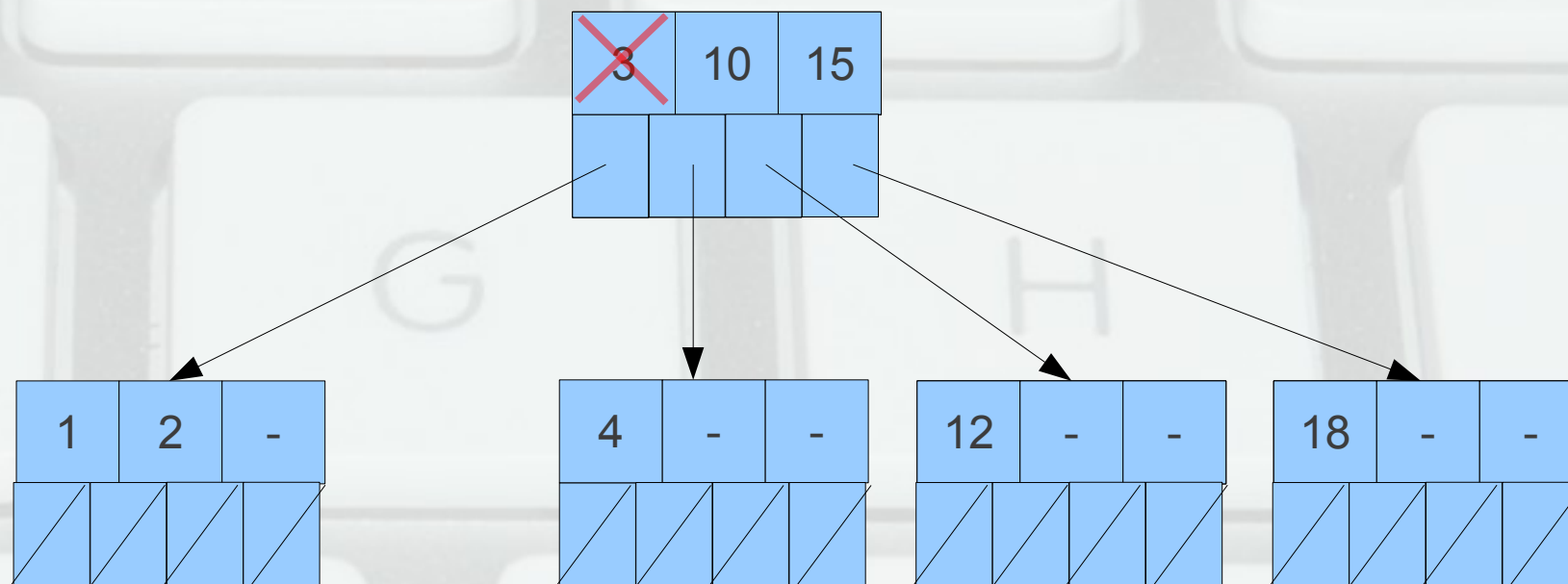
Примери за изтриване на елемент



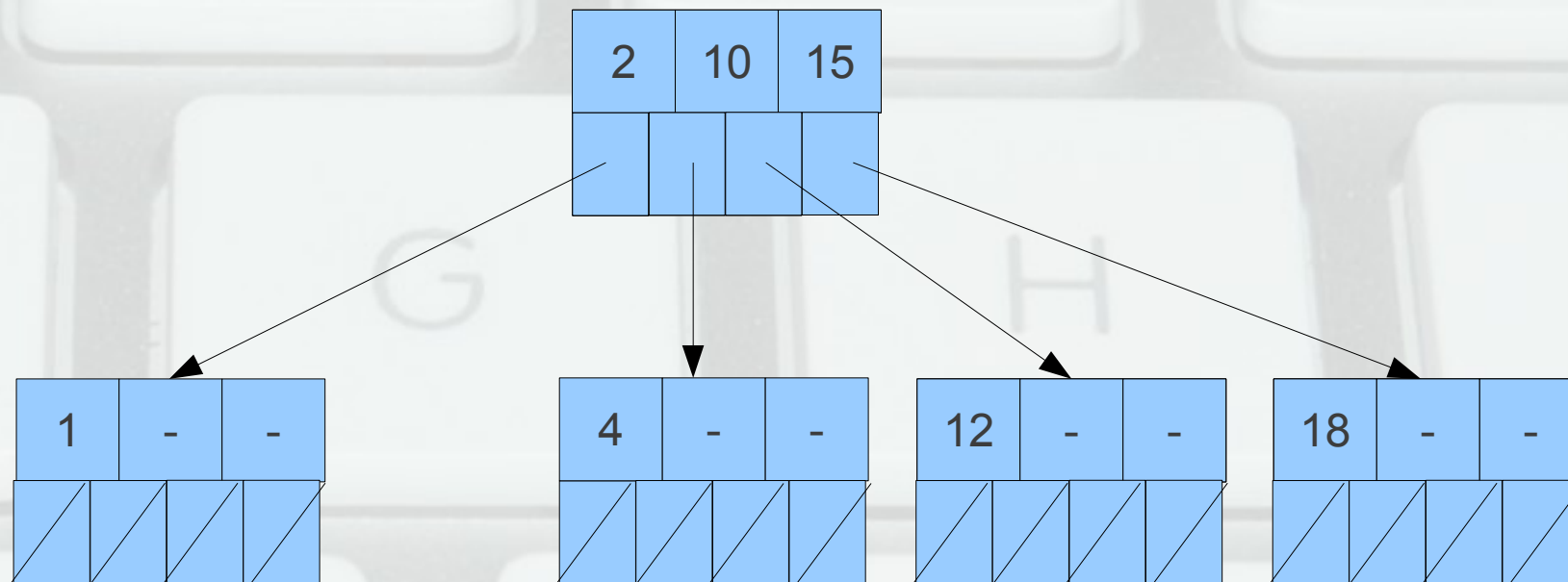
Примери за изтриване на елемент



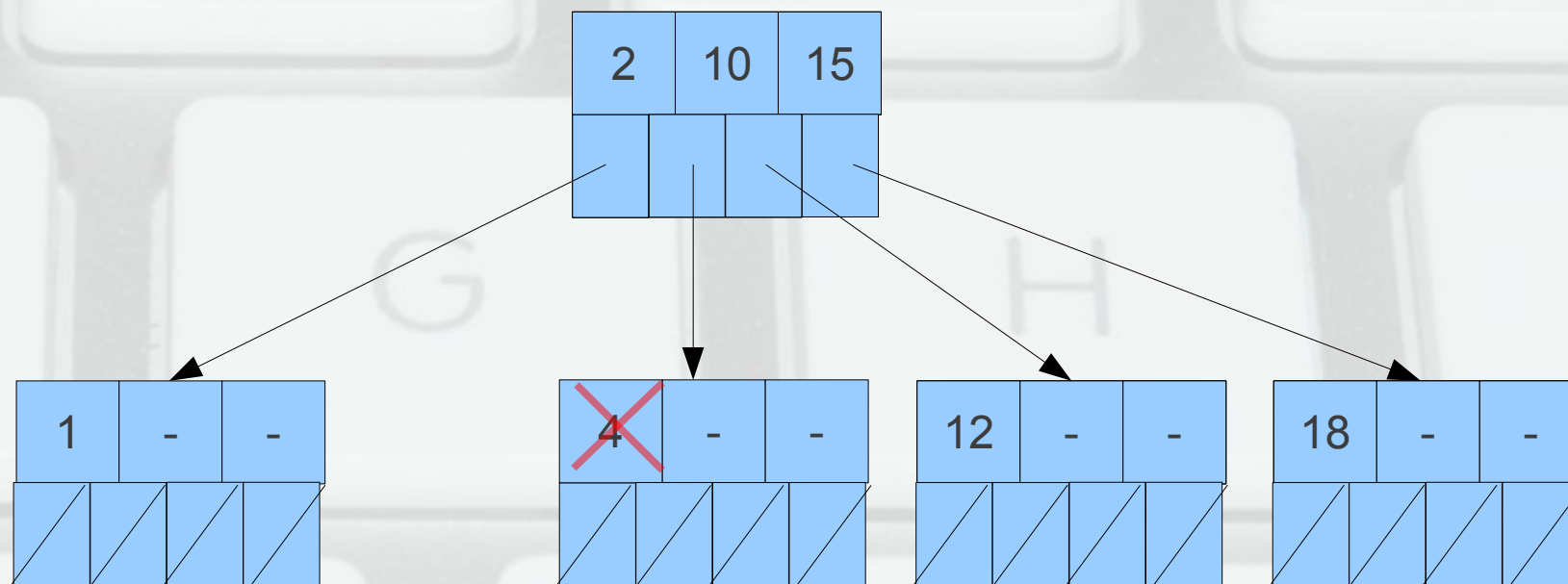
Примери за изтриване на елемент



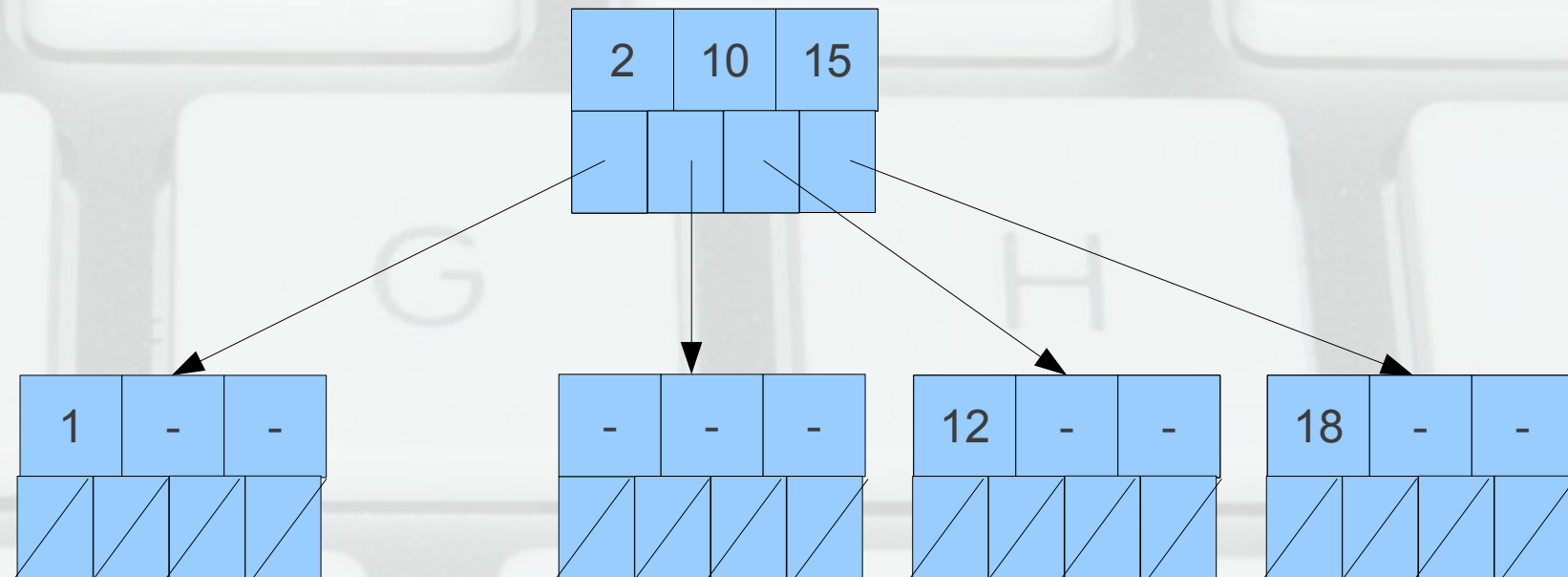
Примери за изтриване на елемент



Примери за изтриване на елемент



Примери за изтриване на елемент



Примери за изтриване на елемент

