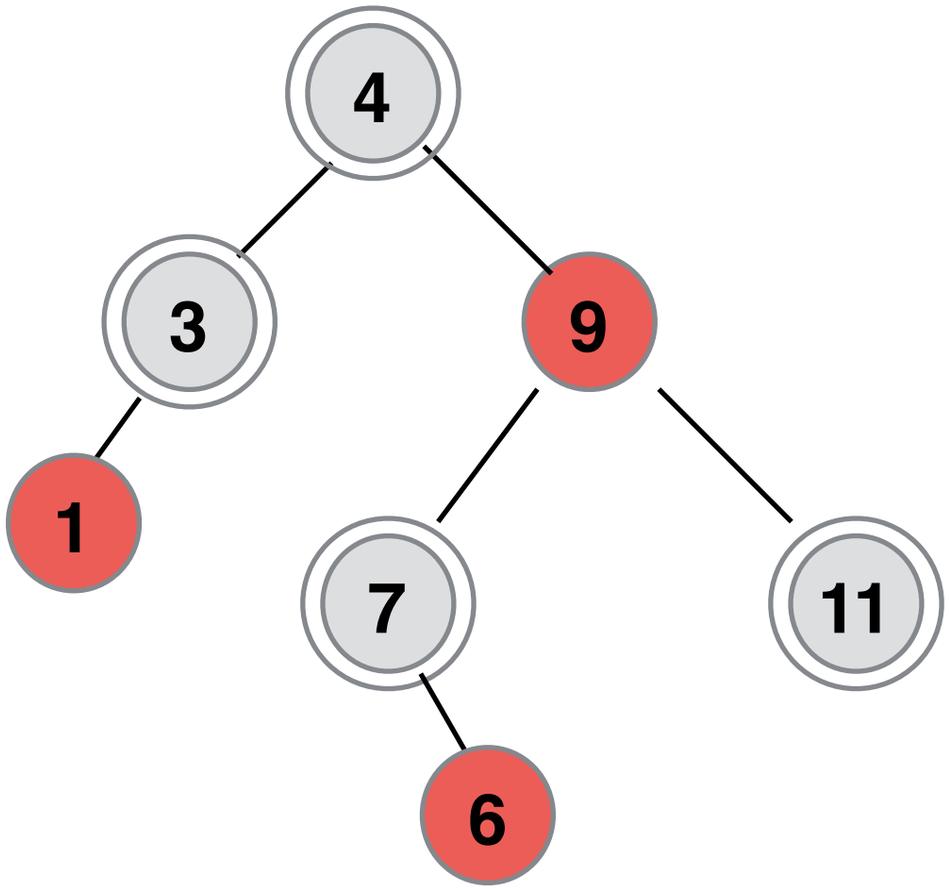
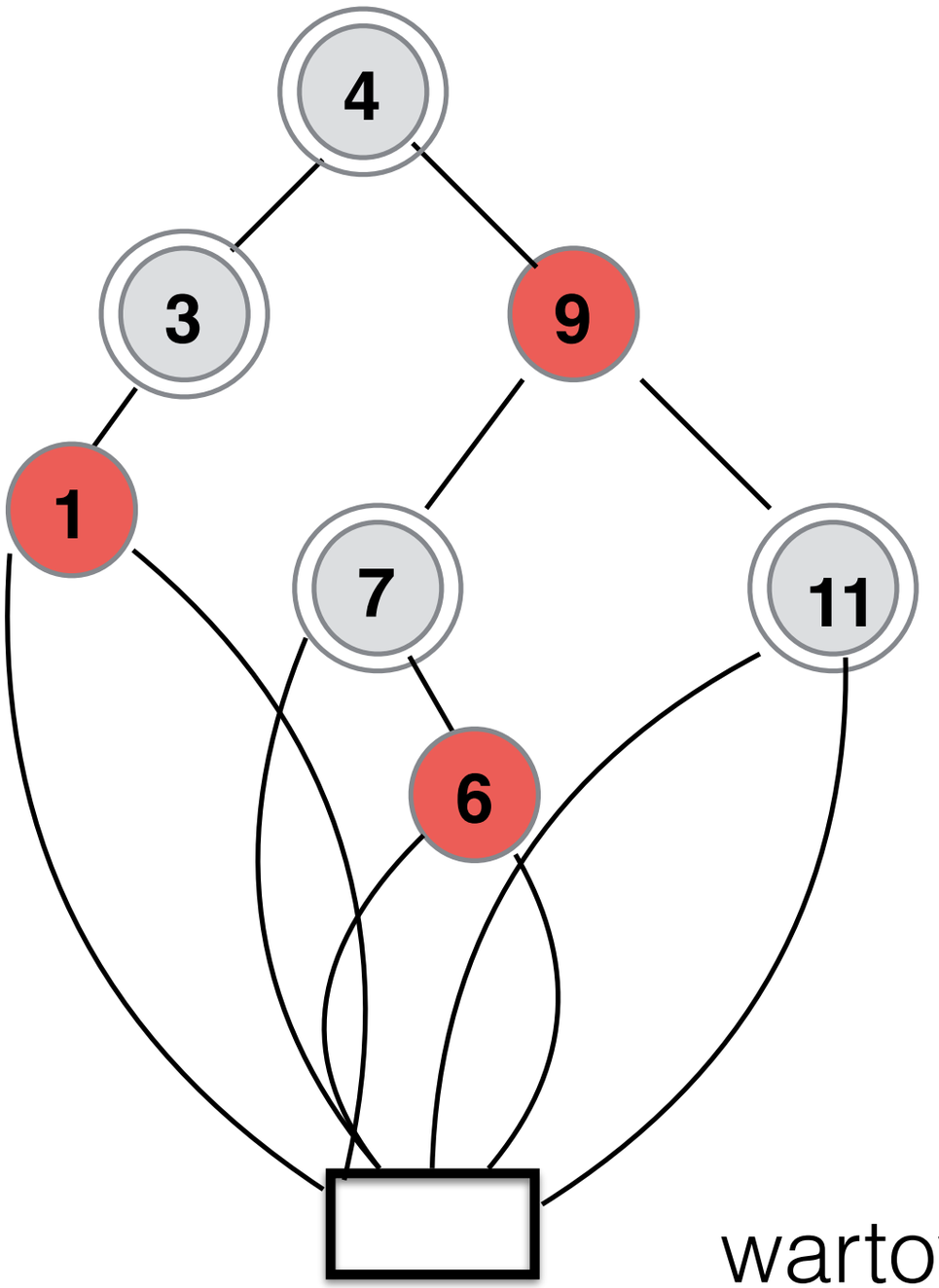
Drzewa czerwono-czarne

Definicja *Drzewo czerwono-czarne* to drzewo poszukiwań binarnych, które spełnia następujące warunki

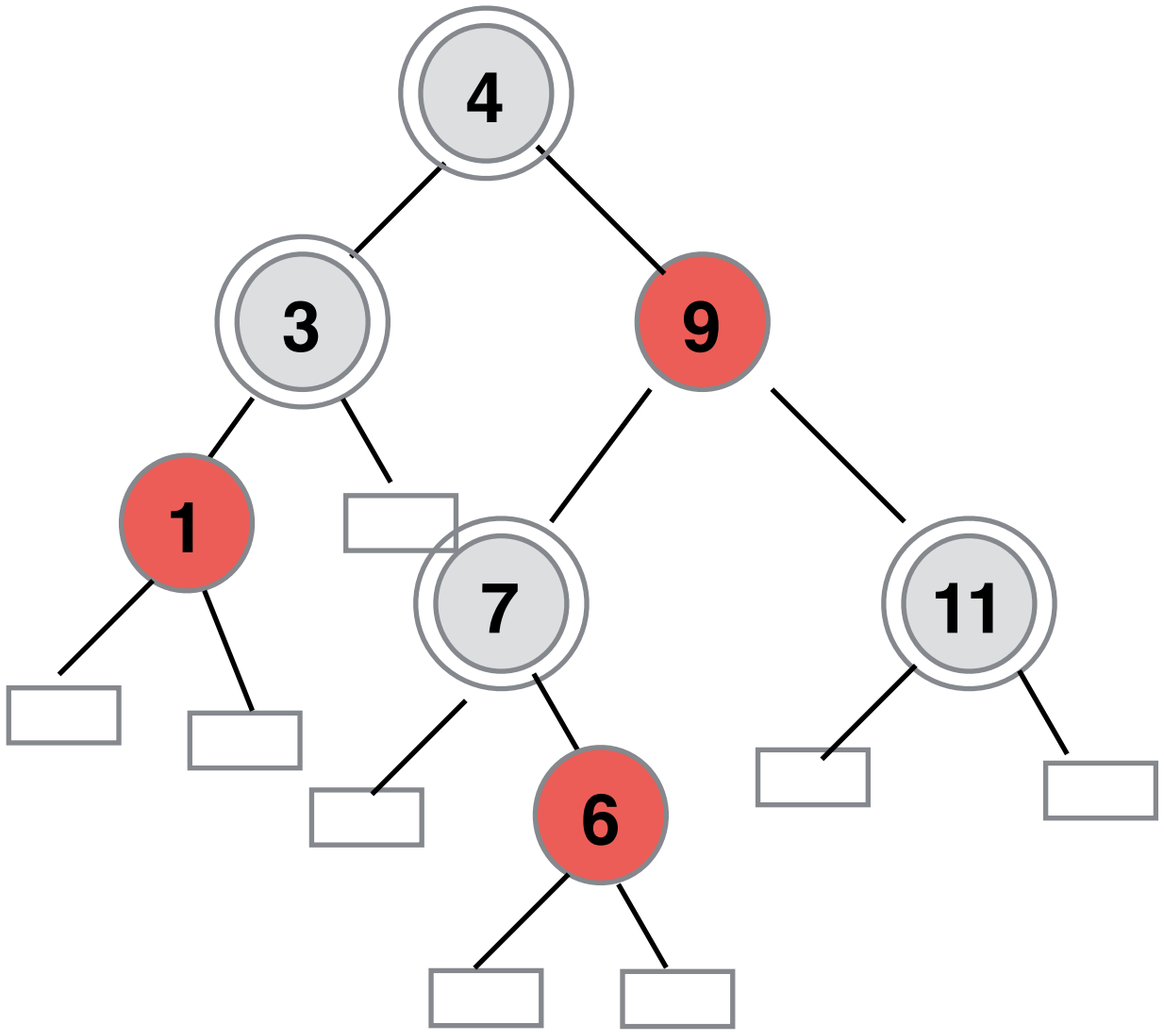
1. każdy węzeł mam przypisany kolor: czerwony lub czarny
2. korzeń jest czarny
3. liście są czarne; przyjmujemy wariant drzewa z wartownikami - liśćmi są wartownicy
4. czerwony węzeł nie może mieć czerwonego syna
5. na każdej ścieżce od korzenia do liści jest tyle samo czarnych węzłów

Przykład



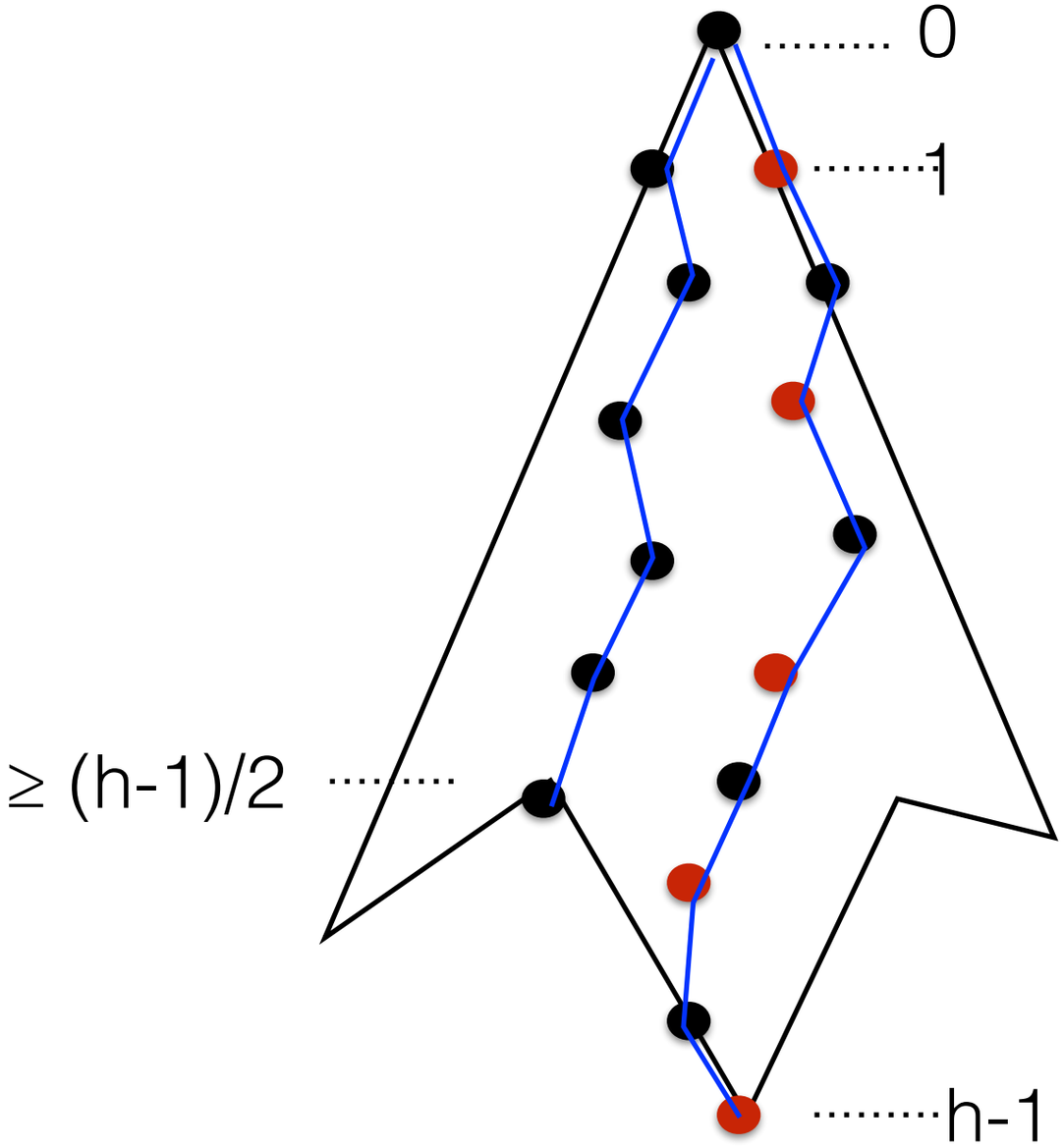


wartownik



wartownik

Stwierdzenie Drzewo czerwono-czarne posiadające *n* węzłów wewnętrznych ma wysokość nie większą niż $2 \cdot lg(n+1)$

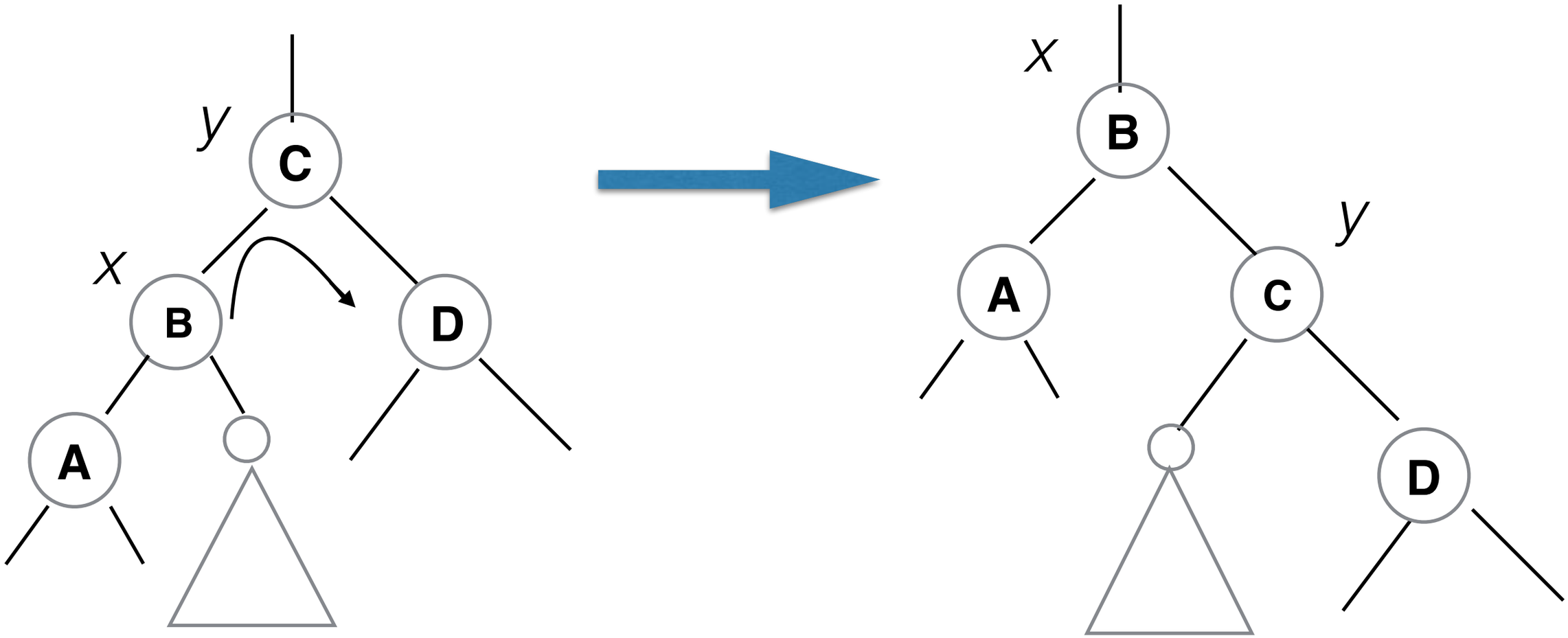


$$n \geq 2^{(h-1)/2+1} - 1 \geq 2^{h/2} - 1$$

$$lg(n+1) \geq h/2$$

$$2\, lg(n+1) \geq h$$

Rotacja w prawo wokół wokół węzła *y*



Rotacja w lewo — symetrycznie

Własność: rotacje zachowują uporządkowanie drzewa poszukiwań binarnych

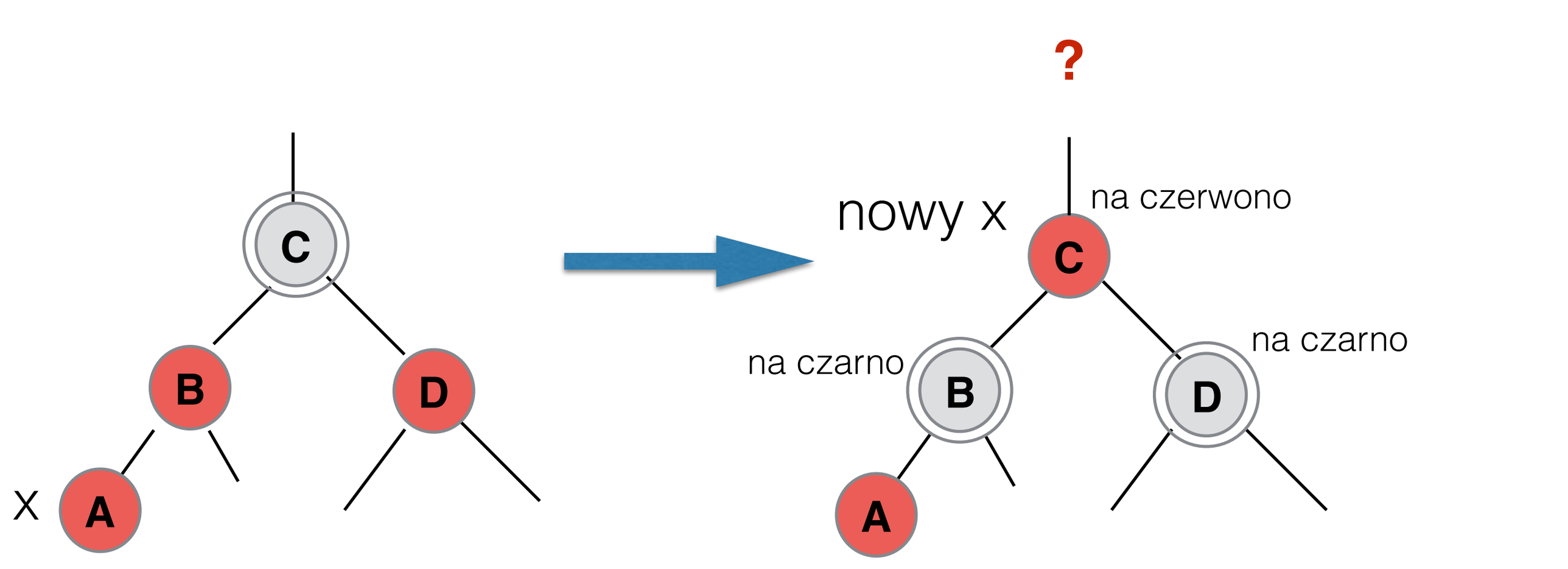
Drzewa czerwono-czarne
wstawianie

- wstawiamy węzeł jak do drzewa poszukiwań binarnych,
- kolorujemy go na czerwono
- i poprawiamy drzewo, jeżeli kolor jego ojca też jest czerwony

x oznacza czerwony węzeł, którego ojciec może być czerwony.
Jeżeli tak jest, dopasowujemy któryś z poniższych przypadków.

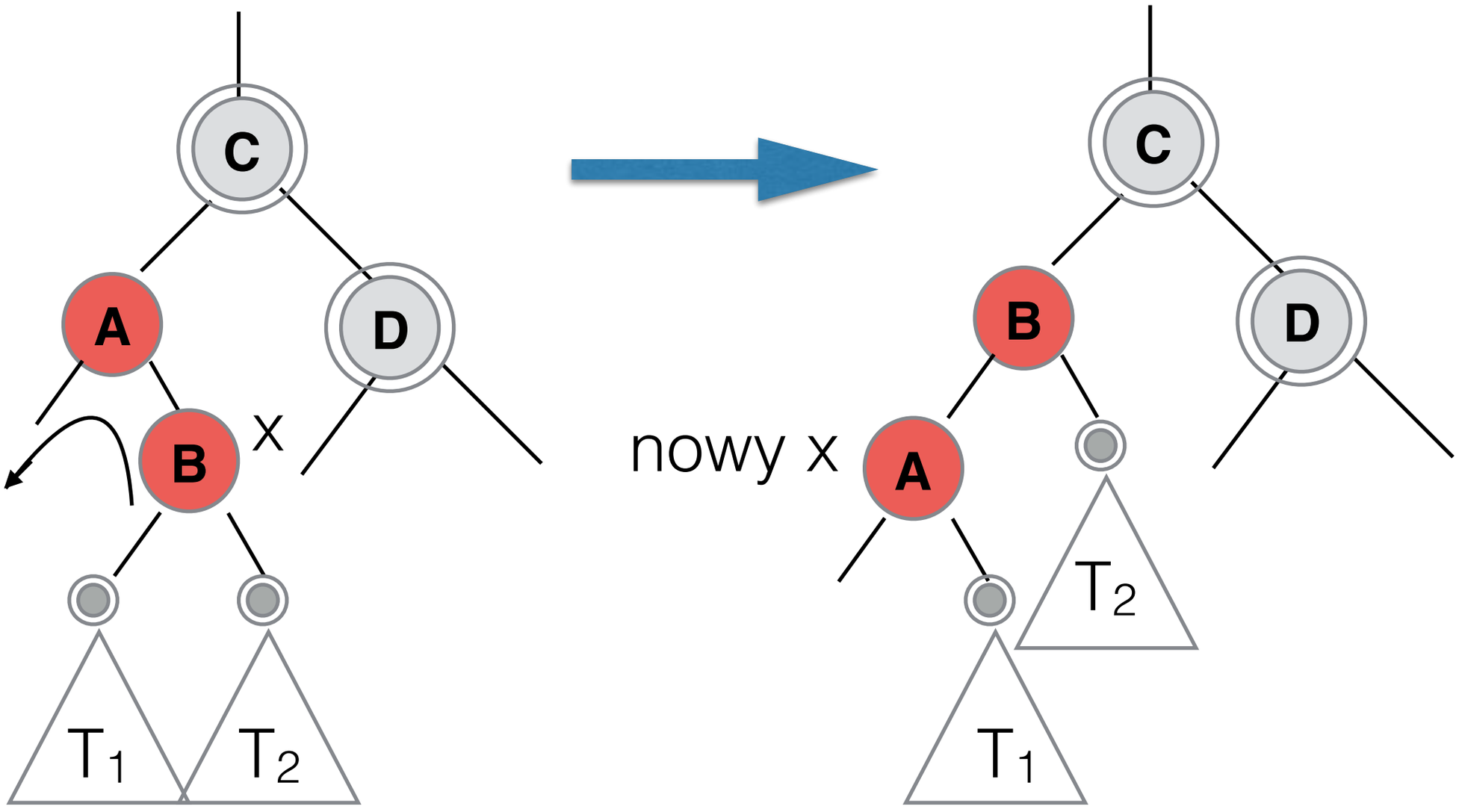
Uwaga: czarne węzły na poniższych rysunkach mogą być wartownikami

Rysunki poniżej przedstawiają “lewe” przypadki: *ojciec x jest lewym synem*. “Prawe” przypadki są symetryczne



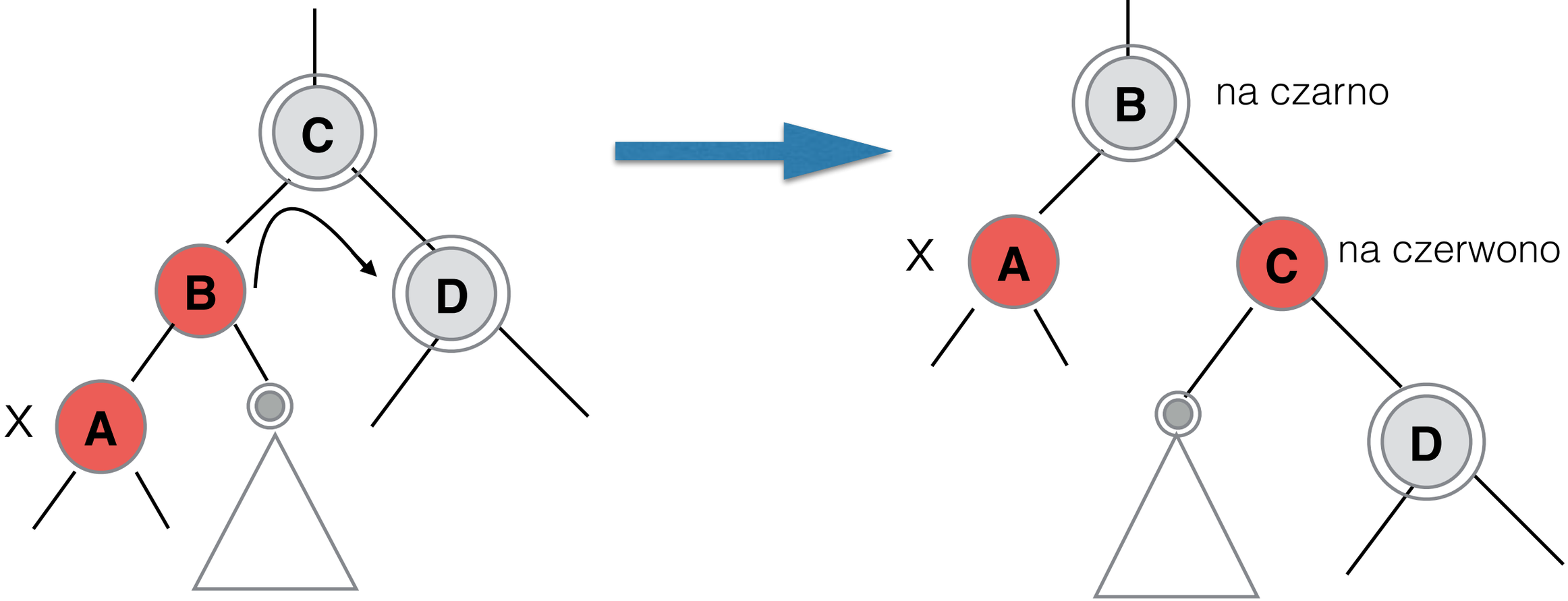
Przypadek 1: brat ojca x czerwony.
(x może być lewym - jak na rysunku - albo prawym synem swojego ojca)

Zmieniamy kolorowanie, x przenosi się o 2 poziomy wyżej, kontynuujemy poprawianie jeżeli ojciec nowego x jest czerwony



Przypadek 2: brat ojca węzła x jest czarny, węzeł x i jego ojciec leżą w różnych kierunkach tworząc zakręt.

Robimy rotację “wyprostowującą” zakręt, przesuwamy x i w ten sposób doprowadzamy do przypadku 3.



Przypadek 3: brat ojca węzła x jest czarny, węzeł x i jego ojciec leżą w ty
samym kierunku.

Robimy rotację wokół ojca x i zmiany kolorów. Drzewo jest naprawione.

```
RB-insert(T,x)
  BST-insert(T,x) // wersja z wartownikiem
  x.color=RED
  while x!=T.root and x.p.color==RED
      if x.p == x.p.p.left  // x.p jest lewym synem
            y = x.p.p.right  // y = brat ojca
            if y.color == RED    // przypadek 1
               x.p.color = BLACK
               y.color = BLACK
               x.p.p.color = RED
               x = x.p.p    // nowy x
         else if x == x.p.right   // przypadek 2
                     x = x.p
                     Left-rotate(T,x)
               x.p.color = BLACK  // przypadek 3
               x.p.p.color = RED
               Right-rotate(T, x.p.p)
else
     // symetrycznie
     // . . . . . .
T.root.color = BLACK
```

Drzewa czerwono-czarne
usuwanie

- usuwamy węzeł *z* jak w drzewie poszukiwań binarnych; niech y oznacza ten węzeł, który został naprawdę usunięty (jeżeli *z* miał dwóch synów, to y jest różny od *z*) i niech x oznacza węzeł, który wszedł w miejsce y (może to być wartownik)

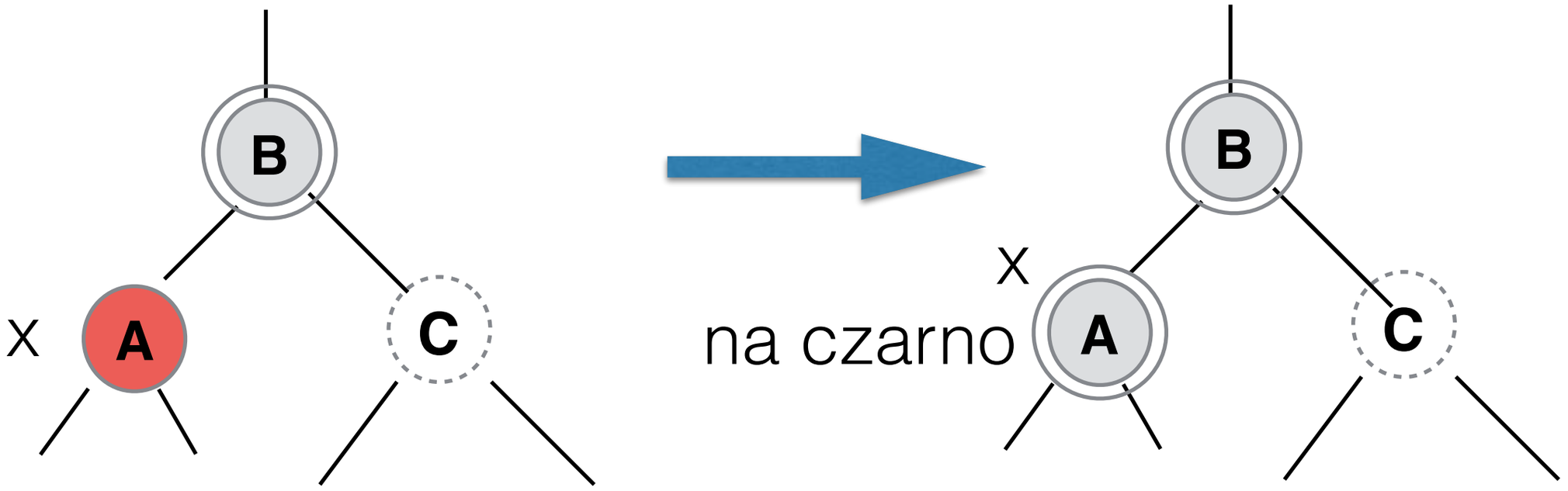
- jeżeli y był czarny, to na ścieżkach od korzenia drzewa do liści przechodzących przez x jest o 1 czarny węzeł mniej niż na innych ścieżkach. W tej sytuacji poprawiamy drzewo.

Drzewa czerwono-czarne
usuwanie

x oznacza węzeł, od którego zaczynając brakuje jednego czarnego węzła na ścieżkach w dół drzewa

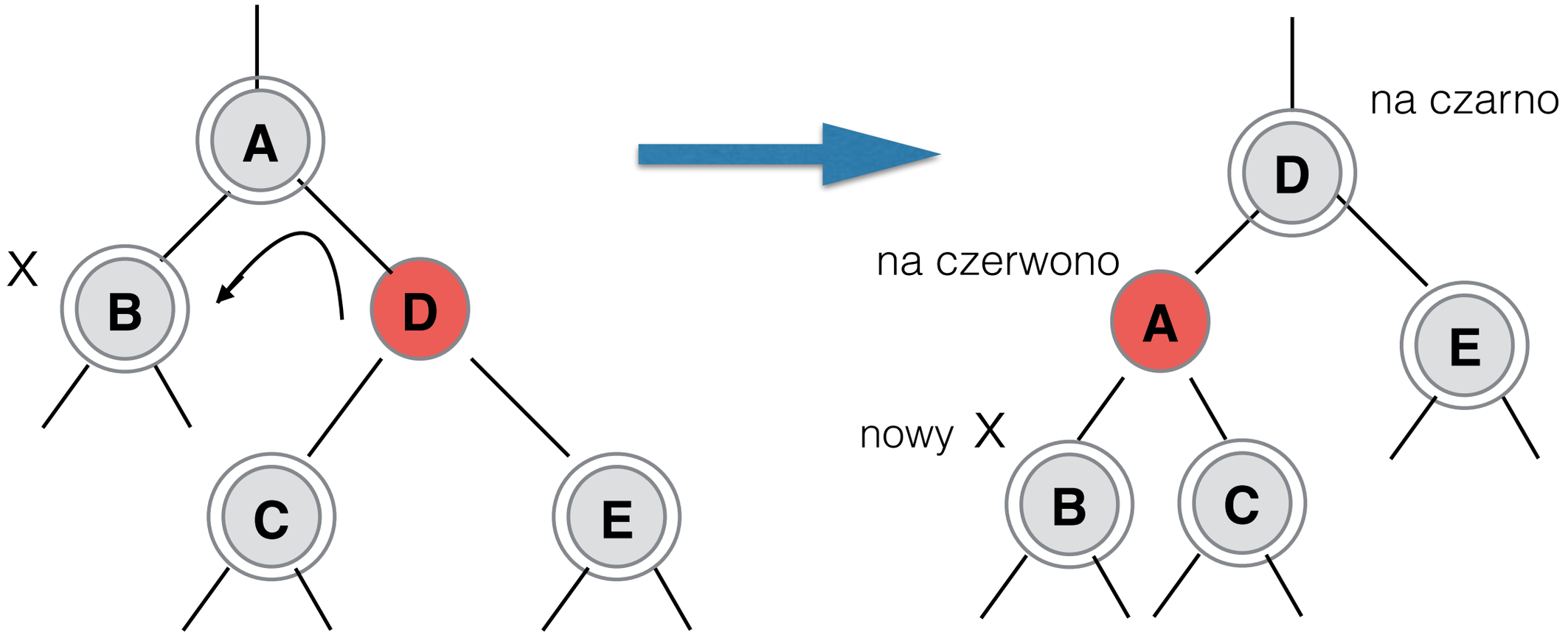
zarówno x jak i inne czarne węzły mogą być wartownikami

Poniżej są narysowane “lewe” przypadki, to znaczy x jest lewym synem. “Prawe” przypadki są symetryczne.



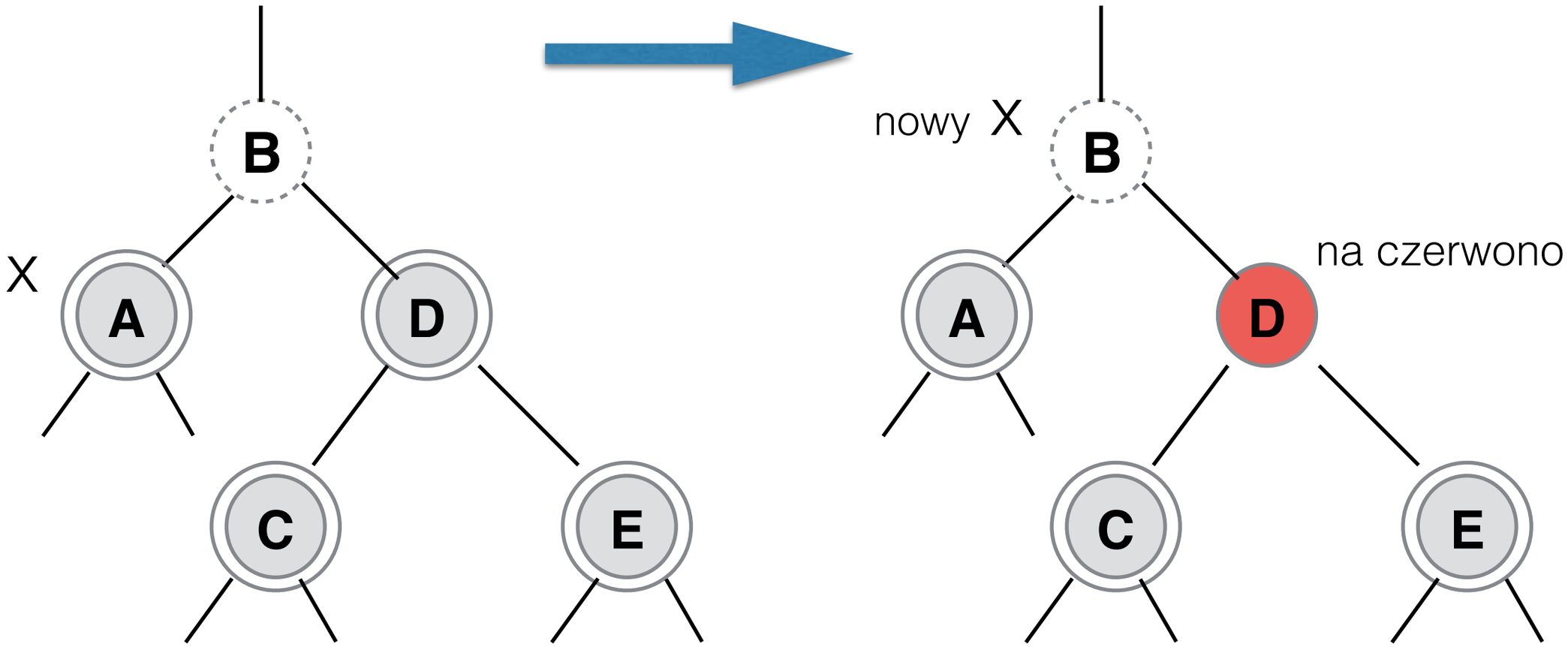
Przypadek 0: węzeł x jest czerwony

Zmieniamy kolor x i drzewo jest naprawione



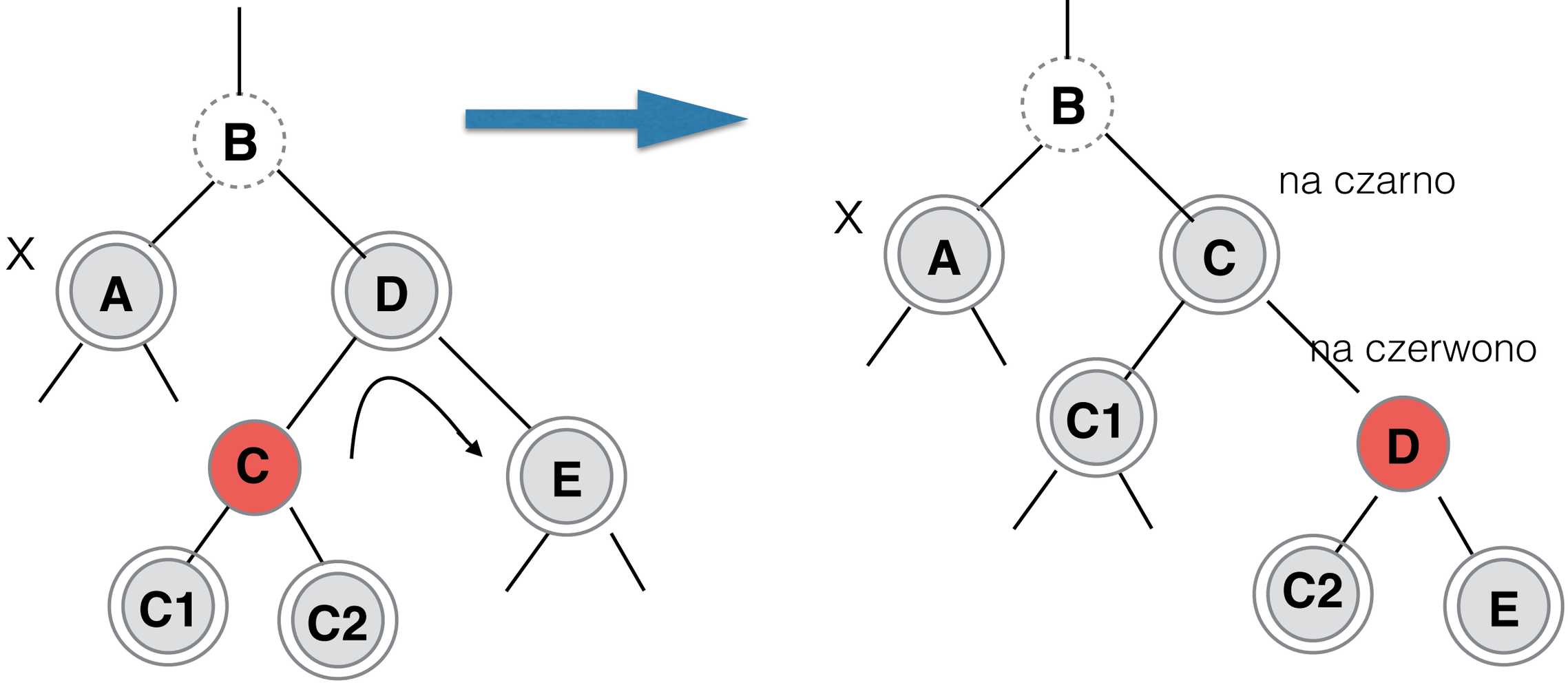
Przypadek 1: brat x czerwony.

Rotacja w stronę x wokół jego ojca i zmiany kolorów. Ojciec nowego x jest czarny, drzewo nie jest naprawione więc kontynuujemy naprawianie drzewa ale będzie już jakiś inny przypadek



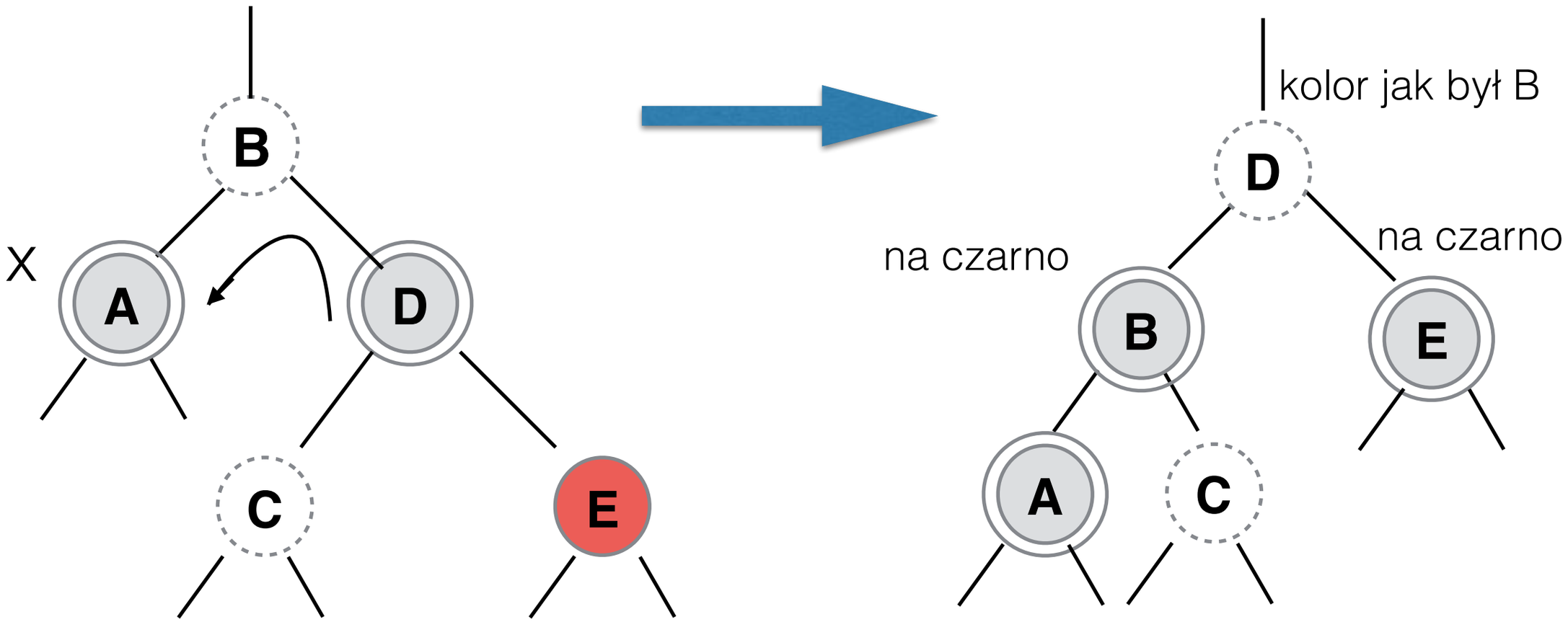
Przypadek 2: brat węzła x i obydwaj synowie brata czarni.

Robimy zmiany kolorów i przesuwamy x o 1 poziom wyżej, kontynuuj
naprawianie drzewa



Przypadek 3: brat x czarny, syn brata skierowany tak jak x jest czerwony a drugi syn czarny.

Doprowadzamy do przypadku 4 poprzez rotację wokół brata x w stronę przeciwną do x i zmieniając kolory



Przypadek 4: brat x czarny, syn brata skierowany przeciwnie niż x jest czerwony, drugi syn czerwony lub czarny

Robimy rotację w stronę x wokół jego ojca i zmieniamy kolory. Drzewo jest naprawione.

Stwierdzenie Przedstawione algorytmy wstawiania i usuwania węzła w drzewie czerwono-czarnym są poprawne, tzn. po wykonaniu każdej z tych operacji drzewo nadal spełnia warunki wymagane od drzewa czerwono-czarnego.

Uzasadnienie. Można sprawdzić, że wszystkie przypadki poprawiania

- albo naprawiają drzewo,
- albo nie pogarszają sytuacji, tzn. zaburzenie, które wnosił węzeł x jest niezmienione, pomimo zmiany struktury drzewa, kolorowania i ewentualnego przesunięcia x

Dodatkowo w przypadku usuwania sprawdzamy, że algorytm się nie zapętla.

Stwierdzenie Operacje wstawiania, szukania i usuwania wykonywane na drzewie czerwono-czarnym mają czas pesymistyczny $\Theta(lg\ n)$.

Dowód

- Drzewo ma wysokość $\Theta(lg\ n)$.
- Algorytmy te polegają na przejściu ścieżką w drzewie maksymalnie od korzenia do liścia i z powrotem.
- W każdym odwiedzonym węźle ilość operacji jest $\Theta(1)$.
-