Drzewa zbalansowane:

Drzewa Czerwono-Czarne

IIUWr. II rok informatyki. Przygotował: Krzysztof Loryś

15 Drzewa zbalansowane -wstęp

Najpoważniejszą wadą binarnych drzew przeszukiwań jest brak zabezpieczenia przed nierównomiernym rozrastaniem się, przez co pesymistyczny czas wykonywania operacji na nich może być liniowy względem liczby wierzchołków. Proste sposoby zaradzenia temu zjawisku mogą być niepraktyczne. Idealnym rozwiązaniem byłoby utrzymywanie drzew w stanie dokładnego zbalansowania, tj. tak, by wszystkie liście leżały na co najwyżej dwóch poziomach. Niestety przywracanie struktury takiego drzewa po zaburzeniu jej operacjami insert czy delete jest niezwykle kosztowne.

Zadanie: Skonstruuj dokładnie zbalansowane drzewo T o n wierzchołkach i znajd"x element x taki, że po dopisaniu x do T i zbalansowaniu powstałego drzewa, $\Omega(n)$ elementów T zmieni swoje pozycje.

Innym, niestety także zbyt kosztownym, rozwiązaniem byłoby pamiętanie sumy długości wszystkich ścieżek od korzenia do wierzchołków drzewa i przeorganizowywanie drzewa dopiero gdy suma ta przekroczy jakąś graniczną wartość (np. $2n \log n$).
Znanych jest wiele różnych odmian drzew zbalansowanych. My poznamy drzewa czerwono-czarne, drzewa AVL oraz B-drzewa. Wspomnimy także o drzewach samoorganizujących się oraz tzw. treap'ach.

16 Drzewa czerwono-czarne

16.1 Definicja

Definicja 1 *Binarne drzewo przeszukiwań jest* drzewem czerwono-czarnym *jeśli spełnia następujące warunki:*

- *w1. Każdy wierzchołek jest czerwony lub czarny.*
- *w2. Każdy liść jest czarny.*
- *w3. Jeśli wierzchołek jest czerwony, to jego obaj synowie są czarni.*
- *w4. Na każdej ścieżce prowadzącej z danego wierzchołka do liścia jest jednakowa liczba czarnych wierzchołków.*

Konwencja: Dla wygody przyjmujemy, że liśćmi są wierzchołki zewnętrzne odpowiadające NIL. Nie zawierają one żadnych informacji poza tym, że są liśćmi (co implikuje, że są czarne).

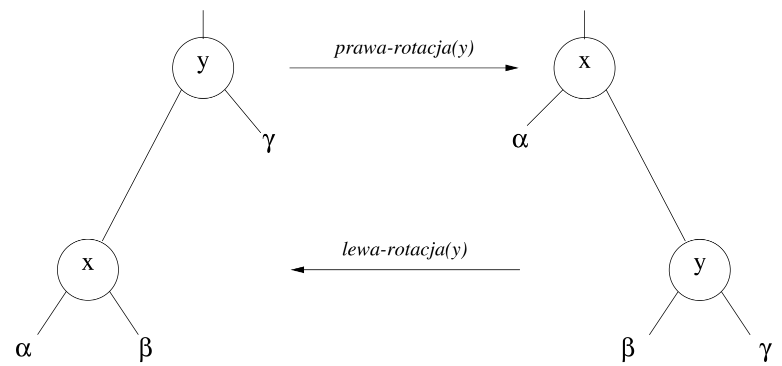
Definicja 2 *Liczbę czarnych wierzchołków na ścieżce z wierzchołka x (ale bez tego wierzchołka) do liścia nazywamy* czarną wysokością wierzchołka x *i oznaczamy* $bh(x)$.

Fakt 18 *Czerwono-czarne drzewo o n wierzchołkach wewnętrznych ma wysokość nie większą niż* $2\log(n+1)$.

DOWÓD. Przez pokazanie, że drzewo zakorzenione w dowolnym wierzchołku x zawiera co najmniej $2^{bh(x)} - 1$ wierzchołków wewnętrznych.

16.2 Operacje słownikowe na drzewach czerwono-czarnych

Operacje wstawiania i usuwania elementu mogą powodować zaburzenie własności *w1-w4*. W procedurach przywracających te własności podstawową rolę odgrywają rotacje.



WAŻNE WŁASNOŚCI:

1. Rotacje nie zmieniają porządku infiksowego elementów zapamiętanych w drzewie.
2. Pojedynczą rotację można wykonać w czasie stałym.

16.2.1 Wstawianie elementu

Wstawianie elementu wykonujemy jak w zwykłym binarnym drzewie przeszukiwań. Następnie nowemu wierzchołkowi nadajemy kolor czerwony i przywracamy drzewu własności drzewa czerwono-czarnego.
Łatwo widać, że jedyną własnością jaka mogła zostać zaburzona jest *w3*.

Przywracanie własności *w3*.

IDEA:
Początkowo *w3* może być zaburzona jedynie w miejscu gdzie nowy wierzchołek x został wstawiony (wtedy gdy ojciec x-a jest czerwony).
Wędrujemy od wierzchołka x w górę. Stosujemy operację zmiany koloru wierzchołków, by przenieść zaburzenie na przodków x-a i operację rotacji do zlikwidowania zaburzenia (uwaga: operacja rotacji będzie wykonana co najwyżej dwa razy). Będziemy przy tym

dbać, by nie zaburzyć pozostałych własności drzewa czerwono-czarnego.

Procedura przywracająca własność *w3* musi rozważyć następujące przypadki (zakładamy, że ojciec x-a jest lewym synem swojego ojca; gdy jest prawym synem otrzymujemy symetryczne przypadki):
Przypadek 1. *Wujek x-a jest czerwony.*

- $\diamond$ Zmieniamy kolory:
  - dziadka x-a malujemy na czerwono (dotąd był czarny, ponieważ ojciec x-a był czerwony i własność *w3* była zachowana),
  - ojca i wujka x-a malujemy na czarno.
- $\diamond$ $x \leftarrow$ dziadek x-a.
- $\diamond$ wywołujemy procedurę rekurencyjnie dla nowego x-a.

Przypadek 2. *Wujek x-a jest czarny i x jest prawym synem swojego ojca.*

- $\diamond$ $x \leftarrow$ ojciec(x);
- $\diamond$ $lewa\text{-}rotacja(x)$

W ten sposób otrzymujemy przypadek 3.

Przypadek 3. *Wujek x-a jest czarny i x jest lewym synem swojego ojca.*

- $\diamond$ Zmieniamy kolory:
  - dziadka x-a malujemy na czerwono.
  - ojca x-a malujemy na czarno.
- $\diamond$ $prawa\text{-}rotacja$(dziadek(x));

16.2.2 Usuwanie elementu

Wykonujemy jak w zwykłym binarnym drzewie przeszukiwań a następnie przywracamy własności *w1-w4*.

PRZYPOMNIENIE: {Usuwanie wierzchołka w drzewie binarnym.}

Niech y będzie usuwanym wierzchołkiem.

- jeśli y jest liściem, to usuwamy y;
- jeśli y ma jednego syna x, to usuwamy y a x podczepiamy pod ojca y-a;
- jeśli y ma dwóch synów, to y zastępujemy przez x - następnik y-a (tj. najmniejszy element w prawym poddrzewie y-a), usuwamy x a syna x-a (x może mieć co najwyżej prawego syna), jeśli istnieje, poczepiamy pod ojca x-a.

□

Jeśli usunięty wierzchołek y miał kolor czarny, to zaburzona zostaje własność *w4* drzewa. Może też zostać zaburzona własność *w3*.

Idea naprawy: Czarny kolor z y-a (nazwijmy go extra czarnym kolorem) przesuwamy na jego syna x (syn ten był jedynakiem i został podczepiony pod ojca y-a lub jest liściem). W ten sposób wlasności *w3* i *w4* zostają przywrócone. Jedyny kłopot spowodowany jest tym, iź x mógł być czarny i teraz zawiera podwójny czarny kolor, a więc *w1* może być zaburzona.

Procedura przywracająca *w1* przesuwa ten extra kolor w odpowiedni sposób w górę drzewa, aź:

- napotka czerwony wierzchołek, którego może pomalować extra kolorem, lub
- przesunie go do korzenia i wówczas może go usunąć, lub
- dojdzie do miejsca, gdzie może wykonać odpowiednie rotacje i zmiany kolorów.

Zakładamy, źe wierzchołek x jest lewym synem swojego ojca (gdy x jest prawym synem, rozważania są symetryczne). Musimy rozważyć następujące przypadki:

Przypadek 1. *Brat x-a jest czerwony (to implikuje, że ojciec x-a jest czerwony).*

◇ Zmieniamy kolory:

- brata x-a malujemy na czarno,
- ojca x-a malujemy na czerwono.

◇ *lewa-rotacja*(ojciec(x)).

Teraz x ma czarnego brata (jest nim były bratanek, który musiał być czarny, ponieważ miał czerwonego ojca) i otrzymujemy jeden z pozostalych przypadków.

Przypadek 2. *Brat x-a jest czarny i obydwaj jego bratankowie są czarni.*

◇ malujemy brata na czerwono,

◇ przesuwamy extra czarny kolor na ojca x-a (tj. $x \leftarrow$ojciec(x)).

Przypadek 3. *Brat x-a jest czarny, lewy bratanek - czerwony a prawy bratanek - czarny.*

◇ Zmieniamy kolory:

- brata x-a malujemy na czerwono,
- lewego bratanka malujemy na czarno.

◇ *prawa-rotacja*(brat(x)).

Teraz bratem x-a jest jego były lewy bratanek (który teraz ma czarny kolor), a prawym bratankiem jego były brat (który teraz ma kolor czerwony). Otrzymujemy przypadek 4.

Przypadek 4. *Brat x-a jest czarny a prawy bratanek czerwony.*

- ◇ Zmieniamy kolory:
  - brata x-a malujemy na kolor, którym pomalowany był ojciec,
  - ojca x-a malujemy na czarno,
  - prawego bratanka malujemy na czarno (extra kolorem z x-a).
- ◇ *lewa-rotacja*(ojciec(x)).

16.3 Koszt operacji

Koszt operacji wstawiania i usuwania elementu wynosi $O(\log n)$.