Notatki z AiSD. Nr 15. 1 maja 2020

Kopce dwumianowe

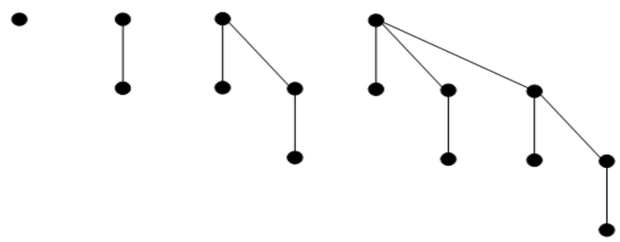
IIUWr. II rok informatyki.

Opracował: Krzysztof Loryś

1 Definicja

Kopce dwumianowe są strukturą danych umożliwiającą łatwe wykonywanie zwykłych operacji kopcowych (*insert, makeheap, findmin* i *deletemin*) a ponadto operacji *meld* łączenia kopców.

Definicja 1 Drzewa dwumianowe *zdefiniowane są indukcyjnie: i-te drzewo dwumianowe B_i składa się z korzenia oraz i poddrzew: B_0, B_1, ..., B_{i-1}.*



Rysunek 1: Cztery pierwsze drzewa dwumianowe

Fakt 1 *Drzewo B_i zawiera 2^i wierzchołków.*

Definicja 2 Kopiec dwumianowy *to zbiór drzew dwumianowych, które pamiętają elementy z uporządkowanego uniwersum zgodnie z porządkiem kopcowym.*

Definicja 3 *Niech T będzie drzewem.* Rzędem *wierzchołka w T nazywamy liczbę jego dzieci.* Rzędem drzewa *T jest rząd jego korzenia.*

Szczegół implementacyjny: Aby umożliwić szybką realizację operacji na kopcu dwumianowym, będziemy zakładać, że dzieci każdego wierzchołka zorganizowane są w cykliczną listę dwukierunkową, a ojciec pamięta wskaźnik do jednego z nich (np. do dziecka o najmniejszym rzędzie).

2 Operacje na kopcach dwumianowych

2.1 Łączenie drzew dwumianowych - operacja *join*

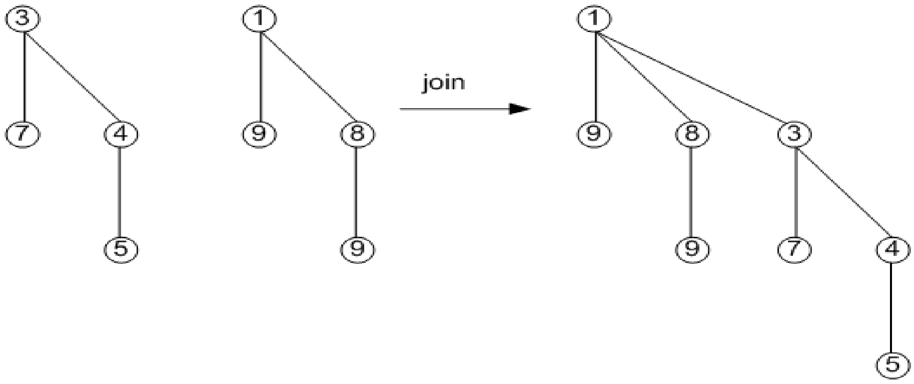
Dwa drzewa B_i łączymy ze sobą tak, że korzeń jednego drzewa staje się synem korzenia drugiego drzewa. W ten sposób otrzymujemy drzewo B_{i+1}.

Uwagi:

(a) Nigdy nie będziemy łączyć drzew o różnych rzędach.

(b) Zawsze podłączamy to drzewo, którego korzeń pamięta mniejszą wartość do tego, którego korzeń pamięta większą wartość.

Koszt operacji *link*: $O(1)$.



Rysunek 2: Łączenie dwóch drzew B_2

2.2 Operacja *makeheap*(i)

Bez komentarza. Koszt operacji - $O(1)$.

2.3 Operacja *findmin*

Z każdym kopcem dwumianowym wiążemy wskaźnik *MIN* wskazujący na minimalny element. Operacja *findmin* polega na odczytaniu tego elementu. Stąd jej koszt wynosi $O(1)$.

2.4 Operacja *insert*(i, h)

Wykonujemy $meld(h, makeheap(i))$.
Koszt tej operacji zależny jest od kosztu *meld*. Podamy go dalej.

2.5 Operacja *deletemin*(h)

Sposób jej wykonania zależy od realizacji *meld*. Omówimy go dalej.

2.6 Operacja *meld*

Rozważymy dwie metody realizacji operacji *meld*:

(a) wersja "*eager*" - w tej wersji kopiec przybiera docelowy kształt przed wykonaniem następnej po meld operacji;

(b) wersja "*lazy*" - w tej wersji pozwalamy, by kopiec utracił strukturę kopca dwumianowego; zostanie ona mu przywrócona dopiero podczas wykonania operacji deletemin.

2.6.1 Eager meld(h,h')

W tej wersji drzewa kopca dostępne są poprzez tablicę wskaźników (będziemy ją oznaczać tą samą nazwą co kopiec). Każdy kopiec zawiera co najwyżej jedno drzewo każdego rzędu. i-ty wskaźnik jest albo pusty albo wskazuje na drzewo i-tego rzędu.

$Meld(h, h')$ tworzy nowy kopiec H; stare kopce ulegają likwidacji.

Procedure Eagermeld(h,h')
  if $key(MIN_h) < key(MIN_{h'})$ **then** $MIN_H \leftarrow MIN_h$ **else** $MIN_H \leftarrow MIN_{h'}$
  $carry \leftarrow nil$;
  for $i \leftarrow 0$ **to** $maxheapsize$ **do**
    $k \leftarrow \#$ wskaźników $\neq nil$ pośród $\{carry, h[i], h'[i]\}$
    case k **of**
      0: $H[i] \leftarrow nil$
      1: $H[i] \leftarrow$ jedyny niepusty wskaźnik pośród $\{carry, h[i], h'[i]\}$
      2: $H[i] \leftarrow nil; carry \leftarrow join(B^1, B^2)$
        gdzie B^1 i B^2 są drzewami wskazywanymi przez
        dwa niepuste wskaźniki spośród $\{carry, h[i], h'[i]\}$)
      3: $H[i] \leftarrow h[i]; carry \leftarrow join(h'[i], carry)$

KOSZT: $O(\log n)$. Korzystamy tu z prostego faktu:

Fakt 2 *Kopiec zawierający n elementów składa się z co najwyżej $\log n$ różnych drzew dwumianowych.*

Operacja *deletemin*(h).

Wskaźnik MIN wskazuje na drzewo dwumianowe B, którego korzeń zawiera najmniejszy element. W stałym czasie usuwamy B z h. Następnie usuwamy korzeń z drzewa B otrzymując rodzinę drzew $B_0, B_1, \ldots, B_{\text{rząd}(B)-1}$. Z drzew tych tworzymy kopiec dwumianowy h' i wykonujemy $meld(h, h')$.

KOSZT: $O(\log n)$.

Operacja *Insert*(i, h)

Pojedyncza operacja *insert* może kosztować $\Omega(\log n)$, np. gdy h zawiera drzewa każdego rzędu. Można jednak pokazać, że czas zamortyzowany można ograniczyć do $O(1)$ (ćwiczenie).

2.6.2 Lazy meld

Chcemy, by wszystkie operacje oprócz *deletemin* kosztowały nas $O(1)$ czasu zamortyzowanego.

Zmieniamy reprezentację kopca: zamiast tablicy wskaźników, drzewa dwumianowe danego kopca łączymy w cykliczną listę dwukierunkową.

Procedura $lazymeld(h, h')$ polega na połączeniu list i aktualizacji wskaźnika MIN. Można tego dokonać w czasie $O(1)$. Teraz jednak kopiec może zawierać wiele drzew tego samego rzędu. Dopiero operacja *deletemin* redukuje liczbę drzew.

Operacja *deletemin*(h)

Usuwamy korzeń x drzewa wskazywanego przez *MIN*, dołączamy poddrzewa wierzchołka x do listy drzew kopca, uaktualniamy wskaźnik *MIN*, a następnie redukujemy liczbę drzew w kopcu. W tym celu wystarczy raz przeglądnąć listę drzew kopca (możemy roboczo wykorzystać tablicę wskaźników na wzór tablicy z wersji eager operacji *meld*).

Pojedyncza operacja *deletemin* może być bardzo kosztowna (nawet $O(n)$ - np. wtedy, gdy kopiec składa się z n drzew jednoelementowych). Pokażemy jednak, że czas zamortyzowany można ograniczyć przez $O(\log n)$.

Utrzymujemy następujący niezmiennik kredytowy:

Każde drzewo kopca ma 1 jednostkę kredytu na swoim koncie.

Operacjom przydzielamy następujące kredyty:

makeheap	-	2
insert	-	2
meld	-	1
findmin	-	1
deletemin	-	$2 \log n$

Kredyty te wystarczają na wykonanie instrukcji niskiego poziomu, związanych z realizacją operacji oraz na utrzymanie niezmiennika kredytowego:

- Operacje *meld* i *findmin* nie zmieniają liczby drzew w kopcach i wykonują się w stałym czasie.
- Operacje *insert* i *makeheap* także wykonują się w stałym czasie, ale tworzą nowe drzewo. Jedna jednostka kredytu przydzielonego tym operacjom zostaje odłożona na koncie tego drzewa.
- Operacja *deletemin* może dodać co najwyżej $\log n$ drzew do listy drzew kopca. Z kredytu operacji *deletemin* przekazujemy po jednej jednostce na konta tych drzew i wobec tego przed fazą redukcji liczby drzew każde drzewo kopca ma jedną jednostkę na swoim koncie.

  Podczas redukcji liczby drzew musimy przeglądnąć listę wszystkich drzew kopca i dokonać pewnej liczby operacji *join*. Koszt operacji *join* możemy pominąć, ponieważ każda taka operacja może być opłacona jednostką kredytu znajdującą się na koncie przyłączanego drzewa. Jednostką tą możemy opłacić także koszt odwiedzenia tego drzewa na liście.

  Odwiedzenie pozostałych drzew musimy opłacić kredytem przydzielonym operacji *deletemin*. Możemy to zrobić, ponieważ takich drzew (tj. tych, które w czasie *deletemin* nie będą podłączone do innego drzewa) jest nie więcej niż różnych rzędów, a więc $O(\log n)$. Tymi kredytami można oczywiście też opłacić obliczenie nowej wartości wskaźnika *MIN*.