Algoritmy a Dátové Štruktúry

Jana Katreniaková
katreniakova@dcs.fmph.uniba.sk

Utriedený zoznam

Abstraktná dátová štruktúra zoznam

- Postupnosť objektov - u nás čísel (nezaujíma nás, ako prichádzali, ale ich hodnota)
- Operácie hľadanie prvku, vloženie prvku, vymazanie prvku, minimum/maximum...

Možnosti implementácie

- Kompaktné množiny - polia, polia s dyn. veľkosťou
- Spájané zoznamy
- Stromy

Utriedený zoznam

Abstraktná dátová štruktúra zoznam

- Postupnosť objektov - u nás čísel (nezaujíma nás, ako prichádzali, ale ich hodnota)
- Operácie hľadanie prvku, vloženie prvku, vymazanie prvku, minimum/maximum...

Možnosti implementácie

- Kompaktné množiny - polia, polia s dyn. veľkosťou
- Spájané zoznamy
- Stromy

Utriedený zoznam pomocou kompaktných množín

Kam ukladáme?

- Veľkosť: explicitne uložená!
- Pole: opäť problém s alokáciou

Operácie

- Member: viem robiť efektívnejšie ako prejdením celého poľa
- Vkladanie: musíme nájsť miesto kam vkladáme
- Vyberanie: pozor aby nezvnikli diery O(N)
- Min, Max: je jednoznačné, kde je minumum-maximum O(1)

Utriedený zoznam pomocou kompaktných množín

Kam ukladáme?

- Veľkosť: explicitne uložená!
- Pole: opäť problém s alokáciou

Operácie

- Member: viem robiť efektívnejšie ako prejdením celého poľa
- Vkladanie: musíme nájsť miesto kam vkladáme
- Vyberanie: pozor aby nezvnikli diery O(N)
- Min, Max: je jednoznačné, kde je minumum-maximum O(1)

Utriedený zoznam pomocou kompaktných množín

```
SortedArrayMember(x,A)
1  N ← length(A); left = 0; right = N − 1;
2  while (left > right)
3  do index =??;
4      if (A[index] == x)
5        then return index;
6        else if (A[index] < x)
7                then left = index + 1;
8                else right = index − 1;
9      return −1;

SortedArrayMax(A)
1  return A[length(A) − 1];

SortedArrayDelete(x,A)
1  pos ← SortedArrayMember(x, A); N ← length(A);
2  for i← pos to N-1
3  do A[i] ← A[i + 1];
4  length(A) − −;
```

Utriedený zoznam pomocou kompaktných množín

```
SortedArrayFindPos(x,A)
 1  N ← length(A); left = 0; right = N − 1;
 2  while (left! = right)
 3  do index = (left + right)/2;
 4      if (A[index] == x)
 5        then return index;
 6        else if (A[index] < x)
 7                then left = index + 1;
 8                else right = index − 1;
 9      if A[left] < x
10        then return left + 1;
11        else return left;

InsertSortedArray(x,A)
1  pos ← SortedArrayFindPos(x, A); N ← length(A);
2  lenght(A) + +;
3  for i← N − 1 downto pos
4  do A[i + 1] ← A[i];
5  A[pos] = x;
```

Utriedený zoznam pomocou spájaných zoznamov

Kam ukladáme?

- Veľkosť: explicitne uložená!
- Implementácia pomocou pointrov
  - struct node int data; node *next
- Explicitne uložený pointer na začiatok resp. aj koniec
  - node *head;

Operácie

- Min, Max: jednoduché O(1)
- Member: pozor nevieme binárne vyhľadávanie – celý zoznam O(N)
- Vkladanie/Vyberanie: potrebujeme nájsť miesto O(N)

Utriedený zoznam pomocou spájaných zoznamov

Kam ukladáme?

- Veľkosť: explicitne uložená!
- Implementácia pomocou pointrov
  - struct node int data; node *next
- Explicitne uložený pointer na začiatok resp. aj koniec
  - node *head;

Operácie

- Min, Max: jednoduché O(1)
- Member: pozor nevieme binárne vyhľadávanie – celý zoznam O(N)
- Vkladanie/Vyberanie: potrebujeme nájsť miesto O(N)

Utriedený zoznam pomocou stromov

Kam ukladáme?

- Veľkosť: explicitne uložená!
- Implementácia pomocou pointrov
  - struct node int data; node *left; node *right;
  - potmkovia vrcholu x: x->left a x->right
- Explicitne uložený pointer na začiatok – koreň stromu
  - node *root;

Vypísanie stromu

- PreOrder
- PostOrder
- InOrder

Binárny vyhľadávací strom - BVS (alebo BST)

- $(x->left).data \leq x.data \leq (x->right).data$
- potom pri vypisovaní má význam InOrder

Utriedený zoznam pomocou stromov

Kam ukladáme?

- Veľkosť: explicitne uložená!
- Implementácia pomocou pointrov
  - struct node int data; node *left; node *right;
  - potmkovia vrcholu x: x->left a x->right
- Explicitne uložený pointer na začiatok – koreň stromu
  - node *root;

Vypísanie stromu

- PreOrder
- PostOrder
- InOrder

Binárny vyhľadávací strom - BVS (alebo BST)

- $(x->left).data \leq x.data \leq (x->right).data$
- potom pri vypisovaní má význam InOrder

Utriedený zoznam pomocou stromov

Kam ukladáme?

- Veľkosť: explicitne uložená!
- Implementácia pomocou pointrov
  - struct node int data; node *left; node *right;
  - potmkovia vrcholu x: x->left a x->right
- Explicitne uložený pointer na začiatok – koreň stromu
  - node *root;

Vypísanie stromu

- PreOrder
- PostOrder
- InOrder

Binárny vyhľadávací strom - BVS (alebo BST)

- $(x->left).data \leq x.data \leq (x->right).data$
- potom pri vypisovaní má význam InOrder

Malá odbočka - binárny strom v poli

Binárny strom nemusíme nutne reprezentovať pomocou pointrov. Dá sa ukladať (nie síce veľmi efektívne) aj v poli.

- Koreň stromu je A[1]
- Predchodca prvku A[i] je Parent(i) = $\lfloor i/2 \rfloor$
- Potomkovia prvku A[i] sú
  - Ľavý syn: Left(i) = 2*i
  - Pravý syn: Right(i) = 2*i + 1

Utriedený zoznam pomocou BVS

```
BVSMEMBER(x,*root)
1   where = *root;
2   while (where! = NULL)
3   do if ((where- > data) == x)
4         then return where;
5         else if ((where- > data) < x)
6                 then where = where- > left
7                 else  where = where- > right;
8       return NULL;
```

```
BVSMAX(*root)
1   where = *root;
2   if ((where = NULL)
3      then return NULL;
4   while (where- > right! = NULL)
5   do where = where- > right;
6   return where;
```

Utriedený zoznam pomocou BVS

Insert

- Vkladať môžem iba ako list.
- Kde je vhodný list – tam kde som skončila, keď som daný prvok hľadala

Delete

- Ak je prvok list alebo má iba jedného potomka: jednoduché
- Keď nie je, tak ho vymazať nemôžem – potrebujem za neho nájsť náhradu. Kto je dobrá náhrada?

Utriedený zoznam pomocou BVS

Insert

- Vkladať môžem iba ako list.
- Kde je vhodný list – tam kde som skončila, keď som daný prvok hľadala

Delete

- Ak je prvok list alebo má iba jedného potomka: jednoduché
- Keď nie je, tak ho vymazať nemôžem – potrebujem za neho nájsť náhradu. Kto je dobrá náhrada?

Utriedený zoznam pomocou BVS

Veľkosť stromu vzhľadom k počtu prvkov

- V peknom prípade O(log N)
- V škaredom O(N)

Časová zložitosť operácii

- Závislá od výšky stromu
- Zlý príklad binárneho stromu - zlá zložitosť
  - Riešenie: kontrolovať tvar stromu

Utriedený zoznam pomocou BVS

Veľkosť stromu vzhľadom k počtu prvkov

- V peknom prípade O(log N)
- V škaredom O(N)

Časová zložitosť operácii

- Závislá od výšky stromu
- Zlý príklad binárneho stromu - zlá zložitosť
  - Riešenie: kontrolovať tvar stromu

Utriedený zoznam pomocou AVL

AVL strom

- Dokonale vyvážený strom – každý vrchol iba ± 1
- Udržiavanie niečo stojí
- Cvičenie: Ako vyzerá najnevyváženejší AVL strom

AVL strom

- Insert – vložíme a následne upravujeme
  - Ideme smerom hore a ak nájdeme príliš nevyvážený vrchol – rotácia
- Delete – ako v BVS a následne upravujeme
  - Ideme smerom hore od nového listu až po koreň a vyvažujeme

Utriedený zoznam pomocou AVL

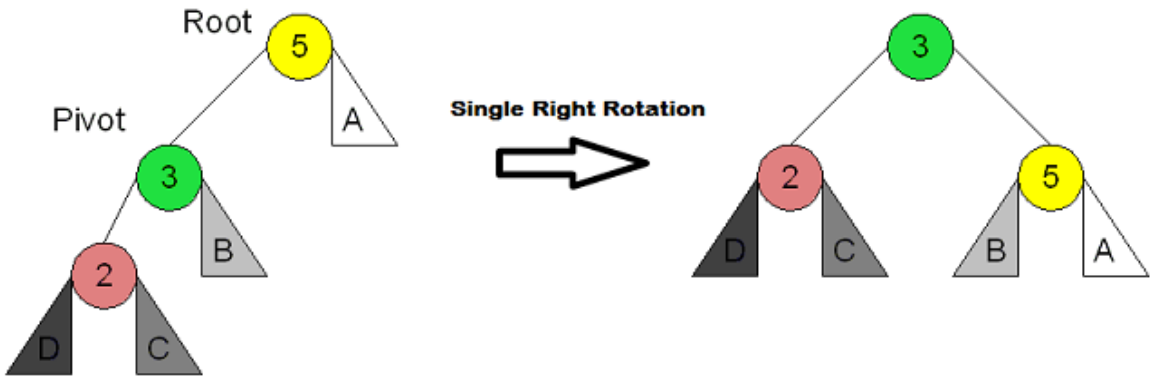
AVL strom

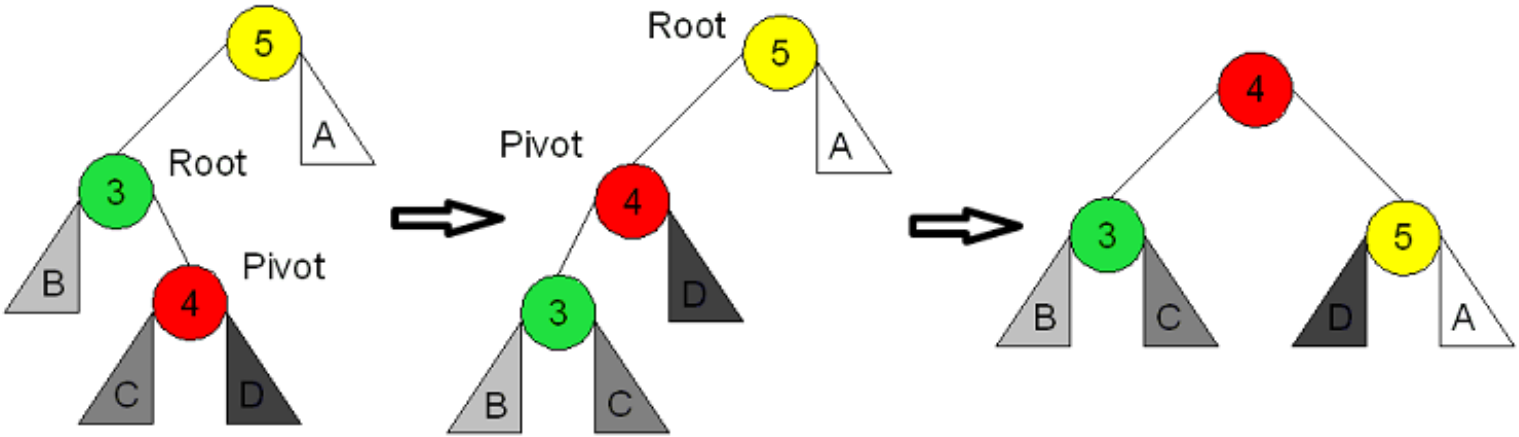
- Dokonale vyvážený strom – každý vrchol iba ± 1
- Udržiavanie niečo stojí
- Cvičenie: Ako vyzerá najnevyváženejší AVL strom

AVL strom

- Insert – vložíme a následne upravujeme
  - Ideme smerom hore a ak nájdeme príliš nevyvážený vrchol – rotácia
- Delete – ako v BVS a následne upravujeme
  - Ideme smerom hore od nového listu až po koreň a vyvažujeme

Rotácie AVL


Root
5
Pivot
3
A
B
2
D
C
Single Right Rotation
3
2
5
D
C
B
A


5
A
Root
3
B
Pivot
4
C
D
Root
5
Pivot
A
4
D
3
B
C
4
3
5
B
C
D
A

Insert do AVL

- Vložím vrchol na miesto kam patrí.
- Postupujem od neho hore a kontrolujem vyváženosť.
- Nech x je vrchol, kde neplatí AVL podmienka. Ak $h(x)$ je h+3, potom sú 4 prípady
  - Výška left(x) je h+2 (i.e. výška right(x) je h)
    - Výška left(left(x))=h+1 $\rightarrow$ SingleRotateL
    - Výška right(left(x))=h+1 $\rightarrow$ DoubleRotateL
  - Výška right(x) is h+2 (i.e. výška left(x) je h)
    - Podobne len vpravo

Delete z AVL

- Delete vrchola x – ako v bin. vyhľadávacom strome
- Prechádzame od nového listu smerom ku koreňu.
- Cestou kontrolujeme, či pre vrchol x výšky spĺňajú podmienku
  - Ak áno, pokračuj vrcholom parent(x).
  - Ak nie, vykonaj vhodnú rotáciu – ako pri Insert
- Musíme pokračovať až po koreň stromu!

Zhrnutie AVL

- Reštrukturalizácia stromu – Rotácia O(1)
- Vyhľadávanie prvku – Member O(log n)
- Vloženie prvku – Insert O(log n)
  - Samotné vloženie O(log n)
  - Úprava smerom hore O(log n)
- Vymazanie prvku – Delete O(log n)
  - Vymazanie a náhrada listami O(log n)
  - Úprava smerom hore O(log n)