Výuka datových struktur

Termín odevzdání:	**18.11.2018 23:59:59**
Hodnocení:	**3.7500**
Max. hodnocení:	**2.0000** (bez bonusů)
Odevzdaná řešení:	15 / 25
Nápovědy:	0 / 0

Datové struktury se na naší fakultě vyučují ve spoustě předmětů -- jmenovitě například v AG1/2, PA1/2, atp. Jakpak by taky ne, jedná se totiž o velmi důležitý nástroj pro efektivní práci s většími objemy dat. Není však vhodné začít vyučovat stuktury počínaje jejich nejpokročilejšími variantami, neboť student by se mohl velmi rychle ztratit. Z toho důvodu je lepší jako první ukázat jednodušší variantu dané datové struktury, ukázat její nevýhody a nedokonalosti, a na těch postupně zakládat její vylepšení. Tímto způsobem student datové struktuře porozumí mnohem lépe.

Problém nastává tehdy, když chceme jednodušší variantu datové struktury vizualizovat během použití velkého objemu dat -- z jednoduchosti struktury totiž plyne přirozený limit na vstupní data. Například základní implementace binárního vyhledávacího stromu může pro určitou posloupnost operací degenerovat až do té míry, že pro provedení dalších operací začne být zapotřebí lineární čas. To se vyučujícím ovšem nelíbí a chtěli by jednoduché struktury vizualizovat studentům v rozumném čase. Proto jste byli na pomoc povoláni vy. Vaše první vizualizovaná struktura bude právě binární vyhledávací strom.

Vašim úkolem je simulovat chování binárního vyhledávacího stromu (dále BVS) **přesně** podle přednášky AG1 pro použití v externím vizualizačním programu. Přednášku s referenčním popisem BVS naleznete **na tomto odkaze**. Vizualizační program bude s vaší implementací komunikovat přes předem určené rozhraní. Vstupem pro vaší strukturu budou standardní operace nad BVS, jako výstup si vizualizační program bude žádat o zodpovězení dotazů na rodiče vrcholu v BVS, který reprezentuje zadaný klíč. Pro zajištění únosné odezvy vizualizačního programu u velkých objemů dat (testy velkými daty) je zapotřebí, aby vaše implementace simulovala chování BVS dostatečně efektivně -- i v těchto testech však vaše implementace musí stále umět zodpovídat dotazy na strukturu vizualizovaného BVS. Jelikož se jedná o první testovací realizaci, je prozatímní vizualizační program zastaralý a nepodporuje komunikaci s programy využívajícími určité knihovny. Z toho důvodu nemůžete ve vaší implementaci použít knihovnu STL.

Formát vstupu a výstupu:

- Vstup sestává z určitého počtu řádek na vstupu, kde každá kóduje jeden příkaz, viz níže uvedené příkazy v uvozovkách:

  `"1 X"`
  : kde `X` je přirozené číslo, $1 \leq X \leq 10^9$, znamená přidej do BVS klíč `X`.

  `"2 X"`
  : kde `X` je přirozené číslo, $1 \leq X \leq 10^9$, znamená odeber z BVS klíč `X`.

  `"3 X"`
  : kde `X` je přirozené číslo, $1 \leq X \leq 10^9$, znamená vypiš pro vrchol BVS reprezentující klíč `X` jeho rodiče v BVS; nemá-li tento vrchol v BVS rodiče, vypiš řetězec `"noparent"` (bez uvozovek).

  `"4 X"`
  : kde `X` je přirozené číslo, $1 \leq X \leq 10^9$, znamená vypiš následníka klíče `X` v BVS; nemá-li klíč `X` v BVS následníka, vypiš řetězec `"nosuccessor"` (bez uvozovek)

  `"5 X Y"`
  : kde `X`, `Y` jsou přirozená čísla, $1 \leq X \leq 10^9$ a $1 \leq Y \leq 2$, znamená proveď jednoduchou rotaci přes vrchol BVS reprezentující klíč `X` (pro `Y = 1` levou, pro `Y = 2` pravou); nedává-li rotace smysl (vrcholu chybí levý syn pro pravou rotaci, či naopak), vypiš řetězec `"norotate"` (bez uvozovek)

  `"6"`
  : znamená, že vstup do struktury skončil a že žádný další příkaz již následovat nebude; tento příkaz je zaručeně jako poslední na vstupu.

- Můžete se spolehnout, že vstup je zadán korektně.

Dále:

- Nenachází-li se pro příkaz typu 3, 4 nebo 5 zadaný klíč `X` v BVS, vypište na výstup řádek s řetězcem `"notfound"` (bez uvozovek).
- Může se stát, že váš program dostane příkaz typu 2 na smazání takového klíče, který se v BVS nevyskytuje. V takovém případě požadavek ignorujte.
- Naopak se nikdy nestane, že by se v BVS najednou vyskytovalo (po příslušných požadavcích na přidání) více klíčů o stejné hodnotě. Pokud však bude klíč o určité hodnotě z BVS v nějaký moment odebrán, může být později do BVS znovu přidán.

Bodové podmínky:

1. Pro splnění povinných testů je zapotřebí, aby program fungoval korektně pro vstupy o nejvýše 1 000 příkazech. Příkazy budou pouze typu 1, 2, 3 a 6.

2. Pro splnění testu rozšířených operací musí program splnit časový limit pro vstupy o nejvýše 1 000 příkazech. Příkazy mohou být všech typů.
3. Pro splnění testu velkými daty #1 musí program splnit časový limit pro vstupy o nejvýše 1 000 000 příkazech. Příkazy budou pouze typu 1, 3, 4, 5 a 6.
4. Pro splnění testu velkými daty #2 musí program splnit časový limit pro vstupy o nejvýše 1 000 000 příkazech. Příkazy mohou být všech typů.

- Testy podle ukázky jsou z důvodu omezení typů příkazů na vstupu provedeny ve dvou různých testech. První z nich testuje ukázkové vstupy 1 a 2 a je povinný. Druhý testuje ukázkové vstupy 3 a 4 a je nepovinný.

Ukázka práce programu:

Příklad vstupu 1:

```
1 30
1 10
1 20
1 25
3 10
3 20
3 30
3 25
3 1000
6
```

Příklad výstupu 1:

```
30
10
noparent
20
notfound
```

Příklad vstupu 2:

```
1 10
1 20
1 15
1 12
1 17
3 10
3 20
3 15
3 12
3 17
2 17
3 10
3 20
3 15
3 12
3 17
2 15
3 10
3 20
3 15
3 12
3 17
6
```

Příklad výstupu 2:

```
noparent
10
20
15
15
noparent
10
20
15
```

```
notfound
noparent
10
notfound
20
notfound
```

Příklad vstupu 3:

```
1 2
1 1
1 4
1 3
1 5
3 1
3 2
3 3
3 4
3 5
4 1
4 2
4 3
4 4
4 5
6
```

Příklad výstupu 3:

```
2
noparent
4
2
4
2
3
4
5
nosuccessor
```

Příklad vstupu 4:

```
1 3
1 1
1 4
1 2
3 1
3 2
3 3
3 4
5 3 2
3 1
3 2
3 3
3 4
5 1 2
6
```

Příklad výstupu 4:

```
3
1
noparent
3
noparent
3
1
3
norotate
```