UNICORN COLLEGE

Zadat úkol skupině č. 1 | Zadat úkol skupině č. 2

Kontext

Uvedli jsme si několik datových struktur umožňujících ukládání a vyhledávání dat. Cílem tohoto cvičení je vyzkoušet si v praxi jejich chování.

Zadání

Implementace

K dispozici (v příloze) naleznete implementaci následujících tří vyhledávacích datových struktur.

- Třída **UnsortedArray** obaluje obyčejné, nijak neudržované pole.
- Třída **SortedArray** obaluje pole, které ovšem při modifikujících operacích udržujeme setříděné. Všechny tři operace (*find*, *insert*, *delete*) používají binární vyhledávání, navíc ještě třída ale obsahuje i implementaci interpolačního vyhledávání.
- Třída **BinarySearchTree** implementuje binární vyhledávací strom. Kromě vyhledávacích operací jsou na ní implementovány ještě dvě užitečné metody, *to_s* a *to_svg*. První z nich vrátí v řetězci uzávorkovanou reprezentaci celého stromu. Druhá metoda reprezentaci stromu uloží do souboru zadaného jména ve formátu SVG (Scalable Vector Graphics). Ten je možné si pak otevřít například v grafickém editoru Inkscape nebo přímo v některých webových prohlížečích a názorně si prohlédnout vizuální zachycení stromu.

Vaším úkolem je porovnat rychlost těchto tří rozdílných datových struktur v závislosti na množství prvků v nich uložených (tzv. Benchmark) a sepsat o tom zprávu (tzv. Referát).

Benchmark

Změřte dobu běhu jednotlivých operací na různě velkých strukturách. Postupně pro následující dvojice čísel M, N: (10, 50000), (50, 10000), (100, 5000), (500, 1000), (1000, 100), (5000, 10) (M, první číslo, vyjadřuje velikost pole, N, druhé číslo, počet opakování) opakujte:

1. Vytvořte N polí velikosti M. Každé z polí bude obsahovat M náhodných hodnot z rozmezí 0 až (M−1). Tato pole budou obsahovat prvky do struktur **vkládané**.
2. Vytvořte (dalších) N polí velikosti M. Každé z polí bude obsahovat M náhodných hodnot z rozmezí 0 až (M−1). Tato pole budou obsahovat prvky ve strukturách **hledané**.
3. Vytvořte (dalších) N polí velikosti M. Každé z polí bude obsahovat M náhodných hodnot z rozmezí 0 až (M−1). Tato pole budou obsahovat prvky ze struktur **mazané**.
4. Vytvořte N prázdných instancí *UnsortedArray*.
5. Vytvořte N prázdných instancí *SortedArray*.
6. Vytvořte N prázdných instancí *BinarySearchTree*.

7. V cyklu pro i od 0 do (N−1) **vložte** všech M hodnot z i-tého pole z bodu 1 do i-té instance *UnsortedArray* z bodu 4. Změřte, jak dlouho tento celý cyklus potrvá.
8. V cyklu pro i od 0 do (N−1) **vložte** všech M hodnot z i-tého pole z bodu 1 do i-té instance *SortedArray* z bodu 5. Změřte, jak dlouho tento celý cyklus potrvá.
9. V cyklu pro i od 0 do (N−1) **vložte** všech M hodnot z i-tého pole z bodu 1 do i-té instance *BinarySearchTree* z bodu 6. Změřte, jak dlouho tento celý cyklus potrvá
10. V cyklu pro i od 0 do (N−1) **proveďte vyhledání** všech M hodnot z i-tého pole z bodu 2 v i-té instanci *UnsortedArray* z bodu 4. Změřte, jak dlouho tento celý cyklus potrvá.
11. V cyklu pro i od 0 do (N−1) **proveďte vyhledání** všech M hodnot z i-tého pole z bodu 2 v i-té instanci *SortedArray* z bodu 5. Změřte, jak dlouho tento celý cyklus potrvá (a) pro **binární vyhledávání**, (b) pro **interpolační vyhledávání**.
12. V cyklu pro i od 0 do (N−1) **proveďte vyhledání** všech M hodnot z i-tého pole z bodu 2 v i-té instanci *BinarySearchTree* z bodu 6. Změřte, jak dlouho tento celý cyklus potrvá.
13. V cyklu pro i od 0 do (N−1) **smažte** všech M hodnot z i-tého pole z bodu 3 v i-té instanci *UnsortedArray* z bodu 4. Změřte, jak dlouho tento celý cyklus potrvá.
14. V cyklu pro i od 0 do (N−1) **smažte** všech M hodnot z i-tého pole z bodu 3 v i-té instanci *SortedArray* z bodu 5. Změřte, jak dlouho tento celý cyklus potrvá.
15. V cyklu pro i od 0 do (N−1) **smažte** všech M hodnot z i-tého pole z bodu 3 v i-té instanci *BinarySearchTree* z bodu 6. Změřte, jak dlouho tento celý cyklus potrvá.

Výsledkem budou celkové doby

- N-násobného vložení M prvků do tří druhů datových struktur (pro každou z nich zvlášť),
- N-násobného vyhledání M prvků v nich a
- N-násobného smazání M prvků z nich.

Počítejte s tím, že výpočet bude několik minut trvat (pochopitelně záleží na použitém hardwaru).

K měření doby trvání výpočtu použijeme modulu Benchmark. Přímo voláním metody Benchmark.realtime, které předáme blok, získáme čas v sekundách, jaký trvalo blok vykonat. Použití tedy může vypadat například přibližně takto:

```
require "benchmark"
#
# 10násobné opakování
# ...
# pro každou dvojici (M, N)
  # vygenerování polí
  # ...
  elapsed = Benchmark.realtime do
    # cyklus přes N
      # cyklus přes M
        # vložení/vyhledání nebo smazání prvku v odpovídající datové struktuře
      # konec cyklu přes M
      # případné uložení počtu prvků ve struktuře
    # konec cyklu přes N
  end
  # ...
```

Pro všechny dvojice M, N

- proveďte měření času 10krát a spočítejte z těchto 10 hodnot *aritmetický průměr* a *medián* **doby provedení jedné sady operací** (vkládání, hledání, mazání) nad N strukturami stejného typu (nesetříděné pole, setříděné pole, binární vyhledávací strom) -- tím dostanete pro každou operaci a pro každou dvojici (M, N) časy $T_{M,N,med}$ a $T_{M,N,avg}$,
- spočítejte **časy M operací nad jedním celým polem** (tedy vlastně spočítané mediány a průměry vydělte číslem N) -- označme tyto časy $t_{M,med} = T_{M,N,med}/N$ a $t_{M,avg} = T_{M,N,avg}/N$, opět pro každou operaci a pro každou dvojici (M, N) zvlášť,
- nakonec spočítejte **čas jedné operace** insert/find/delete nad polem velikosti M, který dostanete vydělením předešlé hodnoty počtem operací, tj. číslem M: $t_{med} = t_{M,med}/M$, resp. $t_{avg} = t_{M,avg}/M$ (opět pro každou operaci a dvojici (M, N) zvlášť).
- Při měření také ukládejte výsledný počet prvků ve strukturách a nakonec spočítejte *průměr* $n_{M,avg}$ a *medián* $n_{M,med}$ **počtu prvků**, které struktury obsahovaly po provedení bloku operací *insert* a po provedení bloku operací*delete* (uvědomte si, že operace insert nevkládá duplicity, takže je velmi pravděpodobné, že po M operacích *insert* nebude vloženo M čísel, stejně tak volbou náhodných hodnot při operacích *delete* mnohdy mažeme čísla, která ve strukturách vůbec nejsou).

Poznámka. Nejzajímavější číslo je určitě *čas jedné operace* t_{med}, resp. t_{avg}, a tyto časy by bylo vhodné i kvantitativně srovnat s teoretickou složitostí algoritmů nad danými strukturami.

Výstup benchmarku

Zpracujte kód měření do skriptu searchbench.rb tak, že výstupem skriptu bude soubor times_avg.txt s hodnotami t_{avg} a times_med.txt s hodnotami t_{med} v následujícím formátu

```
# m       UA_insert SA_insert BST_insert UA_search SA_binarys SA_interps BST_search UA_delete SA_delete BST_delete
10        1.23e-5   1.07e-5   ...        ...       ...
50        2.94e-5   ...       ...
100       7.24e-4   ...
500       3.92e-2
1000      4.37e-1
5000      2.12
```

Dalším výstupem skritup searchbench.rb bude soubor counts.txt s hodnotami $n_{M,med}$ v podobném formátu jako předešlé soubory

```
# m       UA_insert SA_insert BST_insert UA_delete SA_delete BST_delete
10        7         8         ...        ...       ...
50        39        48        ...        ...
100       78.5      62.5      ...
500       342       ...
1000      799
5000      4927
```

Formát těchto tří souborů tedy bude prostý text s úvodním řádkem s popisky sloupců uvozeným znakem #. Následovat budou řádky s hodnotami v zarovnaných sloupcích oddělených mezerami. První sloupec budou hodnoty M (tedy délky polí), další sloupce budou jednotlivé časy nebo počty prvků v uvedeném pořadí a formátu.

Výstup: soubory `searchbench.rb`, `times_avg.txt`, `times_med.txt` a `counts.txt`

Referát

Referativní formou sepište závěry, které vyplývají z naměřených hodnot, především porovnejte rozdíly v době trvání jednotlivých operací nad jednotlivými strukturami v závislosti na počtu prvků v těchto strukturách. Zamyslete se, co má na vhodnost použití té či oné struktury kromě množství prvků vliv (například poměr očekávané četnosti jednotlivých operací v reálném použití). Do referátu vložte tabulku a grafy s mediány a průměry naměřených hodnot, a dále všechny další spočítané hodnoty uvedené v předchozí sekci. V příloze naleznete dokument, který můžete použít jako šablonu.

Referát není slohové cvičení, není potřeba psát dlouhé texty, *jde především o faktický obsah*. Přesto by měl mít určitou úroveň a text musí tvořit souvislé smysluplné odstavce, představte si například, že byste tento text chtěli publikovat jako článek. Toto shrnutí naměřených výsledků a vlastní zamyšlení bude z celé úlohy hodnoceno podstatnou částí bodů. To nicméně neznamená, že ostatním částem je možno věnovat méně pozornosti. Pokud například program starající se o měření doby běhu bude chybný a naměří nesmyslná data, je pochopitelné, že i referát pravděpodobně dojde k nevhodným závěrům.

Výstup: soubor `referat.pdf`

Poznámky

Co se tedy očekává, že odevzdáte:

- skript (skripty) provádějící měření rychlosti operací, tj. soubor `searchbench.rb`
- textové výstupu skriptu benchmarku, tj. soubory `times_avg.txt`, `times_med.txt` a `counts.txt`
- referát naměřených výsledků (ve formátu PDF), tj. soubor `referat.pdf`

Skript(y) obsahující měření budou vhodně zdokumentovány (především bude u skriptu provádějícího měření popsáno, jakým

způsobem jej spustit a jaké jsou jeho očekávané výstupy).

Způsob odevzdání

- všech 5 požadovaných souborů (případně další, které vytvoříte a budou nutné) uložte do adresáře `prijmeni_jmeno_du2` , kde `prijmeni` a `jmeno` jsou vaše příjmení a jméno **malými písmeny bez diakritiky**
- celý adresář zabalte do souboru `prijmeni_jmeno_du2.zip` nebo `prijmeni_jmeno_du2.7z`
- v +4U artefaktu s odevzdáním vyplňte na začátek políčka **Název** řetězec `DU2`
- Nedodržení tohoto způsobu uložení bude mít za následek snížení počtu bodů, případně nemožnost úkol opravit.

Nezapomeňte mít počítač během měření co nejvíce „v klidu“, ideálně tak, aby úlohu nenarušovaly žádné další aplikace. I přehrávač hudby, který působí dojmem, že při hraní nebere takřka žádný procesorový čas (případně by se člověk mohl domnívat, že pokud bere, tak bere „všem stejně“), musí čas od času načíst z disku do paměti nějaká další data k přehrávání, což sice netrvá čas postřehnutelný okem, nicméně měření v řádu desítek či stovek milisekund může snadno ovlivnit.

Řešení

Řešení posílejte tlačítkem níže:

Odeslat řešení domácího úk...

Zdroje