Porównanie wydajności drzew van Emde Boasa, x-fast oraz y-fast

Mateusz Lewko

Wykonane prace Zaimplementowałem trzy struktury danych z wykładu:

- drzewo van Emde Boasa (`v-eb.h`),
- x-fast trie (`x-fast.h`),
- y-fast trie wykorzystujące x-fast oraz drzewo bst (`y-fast.h`),

oraz wrapper na std::set (dla porównania, jest w pliku (`test.h`)).

Każda ze struktur implementuje taki sam interfejs (lookup, successor, insert — plik (`interface.h`)). Konstruktor każdej struktury przyjmuje wykładnik k z rozmiaru uniwersum $U = 2^k$. Jeśli w strukturze nie ma następnika dla danego zapytania, to zwracane jest U. Zarządzanie pamięcią jest zapewnione przez użycie shared pointerów. Skutkiem ubocznym użycia shared pointerów jest duża stała wykonywanych operacji.

Poprawność zaimplementowanych metod jest sprawdzana poprzez wypisanie xora wyników operacji *successor* dla każdej struktury.

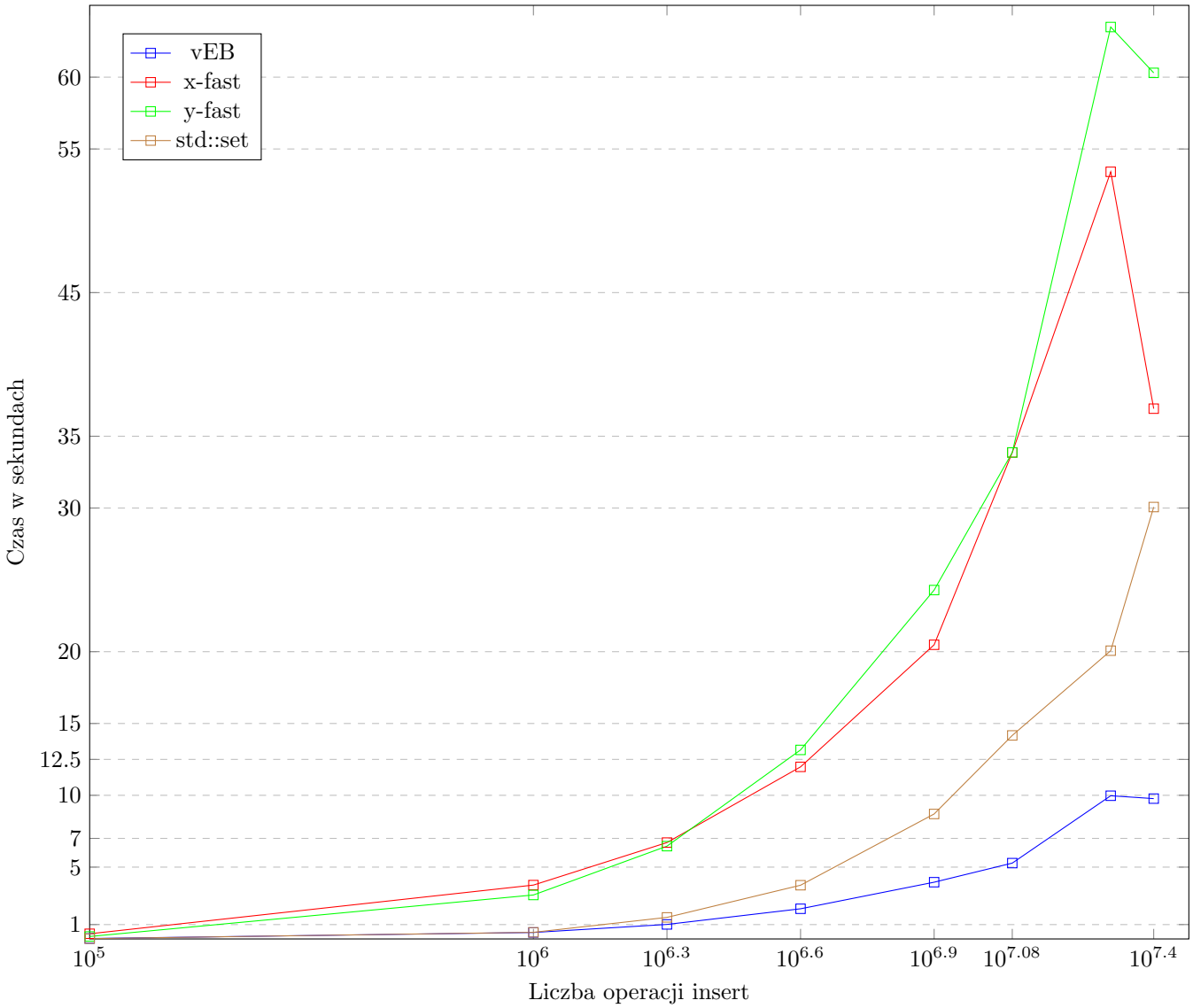
Testowanie Dla danego N losowane jest N liczb jako wejście do operacji insert oraz lookup, a następnie N liczb na których kolejno wykonywane są zapytania o następnika. Liczby są jednostajnie losowane z przedziału $[0, U)$. We wszystkich testach przyjęto $U = 2^{25}$. Mierzony jest sumaryczny czas wykonania wszystkich operacji (z podziałem na typ operacji).

Kompilacja Wszystkie testy można skompilować jednym poleceniem w systemie linux: `g++ test.cpp -o test -O3 -Wall -Wextra -std=c++17`, a następnie uruchomić: `./test`.

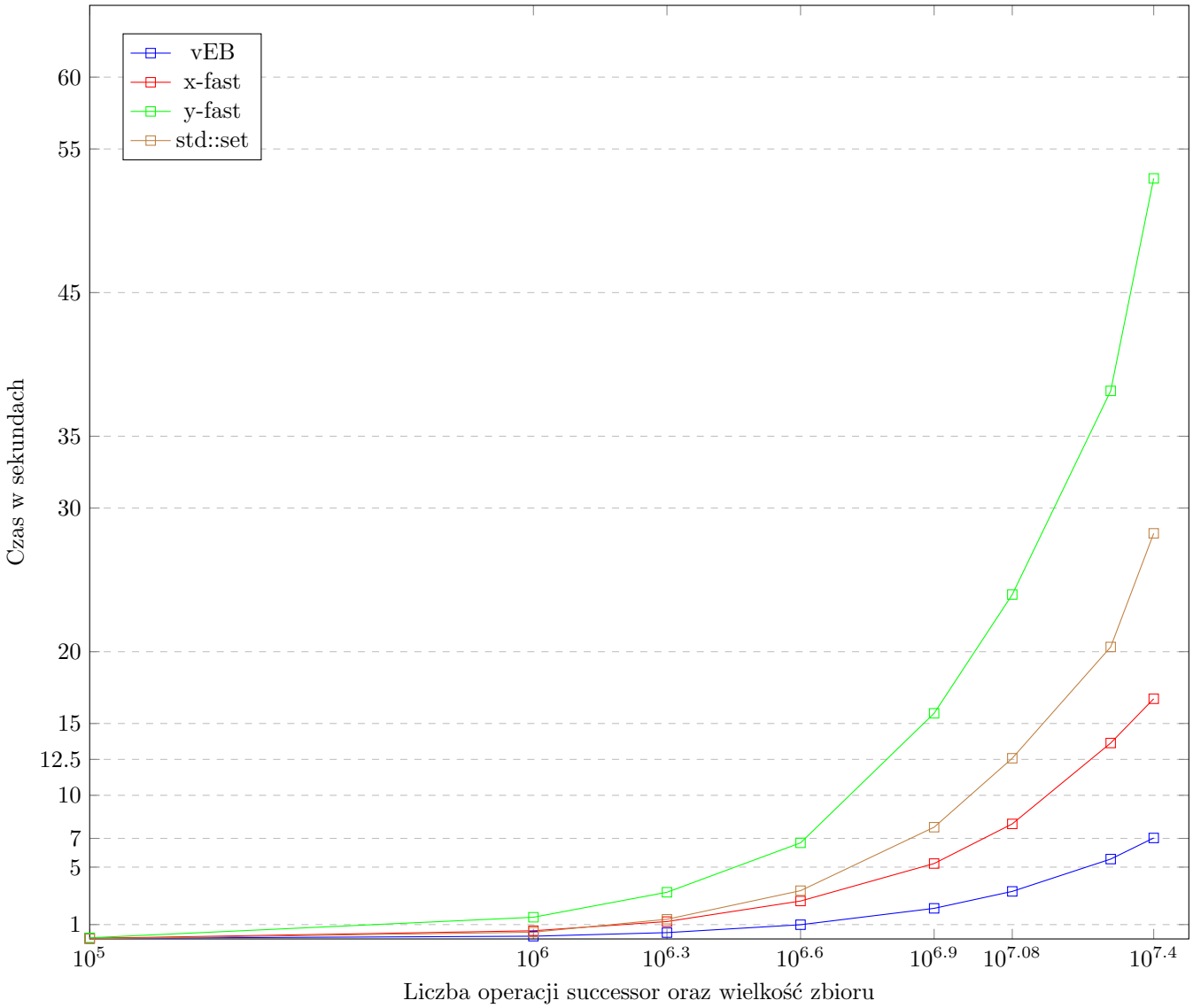
Wnioski Dostępy do pamięci i inne kosztowne operacje nie dają tak dużej różnicy w czasie działania jak można byłoby oczekiwać od złożoności $O(\log \log U)$. Drzewo van Emde Boasa okazało się najszybsze dla każdych danych testowych, jednak bardzo duże zużycie pamięci oznacza małą przydatność w praktyce oraz długi czas tworzenia struktury (alokowanie pamięci). Implementacja drzewa y-fast okazała się mieć na tyle dużą stałą, że jej sumaryczny czas działania jest większy niż struktury std::set z biblioteki standardowej C++ (o gorszej złożoności). Drzewa x-fast wypadły najlepiej pod względem czasu działa i wyprzedzają std::set już dla danych rzędu $2 * 10^6$.

Na końcu pliku `test.cpp` znajduje się zakomentowany wynik programu. Czasy działa poszczególnych operacji zostały przedstawione na poniższych wykresach.

Wykres 1. Sumaryczny czas działania operacji insert



Wykres 2. Sumaryczny czas działania operacji successor



Wykres 3. Sumaryczny czas działania operacji lookup