Zaawansowane biblioteki programistyczne

Temat: Suffix Tree

Autorzy:
Łukasz Spintzyk
Piotr Zemczak

1. Instrukcja obsługi.

   SuffixTree posiada jedną główną metodę: isSuffix(string tekst) która sprawdza czy w drzewie znajduje się dany tekst.

   Dostarczone są także specjalizacje szablonu:
   - typedef SuffixTreeTemplate<unsigned long long> SuffixTreeLL; - używany dla tekstów dłuższych niż 4 GB.
   - typedef SuffixTreeTemplate<unsigned int> SuffixTreeINT; - używany dla tekstów dłuższych niż 65356 znaków
   - typedef SuffixTreeTemplate<unsigned short> SuffixTreeSHRT; - używany dla tekstów dłuższych niż 256 znaków
   - typedef SuffixTreeTemplate<unsigned char, SuffixTreeTraits<unsigned char, char> > SuffixTree; - używany dla tekstów krótszych niż 256 znaków.

   W zasadzie tylko dwie pierwsze opcje mają większy sens, ponieważ dla krótkich tekstów inne metody wyszukiwania mogą być znacznie efektywniejsze.

   Przykład użycia:

   typedef SuffixTreeTemplate<> SuffixTree;
   SuffixTree suffix_tree("Ala ma kota");
   cout << suffix_tree.isSuffix("ma");

2. Porównanie wydajności.

   W celu zbadania wydajności naszego SuffixTree postanowiliśmy porównać go z funkcją find znajdującą się w klasie string.
   Test polegał na wyszukaniu ciągu znaków w zadanym tekscie.

   Tekst w którym dokonywaliśmy wyszukania, to dwe wersje pierwszej księgi "Lalki" Bolesława Prusa. Pierwszy tekst jest oryginalną księgą, a drugi jest powieleniem tego tekstu 10 krotnie:

   lalka.txt (800k
   lalka10.txt (8M

   W powyższych tekstach były wyszukiwane następujące ciągi które faktycznie znajdują się w tekście:

test1 = " I przypomniał sobie całe szeregi ludzi obdartych,"
test2 = "Rzecz dziwna! przecie miał już ustaloną opinię hojnego filantropa. Członkowie Towarzystwa Dobroczynności we frakach składali mu podziękowania za ofiarę dla wiecznie łaknącej instytucji; hrabina Karolowa we wszystkich salonach opowiadała o pieniądzach, które złożył na jej ochronę; jego służba i subiekci sławili go za podwyższenie im pensji. Ale Wokulskiemu rzeczy te nie sprawiały żadnej przyjemności, tak jak on sam nie przywiązywał do nich żadnej wagi. Rzucał tysiące rubli do kas urzędowych dobroczyńców, ażeby kupić za to rozgłos nie pytając, co się zrobi z pieniędzmi."

Pierwszy z nich zawiera 50 znaków, a drugi 570.
Dodatkowo oprócz powyższych w tekstach wyszukiwane były ciągi których nie ma:

test3 = "SuffixTreeI przypomniał sobie całe szeregi ludzi obdartych,"
test4 = "SuffixTreeRzecz dziwna! przecie miał już ustaloną opinię hojnego filantropa. Członkowie Towarzystwa Dobroczynności we frakach składali mu podziękowania za ofiarę dla wiecznie łaknącej instytucji; hrabina Karolowa we wszystkich salonach opowiadała o pieniądzach, które złożył na jej ochronę; jego służba i subiekci sławili go za podwyższenie im pensji. Ale Wokulskiemu rzeczy te nie sprawiały żadnej przyjemności, tak jak on sam nie przywiązywał do nich żadnej wagi. Rzucał tysiące rubli do kas urzędowych dobroczyńców, ażeby kupić za to rozgłos nie pytając, co się zrobi z pieniędzmi."

Pierwszy z nich zawiera 60 znaków, a drugi 580.

Wyszukanie każdego z podtekstów zostało wykonane 10000 razy w pętli po czym wyciągnięta została średnia wyszukania wzorca.
Wyniki badań przedstawione są w poniższej tabeli (czas w ms):

	Lalka (800KB)		Lalka x10 (8MB)	
	Suffix Tree	String	Suffix Tree	String
Test1	0.007	0.631	0.007	0.631
Test2	0.027	0.651	0.026	0.650
Test3	0.002	2.746	0.002	27.627
Test4	0.003	2.744	0.002	27.628

Jak widać z powyższej tabeli wyszukanie za pomocą SuffiX Tree jest kilkunastokrotnie szybsze niż funkcja find ze stringa. Szczególnie jest to widoczne dla dużych tekstów w których szukamy.

3. Skomentowany kod.

   Dokumentacja poszczególnych klas została wykonana przy pomocy Doxygena i została zawarta razem z projektem.