/***
 * Nome: Lucas Stefan Abe
 * Nº USP: 8531612
 ***/

- Mostre a comparação de tempo para entradas grandes e vários thresholds (tamanho mínimo das palavras), mostrando a diferença de tempo entre a versão com caching e a versão sem.
R: Para fazer a comparação das 3 versões, foram usados 3 arquivos de texto com tamanhos variados. Para cada um desses textos foram realizados testes com 4 valores de thresholds (L).

Legenda :
Mais lento
Mais rápido

Bible.txt (4 MB)

L	Red Black (Com cache)	Red Black (Sem cache)	Simbol table (ST.java)
1	0.955 s	1.203 s	1.028 s
4	0.914 s	1.039 s	0.898 s
8	0.743 s	0.821 s	0.706 s
10	0.777 s	0.794 s	0.722 s

leipzig300K.txt (13 MB)

L	Red Black (Com cache)	Red Black (Sem cache)	Simbol table (ST.java)
1	2.486 s	3.413 s	2.872 s
4	2.277 s	3.057 s	2.622 s
8	1.831 s	1.953 s	1.756 s
10	1.61 s	1.671 s	1.541 s

leipzig1M.txt (130 MB)

L	Red Black (Com cache)	Red Black (Sem cache)	Simbol table (ST.java)
1	19.771 s	32.551 s	27.036 s
4	18.409 s	28.31 s	24.218 s
8	13.239 s	15.323 s	14.425 s
10	10.951 s	11.999 s	11.978 s

- A versão com caching é mais rápida que a versão sem caching? Se sim, como você explica essa diferença?
R: Certamente a versão com caching é mais rápida do que a sem caching, em todos os testes realizados a versão sem caching demorou mais tempo do que a versão com caching. Isso deve-se ao fato de que pela interface da RB não podemos acessar o nó diretamente, somente atráves do método get e put que usam
a chave do nó para buscar seu valor e alterar seu valor, assim guardar o nó mais recente é uma boa solução para contornar essee problema. Por exemplo, no caso de uma chave repetida:
if (st.contains(key))

```
    st.put(key, st.get(key) + 1);
```
ocore o triplo de buscas para a versão sem caching. No caso em que a chave é repetida e foi lida anteriormente, nem ocorre busca na versão com caching.

- Em comparação com a versão original no site do livro (que usa a tabela de símbolos ST.java), a sua versão é mais eficiente? Caso sim, como você explica essa diferença? Como é implementada a tabela de símbolos ST.java?

Pelos testes, temos que a versão com caching é mais eficiente para volumes de dadas bem grandes e com pequenos thresholds, de tal forma que a quantidade de palavras válidas seja bem grande. A versão com caching é mais eficiente pois guarda o último nó acessado, poupando vários acessos desnecessários na árvore. A tabela de símbolos ST.java é implementada usando a classe TreeMap da java.util, que também usa uma árvore de busca balanceada.