Google Search Simulation 3.0: Implementation of Red-Black Tree Methods

Uyen Nguyen

CS 146: Data Structures and Algorithms

Professor Mike Wu, Section 6, San Jose State University

About the Design and Implementation

The assignment is based off the simulator used in Programming Assignment II, where the program allows the user to sort and manipulate data through user input and the scores are based on the total sum of the four factors chosen by random (for the sake of simplicity for a simple simulator). The only difference from the previous assignment is the use of the Red-Black Tree. User input is used to determine which function of the Red-Black Tree they want to use as a means of manipulating data through the use of a scanner and switches; this is done by choosing from the menu and selecting the objects by inputting the number that is associated to the task the user wants to do. All the information is used via a web crawler that collects; the web crawler works using an external jar file called jsoup to collect URLs and store them into a data structure for future use. Ranking is assigned based on how the Link types and Node types are sorted; the higher the score, the higher the rank.

To verify that the program is running properly, I used this link (https://www.cs.usfca.edu/~galles/visualization/RedBlack.html) to draw out the RB Tree. This is possible because the nodes are inputted into the tree in ascending order, so there isn't a problem with knowing which order the nodes are put into and not worry about whether the RB tree itself will be different each time.

The assignment overall is about showing three algorithms that were taught in class prior to the assignment as a way to show how each one either sorts or manipulates data, while also showing how these algorithms can be used in real life applications. The assignment shows how a dynamic set can be improved on by implementing methods that can help it become "balanced" and maintain its time complexity.

The Classes and Their Methods

- Link Class: this class is made of five parameters that will be given for each link found by the webcrawler class. The five parameters include the name as a string and the factors - the frequency of the name on the page, time the webpage existed, the amount of associated webpages, and the amount paid by the page to increase advertisement - as ints. Two variables also exist: the index and the score. The score itself is the sum of all int parameters, and can be recalculated when one of those parameters change value. The index is an int used to help find and set the index of the Link at a faster time and for the sake of the simulator also help with debugging if needed. This class has similar implementations as the previous assignment classes.
    - Variables:
        - private String l: the name of the link
        - private int f: the score for frequency of Link on the page
        - private int t: the score for time the page existed
        - private int o: the score for the amount of other pages that reference the same Link
        - private int p: the score for the amount of money paid by the page to increase their placement
        - private int score: the sum of all the factors
        - private int index: the index of where the Link is in the arraylist. Initialized at 0.
        - Private int rank: the rank of the Link. Initialized at 0.

- Methods:
  - Constructor: public Link(String l, int f, int t, int o, int p) {}

    The constructor of the link found by webcrawler. Initializes parameters and sets score to the sum of parameters
  - Get methods (public String getL(), public int getScore(), public int getF() [frequency], public int getT() [time on internet], public int getO() [other website associated with it], public int getP() [payment for exposure], public int getIndex())

    Simple get methods of the class to retrieve values needed.
  - Set methods (public void setL(String theL), public void setF(int aF) [frequency], public void setT(int aT) [time on internet], public void setO(int aO) [other website associated with it], public void setP(int aP) [payment for exposure], public void setIndex(int aIndex))

    Simple set methods of the class to change data.
  - public void calculateScore(int af, int at, int ao, int ap)

    Calculates the score given the factors.

- <u>RedBlackTree Class</u>: This class serves to manipulate the list of links using methods of the binary search tree data structure. The code is based on the textbook's algorithm and altered to manipulate the Links based on their scores. Because RBT is easier to implement using nodes, a node class is created to help manipulate data.

- Node class: The node class is based on the Link class and is used by the RBT class. Like the Link class, the Node class has similar get and set methods along with another original method that converts the Link into the node class.Unlike the original RBT class, an additional parameter
    - Variables:
        - public int key: holds the score of Link
        - public String URL: holds the URL string of Link
        - public Node left, right, p: pointers of the node (left, right, p)
        - public int index = 0: index of the node, initialized to 0;
    - Methods
        - Constructor: public Node(String URL, int score)

          The constructor of the node based on Link. Because it is based on score, the parameters of the Link Type is not needed.
        - Get methods (public String getURL() , public int getKey(), getIndex(), public getRank())

          simple get methods of the class to retrieve values needed.
        - Set methods: (public void setURL(String u), public void setKey(int k), public void setIndex(int aIndex), public void setRank(int r))

          simple set methods of the class to change data.
        - public static Node LinkToNode(String word, int s)

          This is used to convert the Link type into the Node type so it can be used for the RBT.

- RBT Variables:
  - private Node root: the first node. The root of the RBT. Initialized as null to represent an empty RBT.
- RBT Methods
  - Constructor: public RedBlackTree():

    Sets the root to null.
  - public static Node Search(Node x, int k):

    Recursively searches for the node of the given score. Starts the search at root.
  - public static Node Iterative_Search(Node x, int k):

    Iteratively searches for the node of a the given score. Stars the search at root and does not use recursions.
  - public static Node Minimum(Node x):

    Returns the node with the smallest key in the RBT. Starts the search at the root.
  - public static Node Maximum(Node x):

    Returns the node with the biggest key in the RBT. Starts the search at the root.
  - public static Node Successor(Node x):

    Finds the successor of the given node (the node that contains the next biggest key after the provided node).

- public static void Transplant(RedBlackTree B, Node u, Node v): Changes the location of the two given nodes in the given tree by swapping the positions of the two nodes. In addition to the regular implementation of BST, the method is slightly altered to fit the RBT properties.
- public static void Insert(RedBlackTree B, Node z):
  Inserts the given node into the given tree while maintaining RBT properties. In addition to the regular implementation of BST, an additional method is used to ensure that RBT properties are maintained.
- public static void Delete(RedBlackTree B, Node z):
  Deletes the given node from the given tree while maintaining RBT properties. In addition to the regular implementation of BST, and additional method is used to ensure that RBT properties are maintained.
- public static void InsertFixup(RedBlackTree B, Node z):
  Fixes the RBT after insertion so that the tree maintains RBT properties
- public static void DeleteFixup(RedBlackTree B, Node z):
  Fixes the RBT after deletion so that the tree maintains RBT properties
- Public static void leftRotate(RedBlackTree B, Node z):
  Rotates the node and the structure around it towards the left.
- Public static void rightRotate(RedBlackTree B, Node z):
  Rotates the node and the structure around it towards the right.
- public static void inOrder(Node x):
  Returns the list of Nodes sorted in order.

- Web crawler classes: The webcrawler class is the same web crawler used in the last assignment with a few altercations (changed the URL to something more specific and does not take Link input so that it can really scrape the internet for any possible URLs)
    - Spider class: the class is used to find the pages and store them into a needed data structure so that it can be used and manipulated. The class works with the SpiderLeg class to find and collect the needed pages given a URL to scrape.
        - Variables:
            - private static final int MAX_PAGES_TO_SEARCH = 30: puts a max on how many pages to retrieve to avoid infinite loop.
            - private Set<String> pagesVisited = new HashSet<String>(): used to store the pages that have been visited and will be returned when needed
            - private List<String> pagesToVisit = new LinkedList<String>(): Used to keep track of how many pages need to be found.
        - Methods:
            - Get methods (public Set<String> getPages(), public ArrayList<String> getList()):
              Simple get methods of the class to retrieve values needed.
            - public void search(String url):
              Searches for the URLs using the SpiderLeg class
            - private String nextUrl()

Retrieves the next URL to be searched and helps keep track of the URLs.

- SpiderLeg class: The SpiderLeg class does most of the work in searching and scraping the internet for URL links and is used by the Spider class. To use the class, an external jar called jsoup is imported. In order to work, a fake robot is used (provided by the original code) to look through the internet. No significant change is made from the last programming assignment.
    - Variables:
        - private static final String USER_AGENT = "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.112 Safari/535.1": The robot used so that any web browser listed (ex. Mozilla, Chrome, Safari) can be used.
        - private List<String> links = new LinkedList<String>(): the list of links found
        - private Document htmlDocument: the HTML document that will be looked into.
    - Methods:
        - public boolean crawl(String url):
          This is where most of the work is done. Using try and catch, this method asks for HTTP request and then stores the links that
        - public List<String> getLinks(): returns the links that have been found by the crawl method

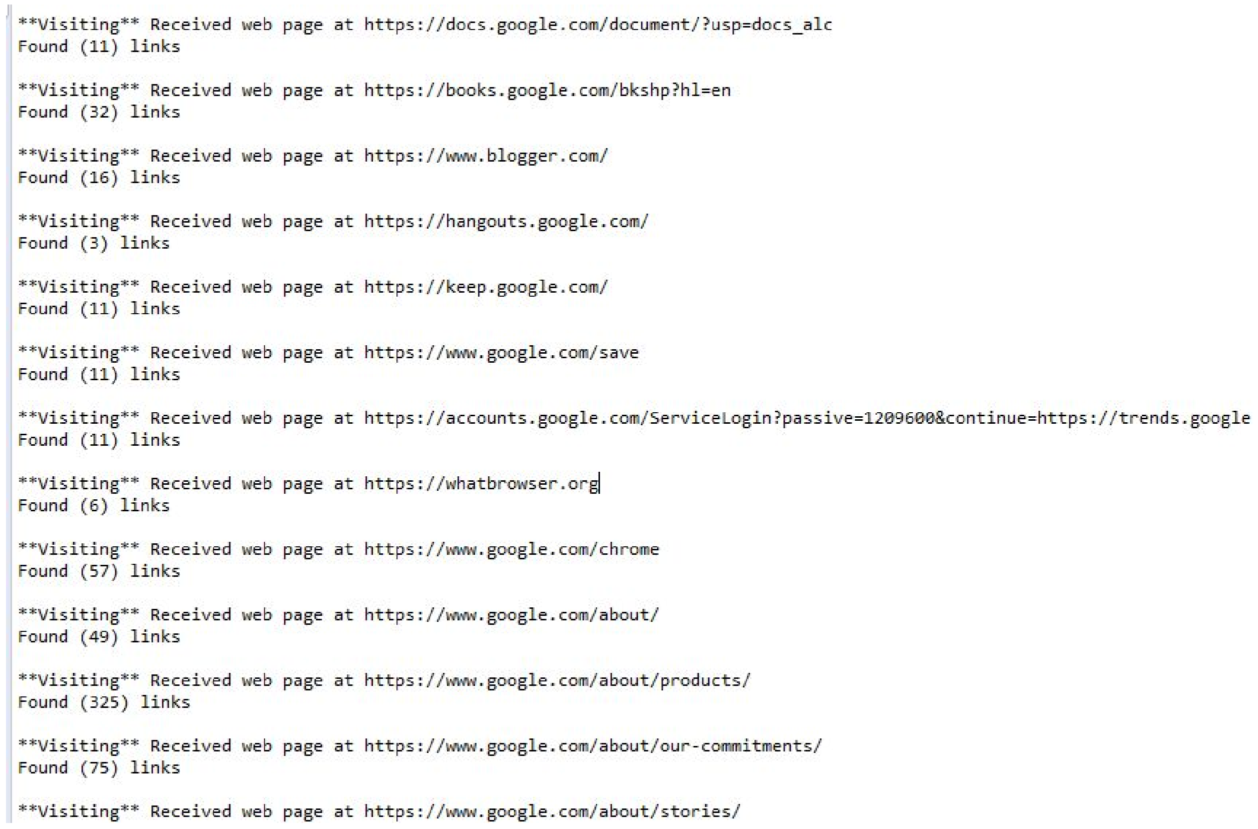
- Sorting classes (QuicksortForRank and InsertionSortForRank): The sorting classes are used to assign the rank to the Link type and the Node type; the sorting also makes it easier to keep track of what is being inserted into the RB Tree. Two sorting algorithms are used: quicksort and insertion sort. Both algorithms are ones that have been used in previous assignments and altered slightly to fit the needs of the program (Using an ArrayList instead of an array and comparing scores/keys rather than numbers). The reason for two algorithms is to make it efficient. Quicksort is only used for inputting the links found by the webcrawler, since it's more likely that the links are assigned scores and placed into the arraylist by random; this will make things efficient. Insertion sort, however, is used when re-assigning nodes when converting from Link to Node and after insertion/deletion. This is because insertion sort runs best case when it is nearly or already sorted; with the quicksort already done, insertion sort can be used to assign the new nodes' ranks, and insertion/deletion only deals with one node at a time. Since these algorithms have been used before, I won't go into details about the methods of both classes except for two: static void assignRankQ(ArrayList<Link> k) and static void assignRankI(ArrayList<RedBlackTree.Node> k). Both serve the same function of setting the rank for their respective types.
  - Methods:
    - static void assignRankQ(ArrayList<Link> k: assigns the rank of the Link type by going through the ArrayList in descending order.
    - static void assignRankI(ArrayList<RedBlackTree.Node> k): assigns the rank of the Node type by going through the ArrayList in descending order.

- Google Simulator Class: The simulator class is where the user can perform the assignment requirements and is supposedly based off the previous assignment's own simulator. The simulator class takes in user input via a scanner and uses switches to take in the scanner input to perform the function needed. With RBT, the simulator asks the user what sort of data manipulation they would like to do. Depending on their choice, they will be asked to input more data or the simulator will simply return what they are asking for. Prior to this, the webcrawler will run and the simulator will assign the URLs as type Link, using the URL as a string, and assigning random numbers for each factor so they could be calculated.

  The cases print the initial order of the tree as a way to debug and also show that the method works. It runs on an infinite loop so all options can be tested.

Screenshot Results

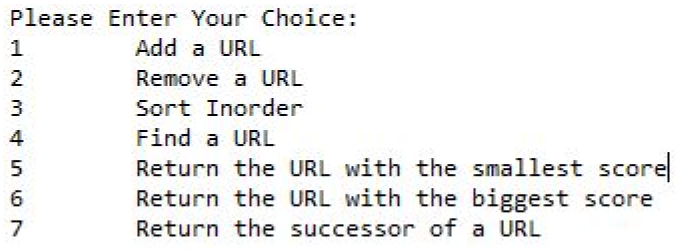
Beginning of simulator (Web crawler working and starting menu):



```
**Visiting** Received web page at https://docs.google.com/document/?usp=docs_alc
Found (11) links

**Visiting** Received web page at https://books.google.com/bkshp?hl=en
Found (32) links

**Visiting** Received web page at https://www.blogger.com/
Found (16) links

**Visiting** Received web page at https://hangouts.google.com/
Found (3) links

**Visiting** Received web page at https://keep.google.com/
Found (11) links

**Visiting** Received web page at https://www.google.com/save
Found (11) links

**Visiting** Received web page at https://accounts.google.com/ServiceLogin?passive=1209600&continue=https://trends.google
Found (11) links

**Visiting** Received web page at https://whatbrowser.org
Found (6) links

**Visiting** Received web page at https://www.google.com/chrome
Found (57) links

**Visiting** Received web page at https://www.google.com/about/
Found (49) links

**Visiting** Received web page at https://www.google.com/about/products/
Found (325) links

**Visiting** Received web page at https://www.google.com/about/our-commitments/
Found (75) links

**Visiting** Received web page at https://www.google.com/about/stories/
```

RBT:

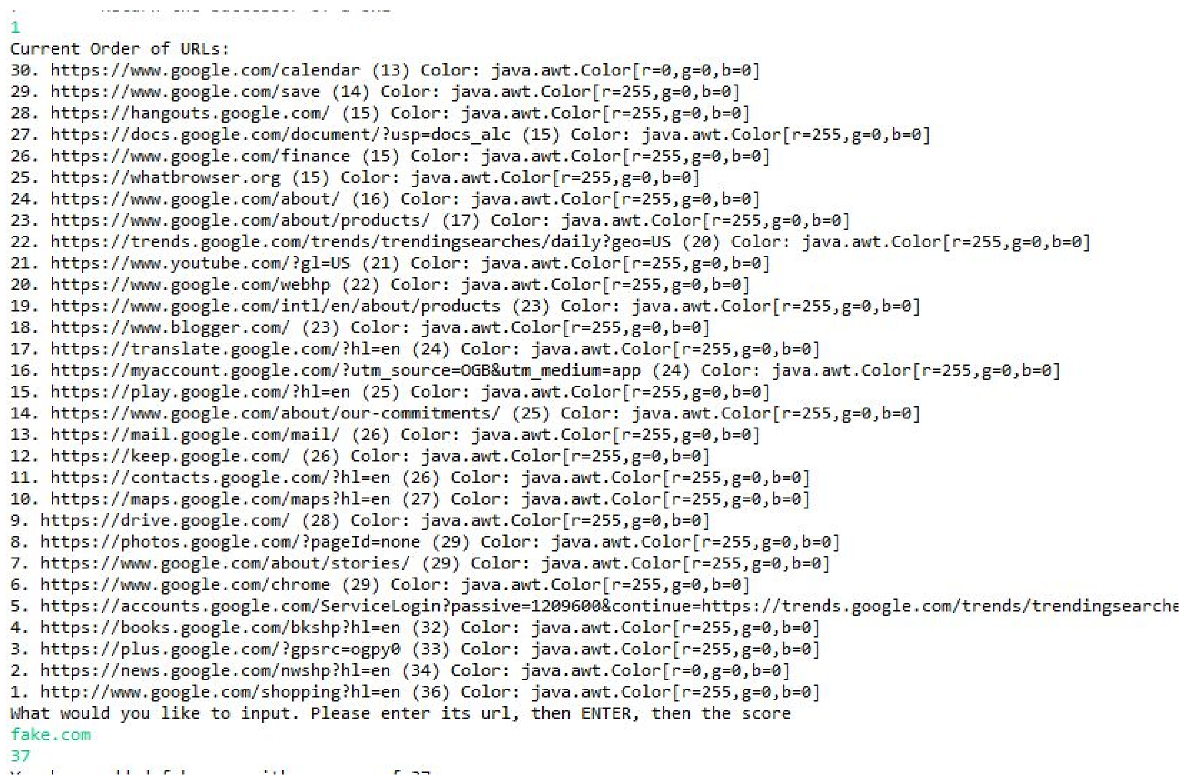
RBT menu



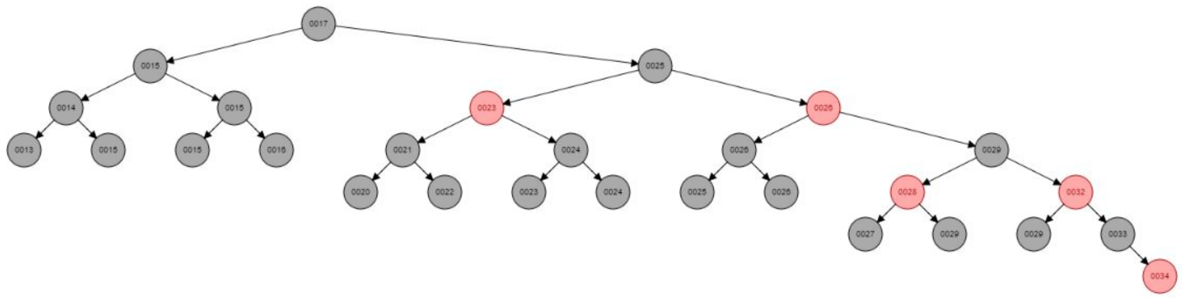
```
Please Enter Your Choice:
1        Add a URL
2        Remove a URL
3        Sort Inorder
4        Find a URL
5        Return the URL with the smallest score
6        Return the URL with the biggest score
7        Return the successor of a URL
```

Insert a URL(fake.com 37) (#5 /#6 has a score of 31 and color RED / RED)

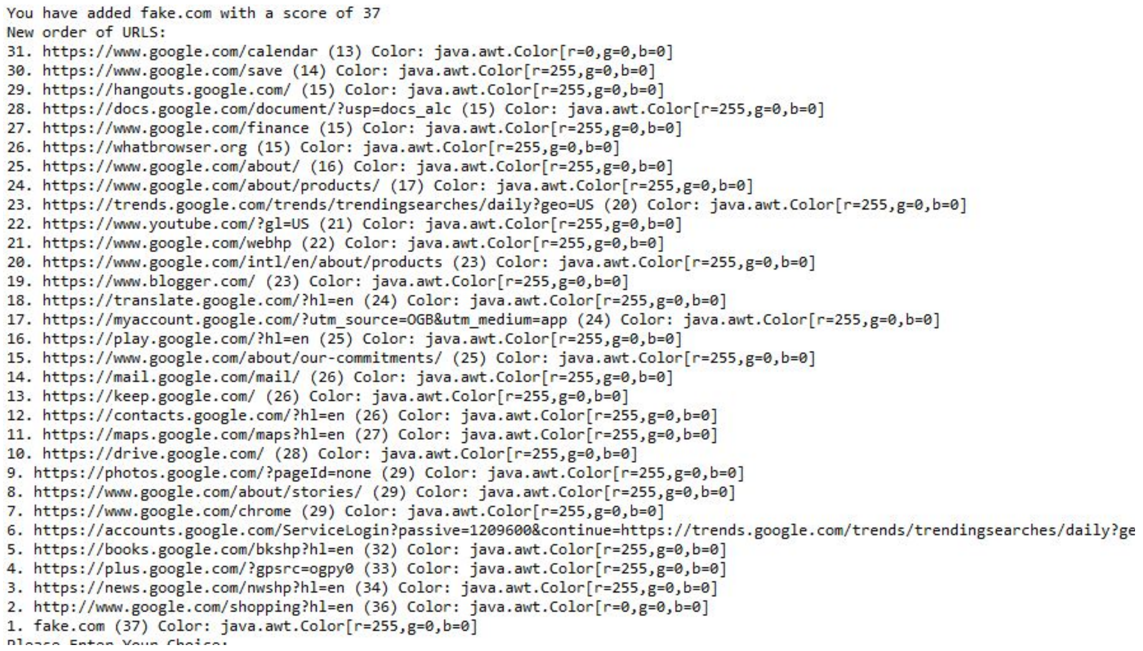
- Before:



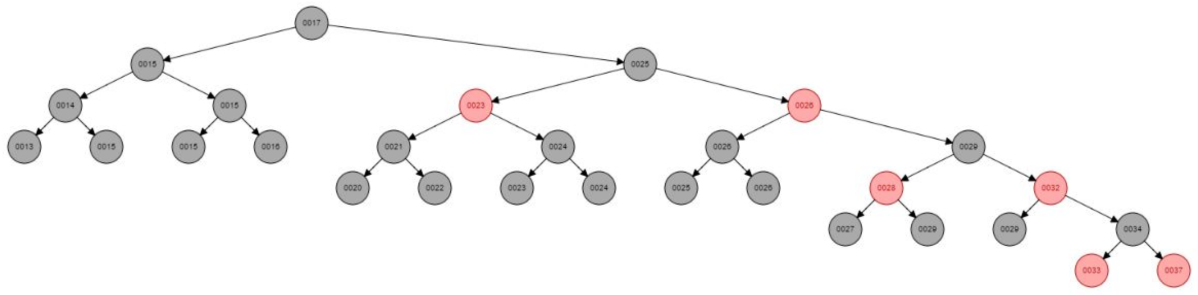
```
1
Current Order of URLs:
30. https://www.google.com/calendar (13) Color: java.awt.Color[r=0,g=0,b=0]
29. https://www.google.com/save (14) Color: java.awt.Color[r=255,g=0,b=0]
28. https://hangouts.google.com/ (15) Color: java.awt.Color[r=255,g=0,b=0]
27. https://docs.google.com/document/?usp=docs_alc (15) Color: java.awt.Color[r=255,g=0,b=0]
26. https://www.google.com/finance (15) Color: java.awt.Color[r=255,g=0,b=0]
25. https://whatbrowser.org (15) Color: java.awt.Color[r=255,g=0,b=0]
24. https://www.google.com/about/ (16) Color: java.awt.Color[r=255,g=0,b=0]
23. https://www.google.com/about/products/ (17) Color: java.awt.Color[r=255,g=0,b=0]
22. https://trends.google.com/trends/trendingsearches/daily?geo=US (20) Color: java.awt.Color[r=255,g=0,b=0]
21. https://www.youtube.com/?gl=US (21) Color: java.awt.Color[r=255,g=0,b=0]
20. https://www.google.com/webhp (22) Color: java.awt.Color[r=255,g=0,b=0]
19. https://www.google.com/intl/en/about/products (23) Color: java.awt.Color[r=255,g=0,b=0]
18. https://www.blogger.com/ (23) Color: java.awt.Color[r=255,g=0,b=0]
17. https://translate.google.com/?hl=en (24) Color: java.awt.Color[r=255,g=0,b=0]
16. https://myaccount.google.com/?utm_source=OGB&utm_medium=app (24) Color: java.awt.Color[r=255,g=0,b=0]
15. https://play.google.com/?hl=en (25) Color: java.awt.Color[r=255,g=0,b=0]
14. https://www.google.com/about/our-commitments/ (25) Color: java.awt.Color[r=255,g=0,b=0]
13. https://mail.google.com/mail/ (26) Color: java.awt.Color[r=255,g=0,b=0]
12. https://keep.google.com/ (26) Color: java.awt.Color[r=255,g=0,b=0]
11. https://contacts.google.com/?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
10. https://maps.google.com/maps?hl=en (27) Color: java.awt.Color[r=255,g=0,b=0]
9. https://drive.google.com/ (28) Color: java.awt.Color[r=255,g=0,b=0]
8. https://photos.google.com/?pageId=none (29) Color: java.awt.Color[r=255,g=0,b=0]
7. https://www.google.com/about/stories/ (29) Color: java.awt.Color[r=255,g=0,b=0]
6. https://www.google.com/chrome (29) Color: java.awt.Color[r=255,g=0,b=0]
5. https://accounts.google.com/ServiceLogin?passive=1209600&continue=https://trends.google.com/trends/trendingsearche
4. https://books.google.com/bkshp?hl=en (32) Color: java.awt.Color[r=255,g=0,b=0]
3. https://plus.google.com/?gpsrc=ogpy0 (33) Color: java.awt.Color[r=255,g=0,b=0]
2. https://news.google.com/nwshp?hl=en (34) Color: java.awt.Color[r=0,g=0,b=0]
1. http://www.google.com/shopping?hl=en (36) Color: java.awt.Color[r=255,g=0,b=0]
What would you like to input. Please enter its url, then ENTER, then the score
fake.com
37
```



- After:

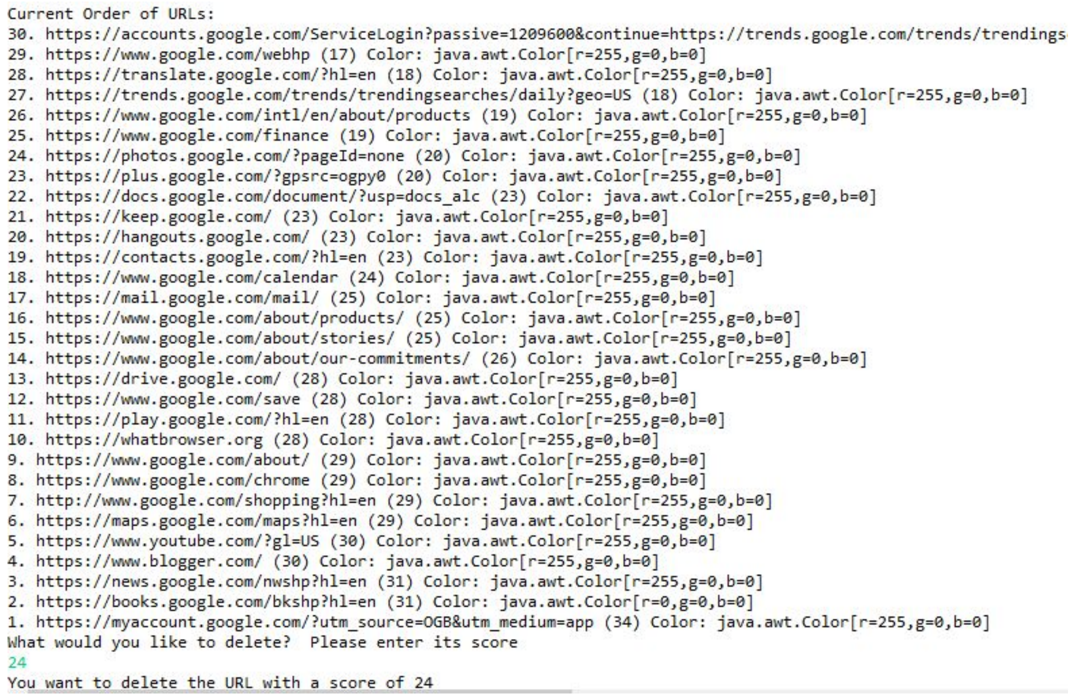


```
You have added fake.com with a score of 37
New order of URLS:
31. https://www.google.com/calendar (13) Color: java.awt.Color[r=0,g=0,b=0]
30. https://www.google.com/save (14) Color: java.awt.Color[r=255,g=0,b=0]
29. https://hangouts.google.com/ (15) Color: java.awt.Color[r=255,g=0,b=0]
28. https://docs.google.com/document/?usp=docs_alc (15) Color: java.awt.Color[r=255,g=0,b=0]
27. https://www.google.com/finance (15) Color: java.awt.Color[r=255,g=0,b=0]
26. https://whatbrowser.org (15) Color: java.awt.Color[r=255,g=0,b=0]
25. https://www.google.com/about/ (16) Color: java.awt.Color[r=255,g=0,b=0]
24. https://www.google.com/about/products/ (17) Color: java.awt.Color[r=255,g=0,b=0]
23. https://trends.google.com/trends/trendingsearches/daily?geo=US (20) Color: java.awt.Color[r=255,g=0,b=0]
22. https://www.youtube.com/?gl=US (21) Color: java.awt.Color[r=255,g=0,b=0]
21. https://www.google.com/webhp (22) Color: java.awt.Color[r=255,g=0,b=0]
20. https://www.google.com/intl/en/about/products (23) Color: java.awt.Color[r=255,g=0,b=0]
19. https://www.blogger.com/ (23) Color: java.awt.Color[r=255,g=0,b=0]
18. https://translate.google.com/?hl=en (24) Color: java.awt.Color[r=255,g=0,b=0]
17. https://myaccount.google.com/?utm_source=OGB&utm_medium=app (24) Color: java.awt.Color[r=255,g=0,b=0]
16. https://play.google.com/?hl=en (25) Color: java.awt.Color[r=255,g=0,b=0]
15. https://www.google.com/about/our-commitments/ (25) Color: java.awt.Color[r=255,g=0,b=0]
14. https://mail.google.com/mail/ (26) Color: java.awt.Color[r=255,g=0,b=0]
13. https://keep.google.com/ (26) Color: java.awt.Color[r=255,g=0,b=0]
12. https://contacts.google.com/?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
11. https://maps.google.com/maps?hl=en (27) Color: java.awt.Color[r=255,g=0,b=0]
10. https://drive.google.com/ (28) Color: java.awt.Color[r=255,g=0,b=0]
9. https://photos.google.com/?pageId=none (29) Color: java.awt.Color[r=255,g=0,b=0]
8. https://www.google.com/about/stories/ (29) Color: java.awt.Color[r=255,g=0,b=0]
7. https://www.google.com/chrome (29) Color: java.awt.Color[r=255,g=0,b=0]
6. https://accounts.google.com/ServiceLogin?passive=1209600&continue=https://trends.google.com/trends/trendingsearches/daily?ge
5. https://books.google.com/bkshp?hl=en (32) Color: java.awt.Color[r=255,g=0,b=0]
4. https://plus.google.com/?gpsrc=ogpy0 (33) Color: java.awt.Color[r=255,g=0,b=0]
3. https://news.google.com/nwshp?hl=en (34) Color: java.awt.Color[r=255,g=0,b=0]
2. http://www.google.com/shopping?hl=en (36) Color: java.awt.Color[r=0,g=0,b=0]
1. fake.com (37) Color: java.awt.Color[r=255,g=0,b=0]
```

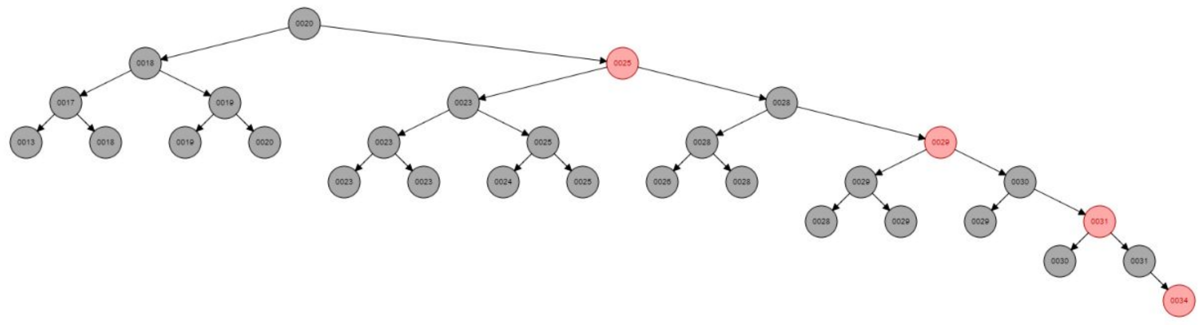


Remove a URL by score (Score: 24)

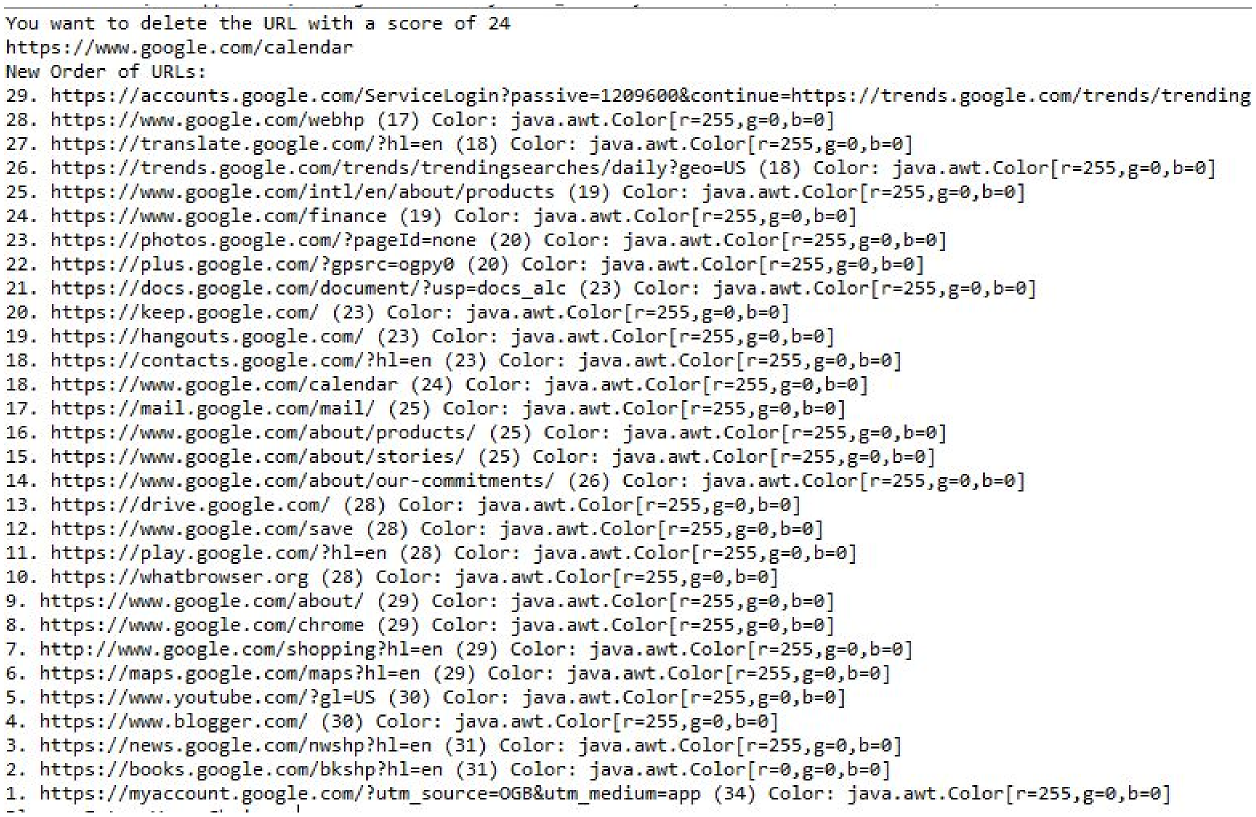
- Before:



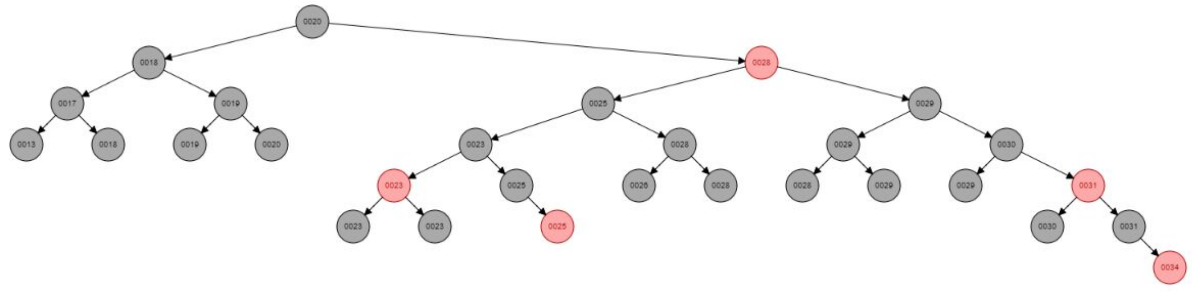
```
Current Order of URLs:
30. https://accounts.google.com/ServiceLogin?passive=1209600&continue=https://trends.google.com/trends/trendings
29. https://www.google.com/webhp (17) Color: java.awt.Color[r=255,g=0,b=0]
28. https://translate.google.com/?hl=en (18) Color: java.awt.Color[r=255,g=0,b=0]
27. https://trends.google.com/trends/trendingsearches/daily?geo=US (18) Color: java.awt.Color[r=255,g=0,b=0]
26. https://www.google.com/intl/en/about/products (19) Color: java.awt.Color[r=255,g=0,b=0]
25. https://www.google.com/finance (19) Color: java.awt.Color[r=255,g=0,b=0]
24. https://photos.google.com/?pageId=none (20) Color: java.awt.Color[r=255,g=0,b=0]
23. https://plus.google.com/?gpsrc=ogpy0 (20) Color: java.awt.Color[r=255,g=0,b=0]
22. https://docs.google.com/document/?usp=docs_alc (23) Color: java.awt.Color[r=255,g=0,b=0]
21. https://keep.google.com/ (23) Color: java.awt.Color[r=255,g=0,b=0]
20. https://hangouts.google.com/ (23) Color: java.awt.Color[r=255,g=0,b=0]
19. https://contacts.google.com/?hl=en (23) Color: java.awt.Color[r=255,g=0,b=0]
18. https://www.google.com/calendar (24) Color: java.awt.Color[r=255,g=0,b=0]
17. https://mail.google.com/mail/ (25) Color: java.awt.Color[r=255,g=0,b=0]
16. https://www.google.com/about/products/ (25) Color: java.awt.Color[r=255,g=0,b=0]
15. https://www.google.com/about/stories/ (25) Color: java.awt.Color[r=255,g=0,b=0]
14. https://www.google.com/about/our-commitments/ (26) Color: java.awt.Color[r=255,g=0,b=0]
13. https://drive.google.com/ (28) Color: java.awt.Color[r=255,g=0,b=0]
12. https://www.google.com/save (28) Color: java.awt.Color[r=255,g=0,b=0]
11. https://play.google.com/?hl=en (28) Color: java.awt.Color[r=255,g=0,b=0]
10. https://whatbrowser.org (28) Color: java.awt.Color[r=255,g=0,b=0]
9. https://www.google.com/about/ (29) Color: java.awt.Color[r=255,g=0,b=0]
8. https://www.google.com/chrome (29) Color: java.awt.Color[r=255,g=0,b=0]
7. http://www.google.com/shopping?hl=en (29) Color: java.awt.Color[r=255,g=0,b=0]
6. https://maps.google.com/maps?hl=en (29) Color: java.awt.Color[r=255,g=0,b=0]
5. https://www.youtube.com/?gl=US (30) Color: java.awt.Color[r=255,g=0,b=0]
4. https://www.blogger.com/ (30) Color: java.awt.Color[r=255,g=0,b=0]
3. https://news.google.com/nwshp?hl=en (31) Color: java.awt.Color[r=255,g=0,b=0]
2. https://books.google.com/bkshp?hl=en (31) Color: java.awt.Color[r=0,g=0,b=0]
1. https://myaccount.google.com/?utm_source=OGB&utm_medium=app (34) Color: java.awt.Color[r=255,g=0,b=0]
What would you like to delete?  Please enter its score
24
You want to delete the URL with a score of 24
```



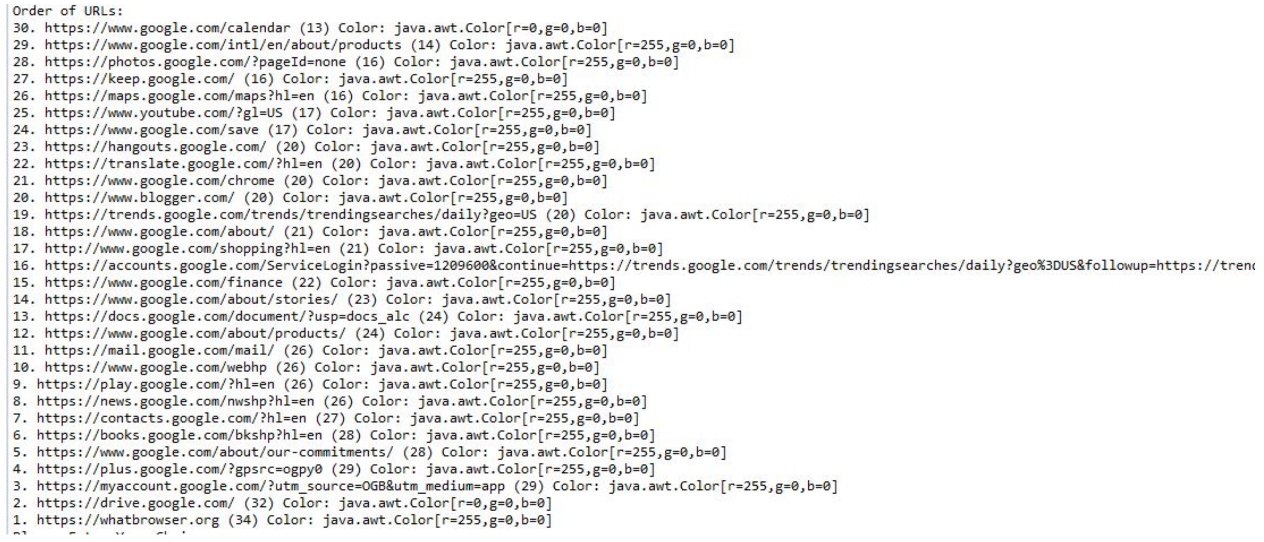
- After:



```
You want to delete the URL with a score of 24
https://www.google.com/calendar
New Order of URLs:
29. https://accounts.google.com/ServiceLogin?passive=1209600&continue=https://trends.google.com/trends/trending
28. https://www.google.com/webhp (17) Color: java.awt.Color[r=255,g=0,b=0]
27. https://translate.google.com/?hl=en (18) Color: java.awt.Color[r=255,g=0,b=0]
26. https://trends.google.com/trends/trendingsearches/daily?geo=US (18) Color: java.awt.Color[r=255,g=0,b=0]
25. https://www.google.com/intl/en/about/products (19) Color: java.awt.Color[r=255,g=0,b=0]
24. https://www.google.com/finance (19) Color: java.awt.Color[r=255,g=0,b=0]
23. https://photos.google.com/?pageId=none (20) Color: java.awt.Color[r=255,g=0,b=0]
22. https://plus.google.com/?gpsrc=ogpy0 (20) Color: java.awt.Color[r=255,g=0,b=0]
21. https://docs.google.com/document/?usp=docs_alc (23) Color: java.awt.Color[r=255,g=0,b=0]
20. https://keep.google.com/ (23) Color: java.awt.Color[r=255,g=0,b=0]
19. https://hangouts.google.com/ (23) Color: java.awt.Color[r=255,g=0,b=0]
18. https://contacts.google.com/?hl=en (23) Color: java.awt.Color[r=255,g=0,b=0]
18. https://www.google.com/calendar (24) Color: java.awt.Color[r=255,g=0,b=0]
17. https://mail.google.com/mail/ (25) Color: java.awt.Color[r=255,g=0,b=0]
16. https://www.google.com/about/products/ (25) Color: java.awt.Color[r=255,g=0,b=0]
15. https://www.google.com/about/stories/ (25) Color: java.awt.Color[r=255,g=0,b=0]
14. https://www.google.com/about/our-commitments/ (26) Color: java.awt.Color[r=255,g=0,b=0]
13. https://drive.google.com/ (28) Color: java.awt.Color[r=255,g=0,b=0]
12. https://www.google.com/save (28) Color: java.awt.Color[r=255,g=0,b=0]
11. https://play.google.com/?hl=en (28) Color: java.awt.Color[r=255,g=0,b=0]
10. https://whatbrowser.org (28) Color: java.awt.Color[r=255,g=0,b=0]
9. https://www.google.com/about/ (29) Color: java.awt.Color[r=255,g=0,b=0]
8. https://www.google.com/chrome (29) Color: java.awt.Color[r=255,g=0,b=0]
7. http://www.google.com/shopping?hl=en (29) Color: java.awt.Color[r=255,g=0,b=0]
6. https://maps.google.com/maps?hl=en (29) Color: java.awt.Color[r=255,g=0,b=0]
5. https://www.youtube.com/?gl=US (30) Color: java.awt.Color[r=255,g=0,b=0]
4. https://www.blogger.com/ (30) Color: java.awt.Color[r=255,g=0,b=0]
3. https://news.google.com/nwshp?hl=en (31) Color: java.awt.Color[r=255,g=0,b=0]
2. https://books.google.com/bkshp?hl=en (31) Color: java.awt.Color[r=0,g=0,b=0]
1. https://myaccount.google.com/?utm_source=OGB&utm_medium=app (34) Color: java.awt.Color[r=255,g=0,b=0]
```

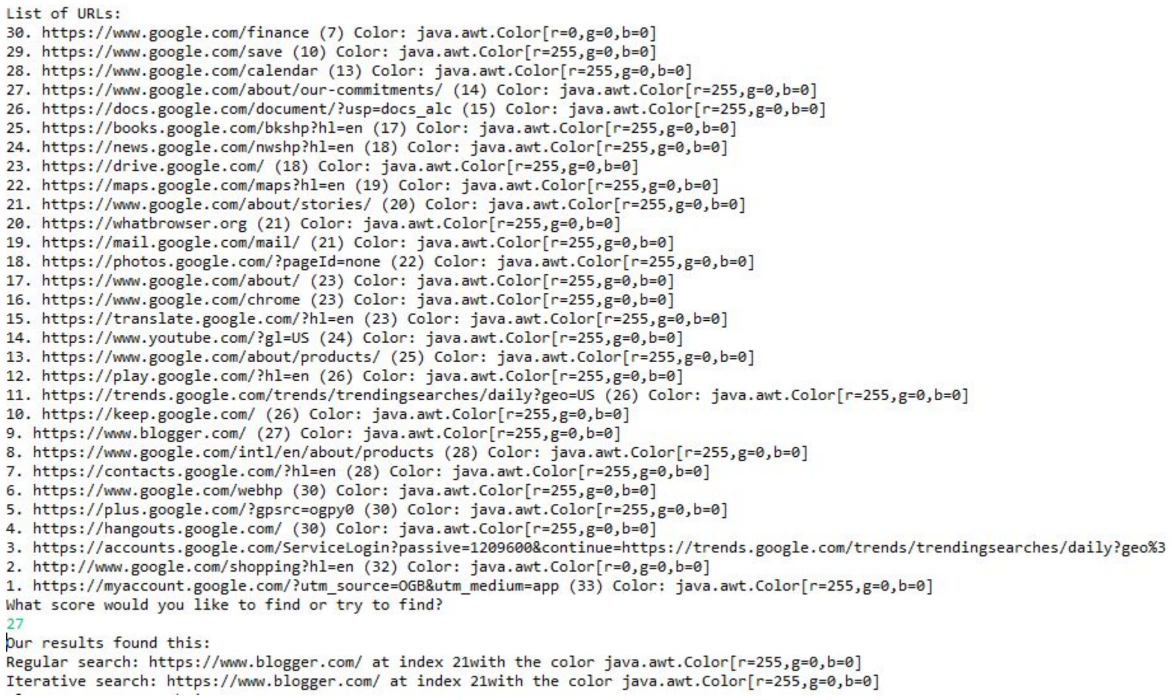


In order sort: (# 16 has a key of 22 and color RED)

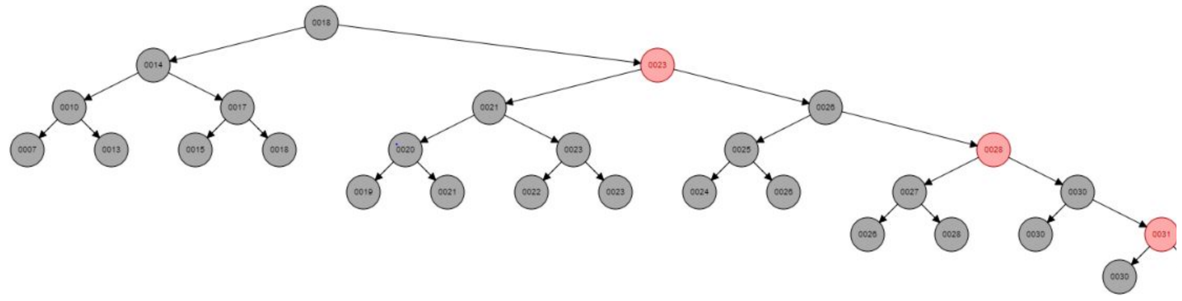


```
Order of URLs:
30. https://www.google.com/calendar (13) Color: java.awt.Color[r=0,g=0,b=0]
29. https://www.google.com/intl/en/about/products (14) Color: java.awt.Color[r=255,g=0,b=0]
28. https://photos.google.com/?pageId=none (16) Color: java.awt.Color[r=255,g=0,b=0]
27. https://keep.google.com/ (16) Color: java.awt.Color[r=255,g=0,b=0]
26. https://maps.google.com/maps?hl=en (16) Color: java.awt.Color[r=255,g=0,b=0]
25. https://www.youtube.com/?gl=US (17) Color: java.awt.Color[r=255,g=0,b=0]
24. https://www.google.com/save (17) Color: java.awt.Color[r=255,g=0,b=0]
23. https://hangouts.google.com/ (20) Color: java.awt.Color[r=255,g=0,b=0]
22. https://translate.google.com/?hl=en (20) Color: java.awt.Color[r=255,g=0,b=0]
21. https://www.google.com/chrome (20) Color: java.awt.Color[r=255,g=0,b=0]
20. https://www.blogger.com/ (20) Color: java.awt.Color[r=255,g=0,b=0]
19. https://trends.google.com/trends/trendingsearches/daily?geo=US (20) Color: java.awt.Color[r=255,g=0,b=0]
18. https://www.google.com/about/ (21) Color: java.awt.Color[r=255,g=0,b=0]
17. http://www.google.com/shopping?hl=en (21) Color: java.awt.Color[r=255,g=0,b=0]
16. https://accounts.google.com/ServiceLogin?passive=1209600&continue=https://trends.google.com/trends/trendingsearches/daily?geo%3DUS&followup=https://tren
15. https://www.google.com/finance (22) Color: java.awt.Color[r=255,g=0,b=0]
14. https://www.google.com/about/stories/ (23) Color: java.awt.Color[r=255,g=0,b=0]
13. https://docs.google.com/document/?usp=docs_alc (24) Color: java.awt.Color[r=255,g=0,b=0]
12. https://www.google.com/about/products/ (24) Color: java.awt.Color[r=255,g=0,b=0]
11. https://mail.google.com/mail/ (26) Color: java.awt.Color[r=255,g=0,b=0]
10. https://www.google.com/webhp (26) Color: java.awt.Color[r=255,g=0,b=0]
9. https://play.google.com/?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
8. https://news.google.com/nwshp?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
7. https://contacts.google.com/?hl=en (27) Color: java.awt.Color[r=255,g=0,b=0]
6. https://books.google.com/bkshp?hl=en (28) Color: java.awt.Color[r=255,g=0,b=0]
5. https://www.google.com/about/our-commitments/ (28) Color: java.awt.Color[r=255,g=0,b=0]
4. https://plus.google.com/?gpsrc=ogpy0 (29) Color: java.awt.Color[r=255,g=0,b=0]
3. https://myaccount.google.com/?utm_source=OGB&utm_medium=app (29) Color: java.awt.Color[r=255,g=0,b=0]
2. https://drive.google.com/ (32) Color: java.awt.Color[r=0,g=0,b=0]
1. https://whatbrowser.org (34) Color: java.awt.Color[r=255,g=0,b=0]
```

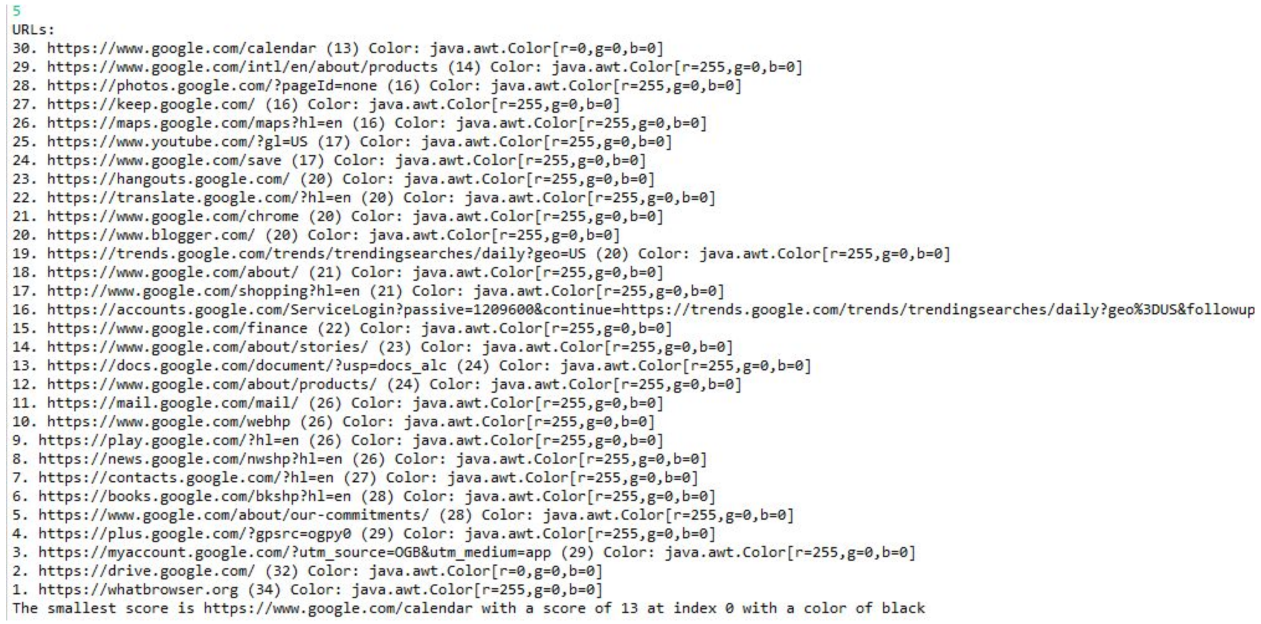
Find URL: (Key: 27)



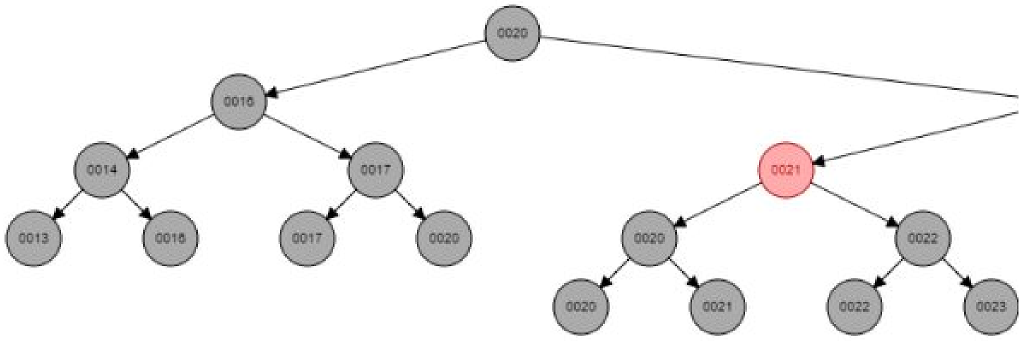
```
List of URLs:
30. https://www.google.com/finance (7) Color: java.awt.Color[r=0,g=0,b=0]
29. https://www.google.com/save (10) Color: java.awt.Color[r=255,g=0,b=0]
28. https://www.google.com/calendar (13) Color: java.awt.Color[r=255,g=0,b=0]
27. https://www.google.com/about/our-commitments/ (14) Color: java.awt.Color[r=255,g=0,b=0]
26. https://docs.google.com/document/?usp=docs_alc (15) Color: java.awt.Color[r=255,g=0,b=0]
25. https://books.google.com/bkshp?hl=en (17) Color: java.awt.Color[r=255,g=0,b=0]
24. https://news.google.com/nwshp?hl=en (18) Color: java.awt.Color[r=255,g=0,b=0]
23. https://drive.google.com/ (18) Color: java.awt.Color[r=255,g=0,b=0]
22. https://maps.google.com/maps?hl=en (19) Color: java.awt.Color[r=255,g=0,b=0]
21. https://www.google.com/about/stories/ (20) Color: java.awt.Color[r=255,g=0,b=0]
20. https://whatbrowser.org (21) Color: java.awt.Color[r=255,g=0,b=0]
19. https://mail.google.com/mail/ (21) Color: java.awt.Color[r=255,g=0,b=0]
18. https://photos.google.com/?pageId=none (22) Color: java.awt.Color[r=255,g=0,b=0]
17. https://www.google.com/about/ (23) Color: java.awt.Color[r=255,g=0,b=0]
16. https://www.google.com/chrome (23) Color: java.awt.Color[r=255,g=0,b=0]
15. https://translate.google.com/?hl=en (23) Color: java.awt.Color[r=255,g=0,b=0]
14. https://www.youtube.com/?gl=US (24) Color: java.awt.Color[r=255,g=0,b=0]
13. https://www.google.com/about/products/ (25) Color: java.awt.Color[r=255,g=0,b=0]
12. https://play.google.com/?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
11. https://trends.google.com/trends/trendingsearches/daily?geo=US (26) Color: java.awt.Color[r=255,g=0,b=0]
10. https://keep.google.com/ (26) Color: java.awt.Color[r=255,g=0,b=0]
9. https://www.blogger.com/ (27) Color: java.awt.Color[r=255,g=0,b=0]
8. https://www.google.com/intl/en/about/products (28) Color: java.awt.Color[r=255,g=0,b=0]
7. https://contacts.google.com/?hl=en (28) Color: java.awt.Color[r=255,g=0,b=0]
6. https://www.google.com/webhp (30) Color: java.awt.Color[r=255,g=0,b=0]
5. https://plus.google.com/?gpsrc=ogpy0 (30) Color: java.awt.Color[r=255,g=0,b=0]
4. https://hangouts.google.com/ (30) Color: java.awt.Color[r=255,g=0,b=0]
3. https://accounts.google.com/ServiceLogin?passive=1209600&continue=https://trends.google.com/trends/trendingsearches/daily?geo%3
2. http://www.google.com/shopping?hl=en (32) Color: java.awt.Color[r=0,g=0,b=0]
1. https://myaccount.google.com/?utm_source=OGB&utm_medium=app (33) Color: java.awt.Color[r=255,g=0,b=0]
What score would you like to find or try to find?
27
Our results found this:
Regular search: https://www.blogger.com/ at index 21with the color java.awt.Color[r=255,g=0,b=0]
Iterative search: https://www.blogger.com/ at index 21with the color java.awt.Color[r=255,g=0,b=0]
```



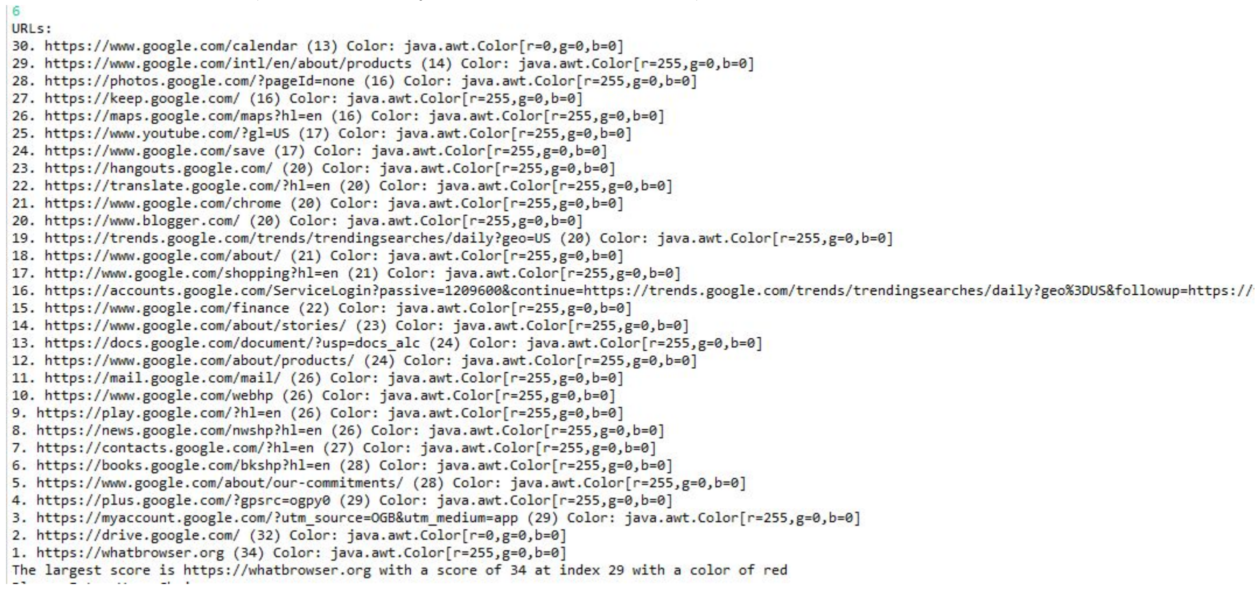
Minimum: (#16 has a key of 22 and color Red)



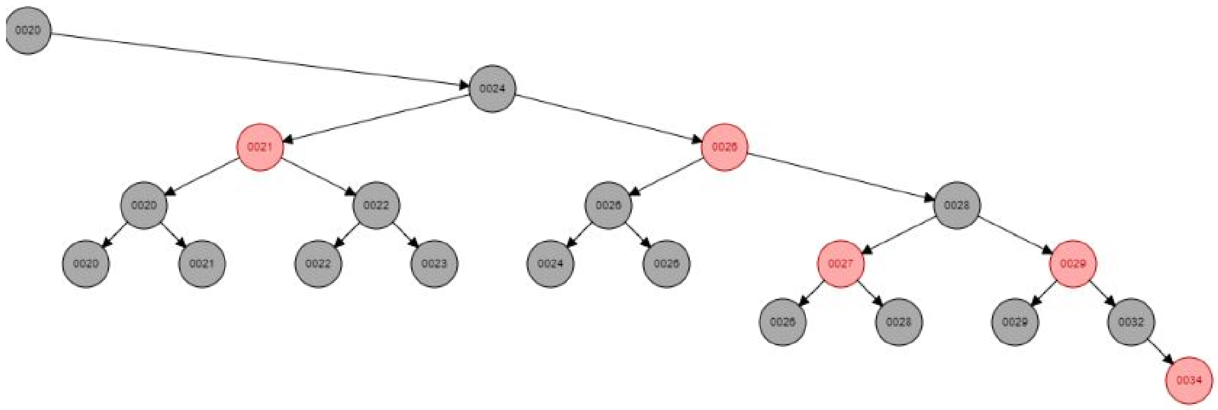
```
5
URLs:
30. https://www.google.com/calendar (13) Color: java.awt.Color[r=0,g=0,b=0]
29. https://www.google.com/intl/en/about/products (14) Color: java.awt.Color[r=255,g=0,b=0]
28. https://photos.google.com/?pageId=none (16) Color: java.awt.Color[r=255,g=0,b=0]
27. https://keep.google.com/ (16) Color: java.awt.Color[r=255,g=0,b=0]
26. https://maps.google.com/maps?hl=en (16) Color: java.awt.Color[r=255,g=0,b=0]
25. https://www.youtube.com/?gl=US (17) Color: java.awt.Color[r=255,g=0,b=0]
24. https://www.google.com/save (17) Color: java.awt.Color[r=255,g=0,b=0]
23. https://hangouts.google.com/ (20) Color: java.awt.Color[r=255,g=0,b=0]
22. https://translate.google.com/?hl=en (20) Color: java.awt.Color[r=255,g=0,b=0]
21. https://www.google.com/chrome (20) Color: java.awt.Color[r=255,g=0,b=0]
20. https://www.blogger.com/ (20) Color: java.awt.Color[r=255,g=0,b=0]
19. https://trends.google.com/trends/trendingsearches/daily?geo=US (20) Color: java.awt.Color[r=255,g=0,b=0]
18. https://www.google.com/about/ (21) Color: java.awt.Color[r=255,g=0,b=0]
17. http://www.google.com/shopping?hl=en (21) Color: java.awt.Color[r=255,g=0,b=0]
16. https://accounts.google.com/ServiceLogin?passive=1209600&continue=https://trends.google.com/trends/trendingsearches/daily?geo%3DUS&followup
15. https://www.google.com/finance (22) Color: java.awt.Color[r=255,g=0,b=0]
14. https://www.google.com/about/stories/ (23) Color: java.awt.Color[r=255,g=0,b=0]
13. https://docs.google.com/document/?usp=docs_alc (24) Color: java.awt.Color[r=255,g=0,b=0]
12. https://www.google.com/about/products/ (24) Color: java.awt.Color[r=255,g=0,b=0]
11. https://mail.google.com/mail/ (26) Color: java.awt.Color[r=255,g=0,b=0]
10. https://www.google.com/webhp (26) Color: java.awt.Color[r=255,g=0,b=0]
9. https://play.google.com/?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
8. https://news.google.com/nwshp?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
7. https://contacts.google.com/?hl=en (27) Color: java.awt.Color[r=255,g=0,b=0]
6. https://books.google.com/bkshp?hl=en (28) Color: java.awt.Color[r=255,g=0,b=0]
5. https://www.google.com/about/our-commitments/ (28) Color: java.awt.Color[r=255,g=0,b=0]
4. https://plus.google.com/?gpsrc=ogpy0 (29) Color: java.awt.Color[r=255,g=0,b=0]
3. https://myaccount.google.com/?utm_source=OGB&utm_medium=app (29) Color: java.awt.Color[r=255,g=0,b=0]
2. https://drive.google.com/ (32) Color: java.awt.Color[r=0,g=0,b=0]
1. https://whatbrowser.org (34) Color: java.awt.Color[r=255,g=0,b=0]
The smallest score is https://www.google.com/calendar with a score of 13 at index 0 with a color of black
```



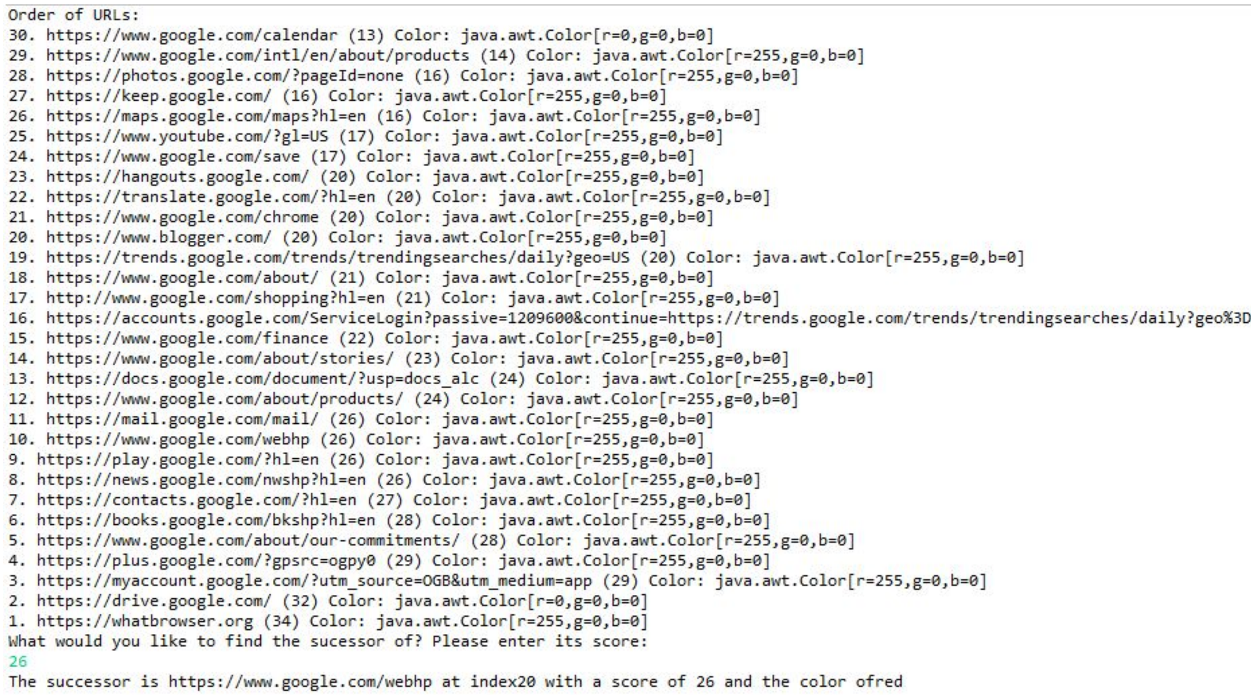
Maximum: (#16 has a key of 22 and color Red)



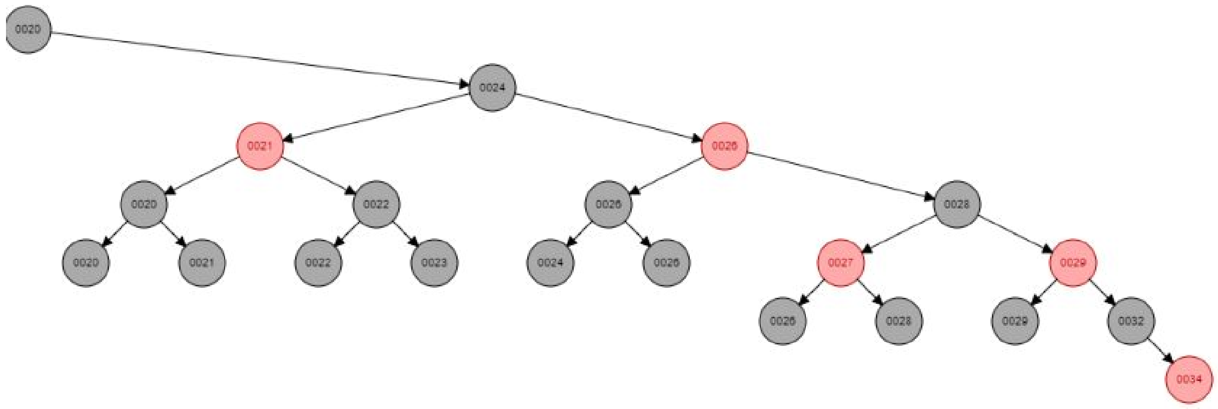
```
6
URLs:
30. https://www.google.com/calendar (13) Color: java.awt.Color[r=0,g=0,b=0]
29. https://www.google.com/intl/en/about/products (14) Color: java.awt.Color[r=255,g=0,b=0]
28. https://photos.google.com/?pageId=none (16) Color: java.awt.Color[r=255,g=0,b=0]
27. https://keep.google.com/ (16) Color: java.awt.Color[r=255,g=0,b=0]
26. https://maps.google.com/maps?hl=en (16) Color: java.awt.Color[r=255,g=0,b=0]
25. https://www.youtube.com/?gl=US (17) Color: java.awt.Color[r=255,g=0,b=0]
24. https://www.google.com/save (17) Color: java.awt.Color[r=255,g=0,b=0]
23. https://hangouts.google.com/ (20) Color: java.awt.Color[r=255,g=0,b=0]
22. https://translate.google.com/?hl=en (20) Color: java.awt.Color[r=255,g=0,b=0]
21. https://www.google.com/chrome (20) Color: java.awt.Color[r=255,g=0,b=0]
20. https://www.blogger.com/ (20) Color: java.awt.Color[r=255,g=0,b=0]
19. https://trends.google.com/trends/trendingsearches/daily?geo=US (20) Color: java.awt.Color[r=255,g=0,b=0]
18. https://www.google.com/about/ (21) Color: java.awt.Color[r=255,g=0,b=0]
17. http://www.google.com/shopping?hl=en (21) Color: java.awt.Color[r=255,g=0,b=0]
16. https://accounts.google.com/ServiceLogin?passive=1209600&continue=https://trends.google.com/trends/trendingsearches/daily?geo%3DUS&followup=https://
15. https://www.google.com/finance (22) Color: java.awt.Color[r=255,g=0,b=0]
14. https://www.google.com/about/stories/ (23) Color: java.awt.Color[r=255,g=0,b=0]
13. https://docs.google.com/document/?usp=docs_alc (24) Color: java.awt.Color[r=255,g=0,b=0]
12. https://www.google.com/about/products/ (24) Color: java.awt.Color[r=255,g=0,b=0]
11. https://mail.google.com/mail/ (26) Color: java.awt.Color[r=255,g=0,b=0]
10. https://www.google.com/webhp (26) Color: java.awt.Color[r=255,g=0,b=0]
9. https://play.google.com/?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
8. https://news.google.com/nwshp?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
7. https://contacts.google.com/?hl=en (27) Color: java.awt.Color[r=255,g=0,b=0]
6. https://books.google.com/bkshp?hl=en (28) Color: java.awt.Color[r=255,g=0,b=0]
5. https://www.google.com/about/our-commitments/ (28) Color: java.awt.Color[r=255,g=0,b=0]
4. https://plus.google.com/?gpsrc=ogpy0 (29) Color: java.awt.Color[r=255,g=0,b=0]
3. https://myaccount.google.com/?utm_source=OGB&utm_medium=app (29) Color: java.awt.Color[r=255,g=0,b=0]
2. https://drive.google.com/ (32) Color: java.awt.Color[r=0,g=0,b=0]
1. https://whatbrowser.org (34) Color: java.awt.Color[r=255,g=0,b=0]
The largest score is https://whatbrowser.org with a score of 34 at index 29 with a color of red
```



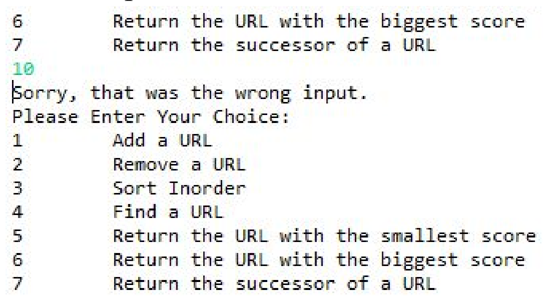
Successor (Find score 26's successor):



```
Order of URLs:
30. https://www.google.com/calendar (13) Color: java.awt.Color[r=0,g=0,b=0]
29. https://www.google.com/intl/en/about/products (14) Color: java.awt.Color[r=255,g=0,b=0]
28. https://photos.google.com/?pageId=none (16) Color: java.awt.Color[r=255,g=0,b=0]
27. https://keep.google.com/ (16) Color: java.awt.Color[r=255,g=0,b=0]
26. https://maps.google.com/maps?hl=en (16) Color: java.awt.Color[r=255,g=0,b=0]
25. https://www.youtube.com/?gl=US (17) Color: java.awt.Color[r=255,g=0,b=0]
24. https://www.google.com/save (17) Color: java.awt.Color[r=255,g=0,b=0]
23. https://hangouts.google.com/ (20) Color: java.awt.Color[r=255,g=0,b=0]
22. https://translate.google.com/?hl=en (20) Color: java.awt.Color[r=255,g=0,b=0]
21. https://www.google.com/chrome (20) Color: java.awt.Color[r=255,g=0,b=0]
20. https://www.blogger.com/ (20) Color: java.awt.Color[r=255,g=0,b=0]
19. https://trends.google.com/trends/trendingsearches/daily?geo=US (20) Color: java.awt.Color[r=255,g=0,b=0]
18. https://www.google.com/about/ (21) Color: java.awt.Color[r=255,g=0,b=0]
17. http://www.google.com/shopping?hl=en (21) Color: java.awt.Color[r=255,g=0,b=0]
16. https://accounts.google.com/ServiceLogin?passive=1209600&continue=https://trends.google.com/trends/trendingsearches/daily?geo%3D
15. https://www.google.com/finance (22) Color: java.awt.Color[r=255,g=0,b=0]
14. https://www.google.com/about/stories/ (23) Color: java.awt.Color[r=255,g=0,b=0]
13. https://docs.google.com/document/?usp=docs_alc (24) Color: java.awt.Color[r=255,g=0,b=0]
12. https://www.google.com/about/products/ (24) Color: java.awt.Color[r=255,g=0,b=0]
11. https://mail.google.com/mail/ (26) Color: java.awt.Color[r=255,g=0,b=0]
10. https://www.google.com/webhp (26) Color: java.awt.Color[r=255,g=0,b=0]
9. https://play.google.com/?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
8. https://news.google.com/nwshp?hl=en (26) Color: java.awt.Color[r=255,g=0,b=0]
7. https://contacts.google.com/?hl=en (27) Color: java.awt.Color[r=255,g=0,b=0]
6. https://books.google.com/bkshp?hl=en (28) Color: java.awt.Color[r=255,g=0,b=0]
5. https://www.google.com/about/our-commitments/ (28) Color: java.awt.Color[r=255,g=0,b=0]
4. https://plus.google.com/?gpsrc=ogpy0 (29) Color: java.awt.Color[r=255,g=0,b=0]
3. https://myaccount.google.com/?utm_source=OGB&utm_medium=app (29) Color: java.awt.Color[r=255,g=0,b=0]
2. https://drive.google.com/ (32) Color: java.awt.Color[r=0,g=0,b=0]
1. https://whatbrowser.org (34) Color: java.awt.Color[r=255,g=0,b=0]
What would you like to find the sucessor of? Please enter its score:
26
The successor is https://www.google.com/webhp at index20 with a score of 26 and the color ofred
```



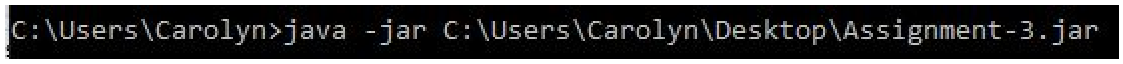
Invalid input:



```
6        Return the URL with the biggest score
7        Return the successor of a URL
10
Sorry, that was the wrong input.
Please Enter Your Choice:
1        Add a URL
2        Remove a URL
3        Sort Inorder
4        Find a URL
5        Return the URL with the smallest score
6        Return the URL with the biggest score
7        Return the successor of a URL
```

How to Run Application

1. <u>After unzipping the file by extracting them to your desired destination, run them on the command prompt using the line: java -jar C:\Users\(Name on PC)\(Location of Folder)\Uyen_Nguyen-PA2\Assignment-3.jar. (If needed, look at the properties of the .jar file to see where the file itself is actually located)</u>
    a. This is based off the command prompt line I used for my computer. For reference, I use

    ```
    C:\Users\Carolyn>java -jar C:\Users\Carolyn\Desktop\Assignment-3.jar
    ```

    I put the file inside my desktop.

    b. Just in case the project needs to be run on an IDE, copy and paste the provided .java files and import the provided jsoup .jar file.
2. <u>The simulator will start by searching for the links before returning three options and prompting you to choose based on the number.</u>
    a. Option 1: Inserting.

    The insert performance requires the user to insert a url name, press the ENTER key, insert a score, and then press ENTER again.

    b. Option 2: Removing.

    The remove performance requires the user to simply insert a score. The simulator will return the URL of that corresponding score and return the new list of URLs. Two URLs will share the same rank; one of them is the URL that is deleted. However the colors will alter if the fixup for deletion is used by the program.

   c. Option 3: Sort in order.

      The program will simply return the RBT sorted in order.

   d. Option 4: Find a URL.

      The program will show the RBT inorder and will ask you for the score you are looking for. The program will return the regular search value and the iterative search value. They should return the same.

   e. Option 5: Find minimum.

      The program will show the RBT inorder as reference and should return the first value of the order.

   f. Option 6: Find maximum.

      The program will show the RBT inorder as reference and should return the last value of the order.

   g. Option 7:

      Find successor. The program will show the RBT inorder as reference. The user will be asked to input the score of the URL and the program will return the successor of that URL.

3. <u>The simulator will continue running and ask for what options to try out. This will allow the user to try out different options without rerunning the code.</u>
4. <u>If an invalid value for the menu is inputted, the program will simply say the value is invalid and show the menu again so the user can run another option.</u>

Problems and Challenges Encountered

1. <u>Understanding RB Trees and their implementations:</u>

   The same challenge with bucket sort happened with RB Tree. When reading the algorithm, I wasn't so sure what the pseudocode is exactly saying because there are parts that simply said repeat but switch left and right, and I spent that time wondering if I switched the right one and which one I should be switching. On top of that, the method of inserting the nodes by ascending order had a drawback. This made it difficult to alter the order and placement in the tree because there is the case of having nothing on the left side of the tree and having more nodes on the right side, so I had to alter some of the code to suit that need (ex. rightRotation(B, z))

   Because of this, there are still errors that does not make the tree an actual balanced tree. Because of the nodes being placed in the tree by ascending order, the tree is more right-shifted and definitely violates RB Tree properties. I noticed that for nodes that share the same score, they also unintentionally share the same color. I wasn't able to make sure all the scores are unique within the time period of completing the programming assignment, and the scores themselves are by a sum of randomly selected numbers.

2. <u>Establishing Rank:</u>

   What I realized from my last programming assignment is that I didn't really set each type with a rank. The rank in the previous assignment is simply numbered, so it does not necessarily show where the link is placed. Because of this, part of this assignment's challenge is to create a method or something that will provide the rank.

I decided to go with using two sorting algorithms since I will be ranking them based on how high they score; initially I was going to use one (just quicksort), but decided to also use insertion sort because it would make the program run more efficiently, especially if the initial quicksort has already sorted everything.

3. Insertion and Deletion:

As a continuation of the first challenge, the methods I had the most trouble with is these two functions; this is a major issue because part of the RB Tree's purpose is being able to perform insertion and deletion. When running the methods, deletion more or less worked alright; the only issue is that the URL is not removed physically from the ArrayList, but the change can be seen because the ranks have shifted and the node that replaces the deleted URL's position now shares the same rank as the now deleted node, and some of the nodes altered colors. I haven't been able

Insertion, on the other hand, has more problems; this took most of my time trying to figure it out. It doesn't work with adding nodes that are not bigger in value. Because it is more right shifted, it is not able to handle anything that requires looking at the left subtree (since it is significantly more empty than the right side) so I wasn't able to add anything less than the highest ranking link's score. I had to alter some of the code to make it possible, but most of it in the time constraint never really succeeded. Besides that, it worked as it should for inserting a URL with a high score, as it shifts everything down and changes some node colors that need to be changed.

Lessons Learned

Generally, I learned that I really need to improve and understand RB Trees a lot better. These problems that I encountered may actually have solutions that could work but I wasn't able to really implement them due to lack of completely understanding and applying the properties of RB Trees and the syntax needed. Still, now that I know what I should look up on, I actually want to retry this program at another time to improve on it and approach it differently. I learned that things I thought would make the implementation easier actually made it harder, so from this assignment I learned that sometimes going a more simple route (not sorting them and just putting everything in arbitrarily) can be better.

In terms of RB trees, despite the errors and violations my program did, I can see how my RB Tree violated its required properties and how, when needed, the color of the nodes change to suit the properties in the way they can. With algorithms and data structures like RB Tree, I learned that it does really help to draw things out, especially with a RB Tree, to see what I've done wrong and what it should be like (even if I used the visualization, I was able to understand better where my program ran incorrectly when I drew it out). There were some minor improvements on the simulator itself (now I can continuously test out different options instead of restarting the simulator again)- which brought another lesson of sometimes the solution is a lot more simple than what I think it would be; I feel like when I try this program again, I will realize that the problems I have are easily solvable.

From the last programming assignment, I did the same approach in terms of dealing with how to handle difficult problems in my code, and it worked to an extent; some problems are fixed but others remained unfixed. At the same time, it was almost tempting to procrastinate when I tell myself to cool down and think about how to solve the problem, which was a problem I didn’t mention in the last report.. I think I simply need to practice doing the cool down and think about it method more and control it so I don’t use it as an excuse to slack off; that way, I’m more used to it and it can effectively work. This program seems like a good step forward to that.