nodeCoin: A Blockchain sim-tool for Crypto Currency

CS210 Data Structures and Algorithms with Dr. Basit Qureshi

Posted: Monday 21 October 2024

Due: Wednesday 30 October 2024

Weight: 15 %

Test site: https://www.hackerrank.com/241cs210p1

Submission on: https://lms.psu.edu.sa/

nodeCoin is an innovative cryptocurrency specifically designed for CS210 students studying data structures and algorithms. Unlike traditional cryptocurrencies, nodeCoin's architecture is built on core data structures including double ordered linked lists and binary max heaps, providing a unique learning opportunity that directly applies classroom concepts to real-world applications. Transactions within nodeCoin are recorded using a double ordered linked list, allowing efficient insertion, deletion, and traversal of transaction records. This structure maintains transactions in a sequential order, preserving both chronological and custom-sorted views, enhancing data access and manipulation. The binary max heap is employed to prioritize high-value transactions. Together, these data structures enable nodeCoin to perform key operations such as inserting new transactions, removing old or irrelevant records, and querying specific transactions efficiently, making it an ideal hands-on learning tool for understanding complex data structures in a blockchain context.

The nodeCoin blockchain doubly linked list contains n number of nodes/blocks. Each node contains a String Date (DDMMYYYY) and a pointer to the root of a MaxHeap (MaxHeapR). The MaxHeap stores two double values, i.e. the transaction amount (tAmt) and the transaction number (tNum). The MaxHeap maintains a record of transactions for that

day, prioritizing the transaction with highest value at the top. The following diagram depicts a nodeCoin blockchain composed of four nodes/blocks, structured as an ordered doubly linked list. In the illustration, only one MaxHeap is shown for the first node/block, but each node/block is designed to include its own MaxHeap.

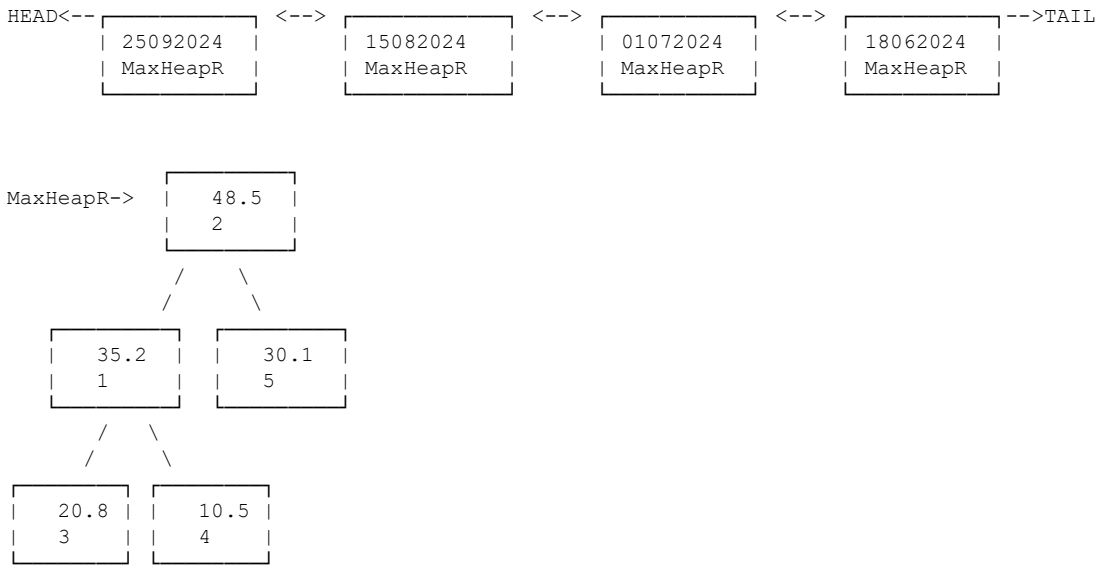


Figure 1. Illustration of the nodeCoin Blockchain

Students are expected to use the following API to develop their software.

nodeCoin
Node head **Node tail** **int size**
nodeCoin() **void insert(Node)** **Node getMax(Date)** **Node removeMax(Date)** **String getAll(Date)**

.

Node
String Date **Node nextHash** **Node prevHash** **MaxHeap root**
Node() **String toString(Date)**

.

MaxHeap
Transaction [] **int capacity** **int currentSize**
MaxHeap(capacity) **Swim(int)** **Sink(int)** **Insert(Transaction)** **Transaction RemoveMax()**

.

Transaction
double tAmt **int tNum**
Transaction(double, int)

Students are allowed to add/modify the API to suit their needs such as inclusion of setters and getters, however the attributes and methods names in the nodeCoin, Node and MaxHeap classes must persist. In addition to these classes, students will also write a tester program that reads from the console, processes the input and produces the correct output for this software considering the following conditions.

The integer value, `capacity` defines the maximum number of transactions allowed per day where $1 \leq capacity \leq 100000$.

On the console each line begins with an integer value $1 \leq i \leq 4$.

- The value $i = 1$ signifies that the `insert` method should be invoked. The subsequent string contains a date followed by the transaction amount.
- The value $i = 2$ indicates that the `getMax` method will be called with the specified date, which returns the maximum transaction amount for that date.
- When $i = 3$ is specified, the `removeMax` method is triggered with the date, removing the highest transaction value from the node.
- Finally, with $i = 4$, the `getAll` method is called using the date, which generates a string containing all information from the node, ordered by the highest transaction value first. All values are then cleared from the node, and the node is removed from the blockchain.
- **For any invalid value or input, the program writes -1 on the console.**

Running your program

The following is a test run with explanation.

Standard Input:

```
1 01072024 10.0
1 01072024 15.0
1 25092024 35.2
1 25092024 48.5
1 25092024 20.8
1 25092024 10.5
1 25092024 30.1
1 01072024 5.5
2 01072024
2 25092024
3 01072024
2 01072024
4 01072024
2 01072024
```

Standard Output:

```
15.0 2
48.5 2
10.0 1
10.0 1
5.5 3
-1
```

Explanation:

1 01072024 10.0 1 01072024 15.0
The two lines starting with 1 indicate insert method to be called. This inserts these two transactions in the node with date 01072024. The transaction with value 15.0 appears at the top/root of the Maxheap.
1 25092024 35.2 1 25092024 48.5 1 25092024 20.8 1 25092024 10.5 1 25092024 30.1
Similar to the above, the five transactions are inserted into the node/block with date 25092024. Note, the Max transaction with value 48.5 appears at the top.
1 01072024 5.5
This transaction is added to the node with Date 01072024. Note, this will not be appearing at the top as its value is less compared to the transaction at the top.
2 01072024
Returns the node with Date 01072024. At the moment, this node has 3 transactions stored. The Largest valued transaction is 15.0. The order of this transaction was numbered 2. Hence, 15.0 2 is printed on the screen
2 25092024
Similar to the above, from the node with Date 25092024, the largest valued transaction 48.5 with transaction number 2 is printed.
3 01072024
3 removes the highest transaction in the node with Date 01072024. The value 15.0 with transaction num 2 is removed from the node. Note, nothing prints on the screen.
2 01072024
At the moment, this node has 2 transactions stored. The Largest valued transaction is 10.0 with transaction num 1. Hence, 10.0 1 is printed on the screen
4 01072024
i=4 removes all transactions from the node with date 01072024. Note there are two transactions currently stored. Both of these are removed with the following printed on the screen: 10.0 1 5.5 3
2 01072024
Note, the node with date 01072024 does not exist anymore, hence a -1 is printed on the screen.

Evaluation:

You are allowed to work as a group with maximum 2 members in a group. Your work's evaluation would be based the following criterion.

Step 1: Verify correctness of your program on Hackerrank

You have unlimited chances to submit your code to hackerrank before the deadline. When you submit your code, the hackerrank platform tests your program against pre-built (hidden) testcases and grades your work. You need to fix all the mistakes in the project and try to earn a full score. For this project, there are no limitations on time and memory usage.

Step 2: Writing Report

Complete a report consisting of the following sections:

1. **Cover:** Use the **coversheet.** Add personal details and a screenshot from Hackerrank showing your score.

2. **Introduction**: Explain why you think the data structure you implemented is a good choice for implementing nodeCoin.
3. **Performance**:
    a. Give the runtime of your program using big Oh notation.
    b. Give the best case and worst case scenario using big Oh notation.
    c. Compare nodeCoin data structure to a Doubly Linked list.
    d. Compare nodeCoin data structure to a Max Heap.
4. **Conclusion**: Give reasons why nodeCoin data structure is better than a doubly linked list or a Max Heap implementation.
5. **Code**: Copy paste all of you code

Step 3: Upload your final submission to LMS. The submission will consist of

- One Microsoft word file consisting of your report.

Grading:

- Correctness of program verified on hackerrank - 12 points
- Completeness of report – 3 points

Warning: Not allowed to use java collections framework or any of its classes such as ArrayList etc.

The instructor reserves the right to determine the scores of each test case. Test-cases will be posted on hackerrank, students will have unlimited number of opportunities to post and test their project until the due date. **The system will not take any submissions after the due date.**

Code Inspection and plagiarism:
The code would be inspected by the instructor. The instructor would use the MOSS tool (https://theory.stanford.edu/~aiken/moss/) to determine the originality of submission.

If the code similarity is above 40%, the students would earn ZERO score on the project. This includes *all* group members for all teams involved as well (i.e. all other groups with similar code)

Additional Note:

Any Student would be requested to present their work. The instructor reserves the right to "**interview**" any student on their submission to see the understanding of the submission. The instructor may also ask the student to modify the code to satisfy any test-case(s) there in.

Submission Dead-Line:

The submission deadline is final. Late Submissions will be awarded ZERO points.

Important Notes:

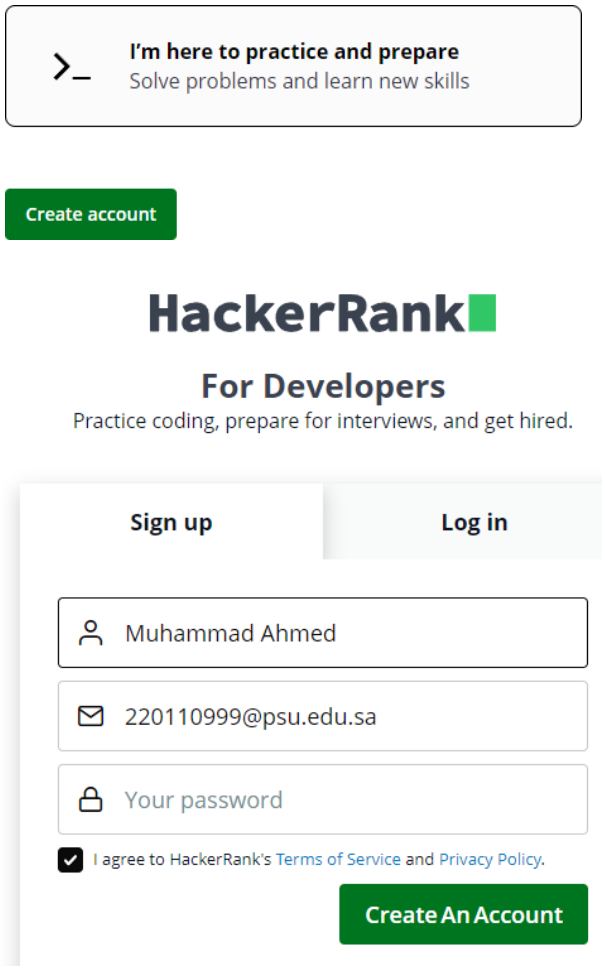
- It is the student's responsibility to check/test/verify/debug the code before submission.
- It is the student's responsibility to check/test/verify all submitted work (including jar files)
- It is the student's responsibility to verify that all files have been uploaded to the LMS.
- Incomplete or wrong file types that do not execute will NOT be graded.
- For each project, instructor will provide a few sample test-cases to verify the execution of your program.
- After an assignment/project has been graded, re-submission with an intention to improve an assignments score**will not be allowed**.

TUTORIAL:

Submission on Hacker Rank

Step 1. Register on https://www.hackerrank.com/

Make sure your username as your PSU ID number.



2. Join contest "241CS210p1"

Start working on your project.