Medium

**1**239

**₽** 232

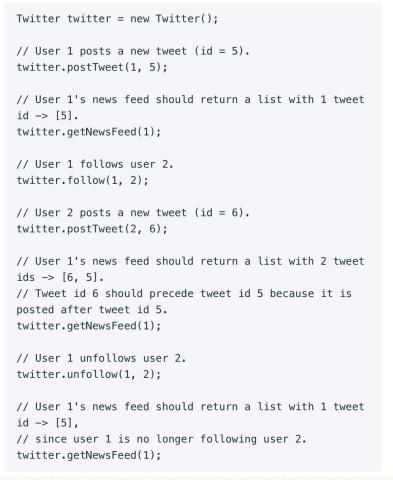
Add to List

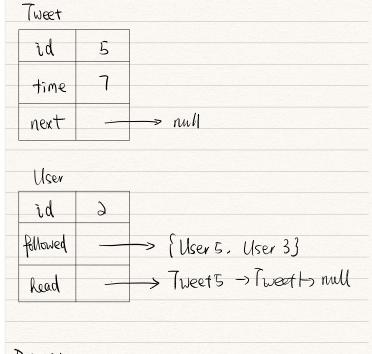
Share
 Share

Design a simplified version of Twitter where users can post tweets, follow/unfollow another user and is able to see the 10 most recent tweets in the user's news feed. Your design should support the following methods:

- 1. postTweet(userId, tweetId): Compose a new tweet.
- getNewsFeed(userId): Retrieve the 10 most recent tweet ids in the user's news feed. Each item in the news feed must be posted by users who the user followed or by the user herself. Tweets must be ordered from most recent to least recent.
- 3. follow(followerld, followeeld): Follower follows a followee.
- 4. unfollow(followerld, followeeld): Follower unfollows a followee.

## **Example:**





Priority queue.

```
ø
```

```
1 🔻
      class Tweet {
 2
      private:
 3
          int id;
          int time;
 4
 5
      public:
 6
 7
          Tweet *next;
 8
 9 🔻
          Tweet(int id, int time) {
10
              this->id = id;
11
              this->time = time;
12
              this->next = nullptr;
          }
13
14
15 ▼
          int getTime() {
16
              return this->time;
17
18
19 ▼
          int getId() {
20
              return this->id;
21
22
23
      };
24
25 ▼
      struct cmp{
          bool operator()(Tweet *a, Tweet *b){
26 ▼
27
              return (b->getTime() - a->getTime()) > 0;
28
29
      };
30
      class User {
31 ▼
      private:
32
33
          int id;
34
35
      public:
36
          unordered_set<int> *followed;
37
          Tweet *head;
38
          User(int userId) {
39 ▼
40
              this->followed = new unordered_set<int>();
              this->id = userId;
41
              this->head = nullptr;
42
43
              // follow self
              follow(this->id);
44
45
          }
46
47 ▼
          void follow(int userId) {
48
              this->followed->insert(userId);
49
          }
50
          void unfollow(int userId) {
51 ▼
              // cant unfollow self
52
53 ▼
              if (userId != this->id) {
54
                  this->followed->erase(userId);
55
          }
56
57
58 ▼
          void post(int tweetId, int &timestamp) {
              Tweet *tweet = new Tweet(tweetId, timestamp);
59
60
              timestamp++;
              // insert tweet in time order
61
              tweet->next = this->head;
62
63
              this->head = tweet;
64
          }
65
      };
66
```

```
67 ▼
      class Twitter {
 68
       private:
 69
           int timestamp;
 70
           unordered_map<int, User*> *userMap;
 71
 72
 73 ▼
           /** Initialize your data structure here. */
 74 ▼
           Twitter() {
 75
               timestamp = 0;
 76
               userMap = new unordered_map<int, User*>();
 77
 78
 79 ▼
           /** Compose a new tweet. */
 80 *
           void postTweet(int userId, int tweetId) {
 81 *
               if (!this->userMap->count(userId)) {
 82
                   this->userMap->insert(pair<int, User*>(userId, new User(userId)));
 83
 84
               this->userMap->at(userId)->post(tweetId, this->timestamp);
 85
 86
 87 *
           /** Retrieve the 10 most recent tweet ids in the user's news feed. Each item in the news feed must be posted by
       users who the user followed or by the user herself. Tweets must be ordered from most recent to least recent. */
 88 🔻
           vector<int> getNewsFeed(int userId) {
 89
               auto *res = new vector<int>();
               if (!this->userMap->count(userId)) {
 90 .
 91
                   return *res;
 92
               }
 93
 94
               unordered_set<int> *users = this->userMap->at(userId)->followed;
 95
 96
               auto *pq = new priority_queue<Tweet*, vector<Tweet*>, cmp>();
 97
 98 •
               for (auto user : *users) {
 99
                   Tweet *tweet = this->userMap->at(user)->head;
100 -
                   if (tweet == nullptr) {
101
                       continue;
102
103
                   pq->push(tweet);
104
105
106 ▼
               while (!pq->empty()) {
107 ▼
                   if (res->size() == 10) {
108
                       break;
109
                   Tweet *tweet = pq->top();
110
111
                   pq->pop();
112
                   res->push_back(tweet->getId());
113 ▼
                   if (tweet->next != nullptr) {
114
                       pq->push(tweet->next);
115
116
               }
117
118
               return *res;
119
120
121 ▼
           /** Follower follows a followee. If the operation is invalid, it should be a no-op. */
122 ▼
           void follow(int followerId, int followeeId) {
123 •
               if (!this->userMap->count(followerId)) {
124
                   this->userMap->insert(pair<int, User*>(followerId, new User(followerId)));
125
126 ▼
               if (!this->userMap->count(followeeId)) {
127
                   this->userMap->insert(pair<int, User*>(followeeId, new User(followeeId)));
128
129
               this->userMap->at(followerId)->follow(followeeId);
           }
130
131
```

66

```
132 ▼
           /** Follower unfollows a followee. If the operation is invalid, it should be a no-op. */
          void unfollow(int followerId, int followeeId) {
133 ▼
134 ▼
              if (!this->userMap->count(followerId)) {
135
                  this->userMap->insert(pair<int, User*>(followerId, new User(followerId)));
136
137 ▼
              if (!this->userMap->count(followeeId)) {
                  this->userMap->insert(pair<int, User*>(followeeId, new User(followeeId)));
138
              }
139
140
               this->userMap->at(followerId)->unfollow(followeeId);
141
142
      };
143
```

131