Twitter

Twitter API

- User-related functionality
  - Search, Followers, Friends, Send Messages …
- Timeline-related functionality
  - Look-up, Search, Retweet …
- Status-related functionality
- Account-related functinality
- ...

https://dev.twitter.com/

oauth2 based access

- Create Developer Account
- Register your Application
- Acquire Access Token
  - Consumer Key
  - Consumer Secret
  - Access Token
  - Access Token Secret

Twitter + Python

Tweepy library

pip3 install tweepy

```
import tweepy
from tweepy import OAuthHandler

cfg = {
    "consumer_key"        : "...",
    "consumer_secret"     : "...",
    "access_token"        : "...",
    "access_token_secret" : "..."
}
auth = OAuthHandler(cfg["consumer_key"], cfg["consumer_secret"])
auth.set_access_token(cfg["access_token"], cfg["access_token_secret"])
api = tweepy.API(auth)
```

User Look-up

```
me = api.me()
print(me)


user = api.get_user("ichatzi")
print(user.screen_name)
print(user.followers_count)
for friend in user.followers():
   print(friend.screen_name)
```

https://dev.twitter.com/overview/api/users

User Timeline Look-up

```
statuses = api.user_timeline()

for status in statuses:

    # process status here

    print(status.text)

    print(status.user.screen_name)

    print(status.created_at, "Fav: ",
status.favorite_count)

    Print("---")
```

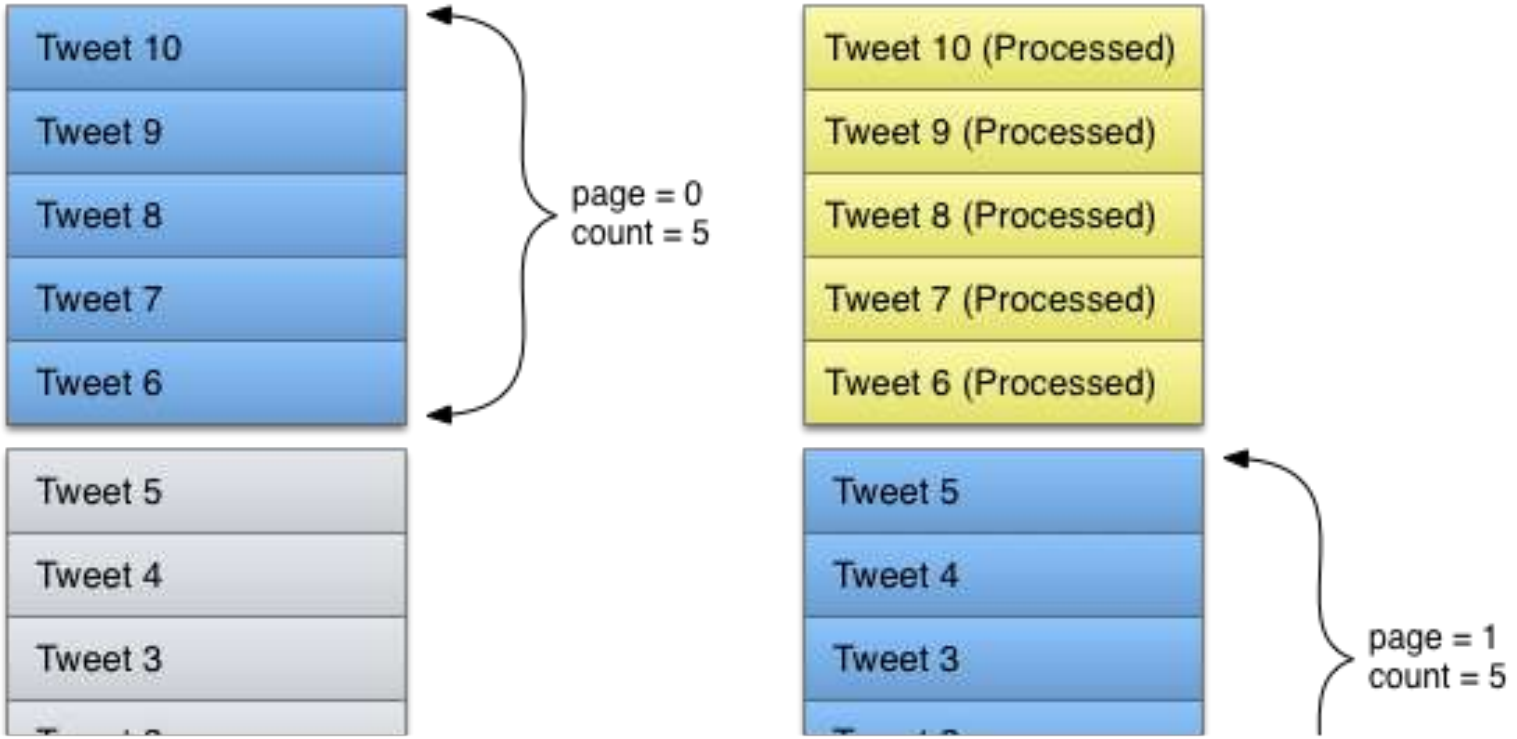
https://dev.twitter.com/overview/api/tweets

Home Timeline Look-up

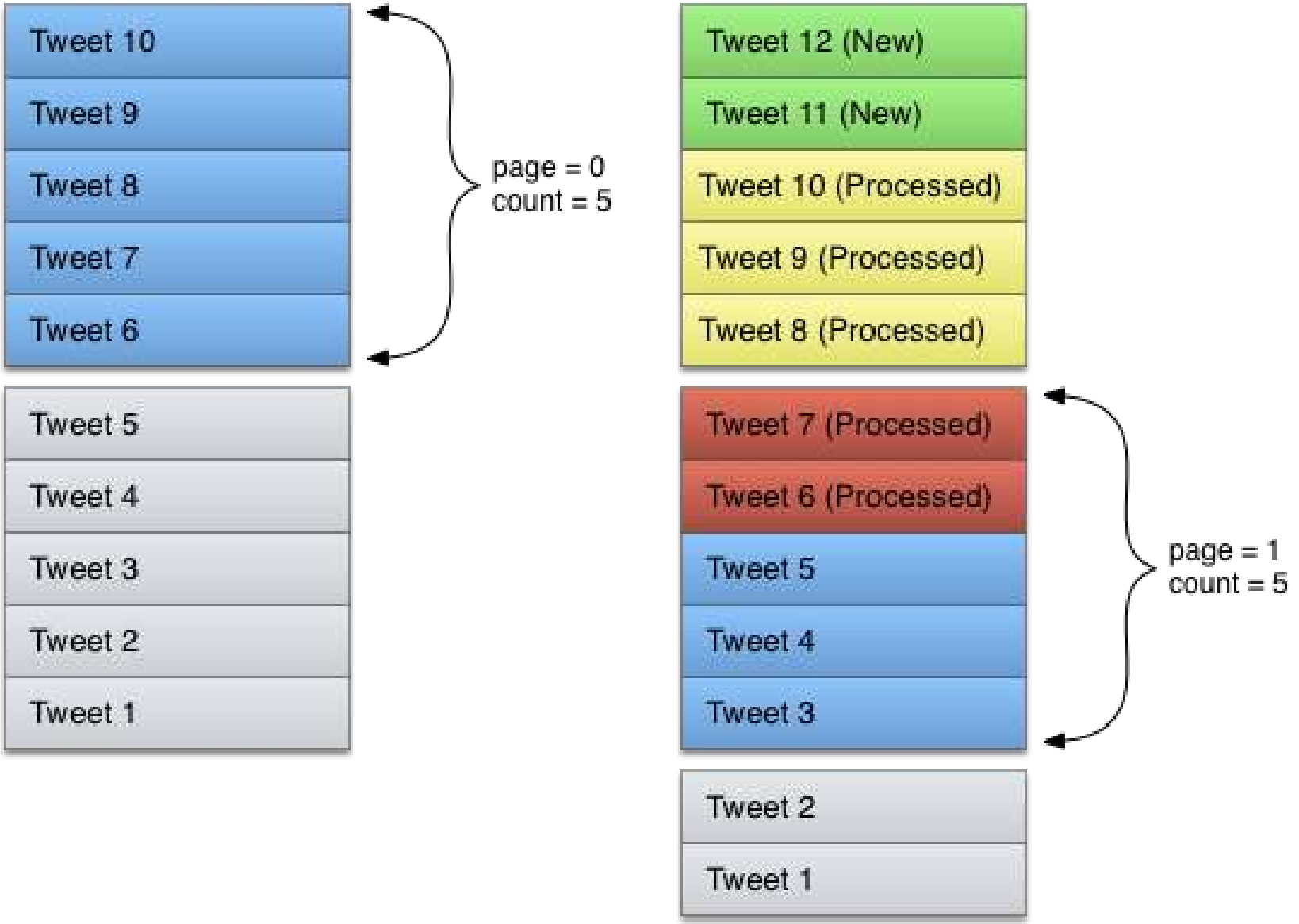
```
statuses = api.home_timeline()

for status in statuses:

    # process status here

    print(status.text)

    print(status.user.screen_name)

    print(status.created_at, "Fav: ",
status.favorite_count)

    print("---")
```

The problem with Pages

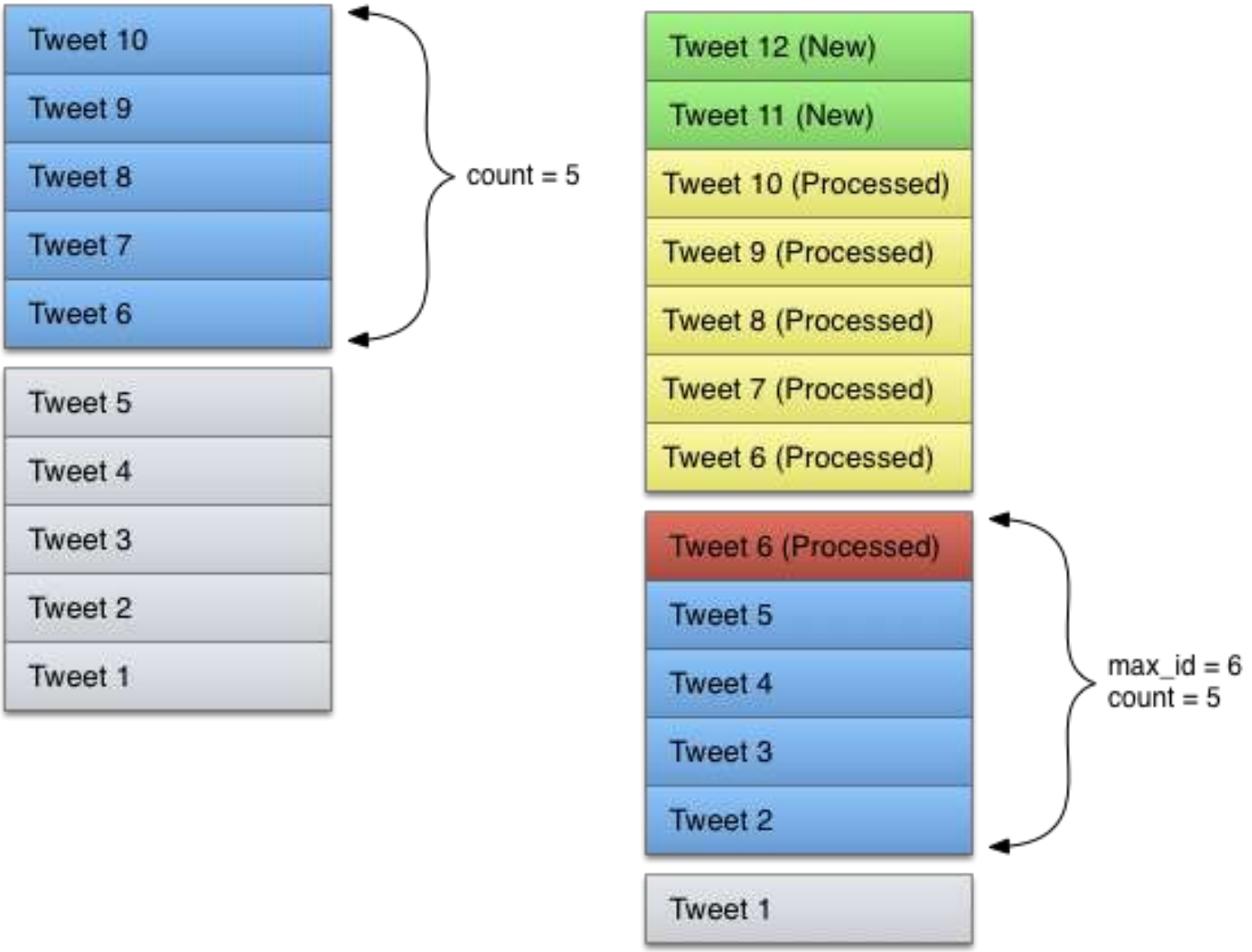
- A timeline has 10 reverse-chronologically sorted Tweets
- page size = 5 elements and requesting the first page



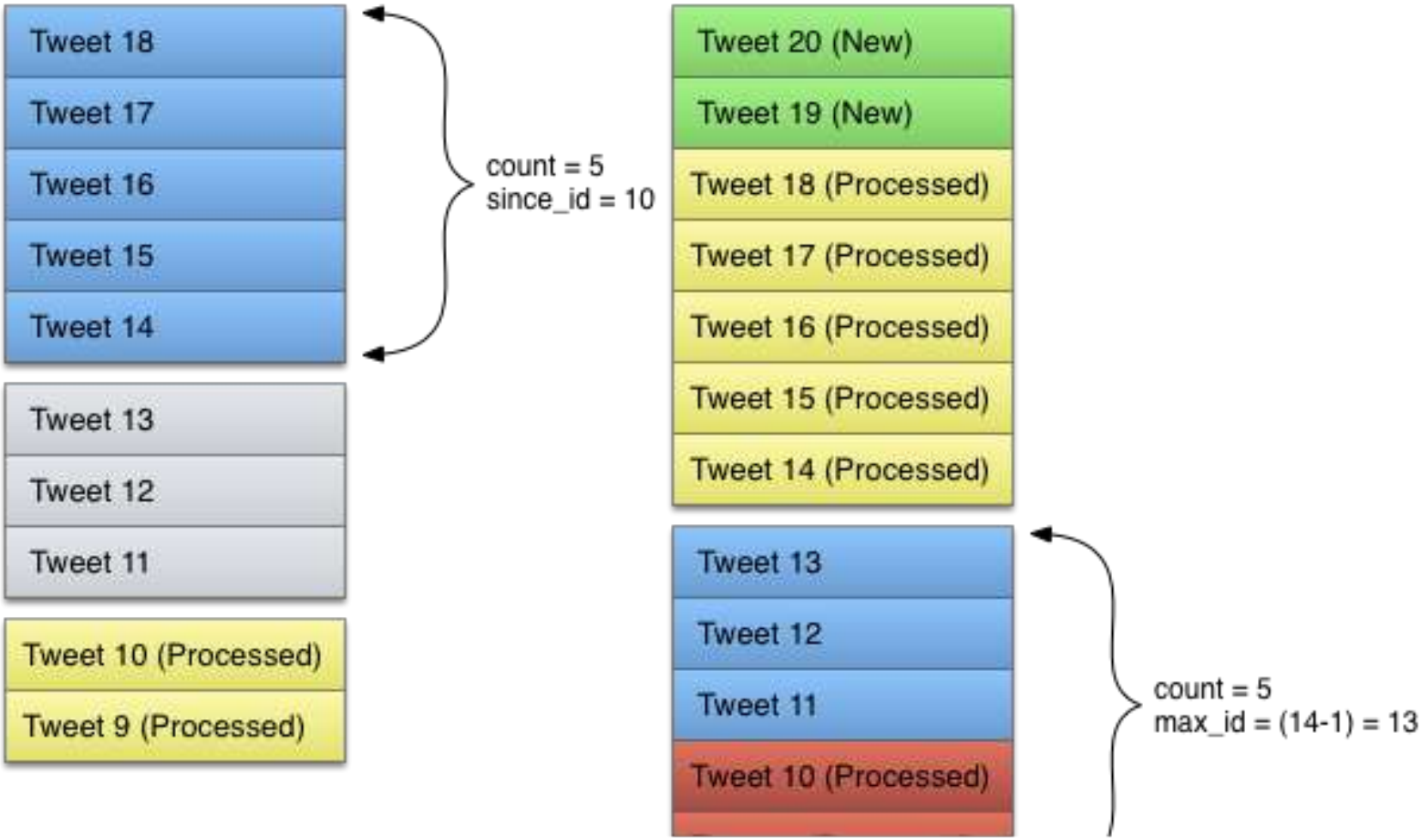
New Tweets added to the **front**



The max_id parameter

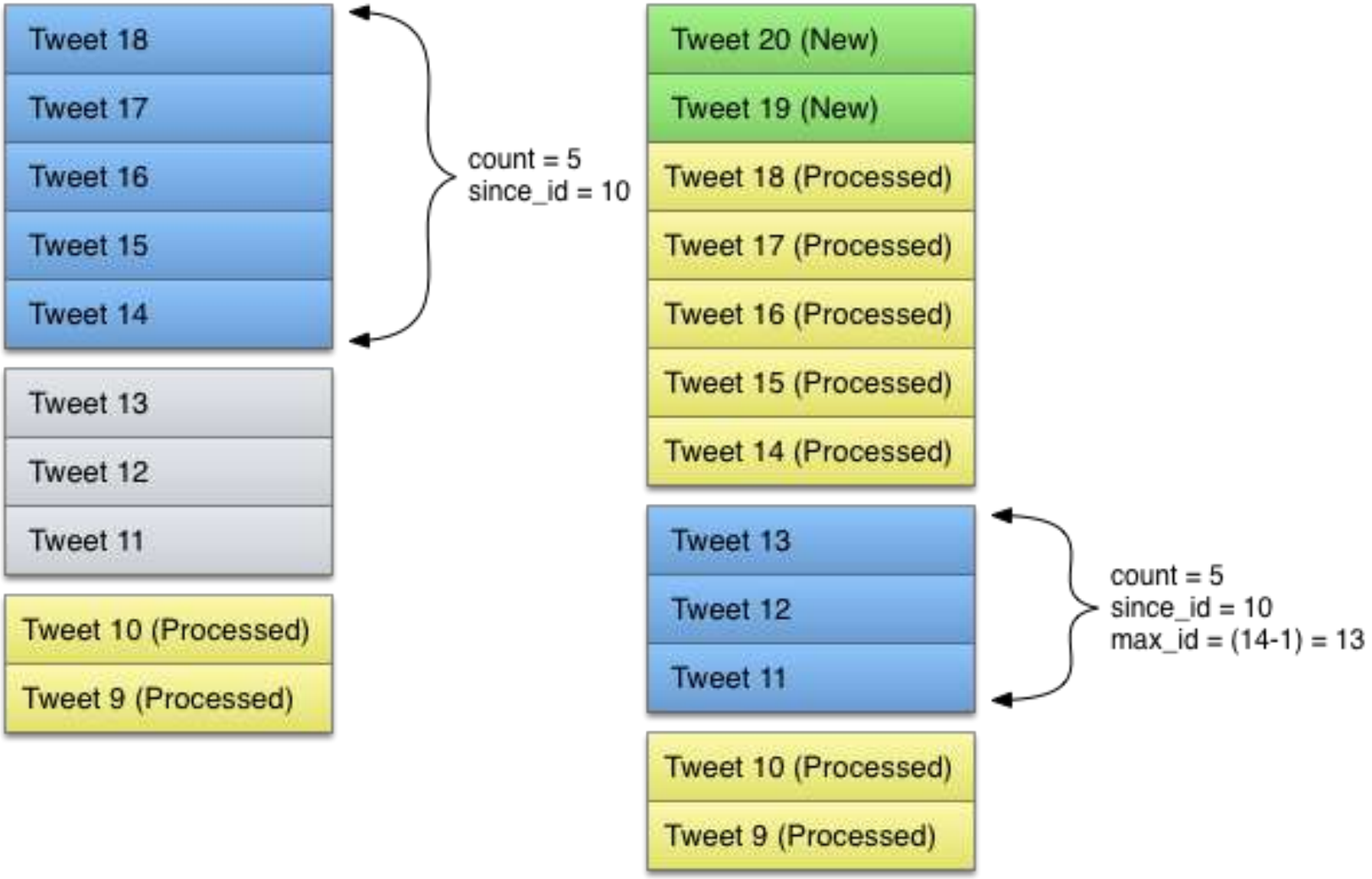


since_id for the greatest efficiency



Inefficient!

since_id for the greatest efficiency



Use both max_id and since_id to minimize amount of redundant data

Pagination vs Cursor

```
for status in tweepy.Cursor(api.home_timeline).items():
    # process status here
    print(status.text)
    print(status.user.screen_name)
    print(status.created_at, "Fav: ",
status.favorite_count)
    print("---")
```

API Rate Limits

```
api.rate_limit_status()
```

https://dev.twitter.com/rest/public/rate-limits

Searching Tweets

```
for status in api.search("#SpecialePessoa"):
    # process status here
    print(status.text)
    print(status.user.screen_name)
    print(status.created_at, "Fav: ",
status.favorite_count)
    print("---")
```

Searching Users

```
users = api.search_users("Sapienza", 10)
for user in users:
    print(user.screen_name)
```

Followers vs Followers_ID

```
for friend in tweepy.Cursor(api.followers).items():
    print(friend.screen_name)
```

```
flist = []
for friend in tweepy.Cursor(api.followers_ids).items():
    flist.append(friend)
```

https://dev.twitter.com/rest/reference/get/users/lookup

Followers vs Followers_ID

```
flist = []
for friend in tweepy.Cursor(api.followers_ids).items():
    flist.append(friend)

batch = []
while flist:
    batch.append(flist.pop())
    if (len(batch) >= 100):
        print(batch)
        users = api.lookup_users(user_ids=batch)
        for u in users:
            print(u.screen_name)
        batch = []

users = api.lookup_users(user_ids=batch)
for u in users:
    print(u.screen_name)
```

Handling Rate Limit Exception

```
import time
def getFollowers(userId, friendids=[]):
    friend_count = 0
    c = tweepy.Cursor(api.followers_ids, id=userId).items()

    while True:
        try:
            friend = c.next()
            friendids.append(friend)

        except tweepy.TweepError:
            # hit rate limit, sleep for 15 minutes
            print('Rate limited. Sleeping for 15 minutes.')
            time.sleep(15 * 60 + 15)
            continue
        except StopIteration:
            break

    return friendids
```

Mining Social Relations

- Start from your account
  - Or any other tweeter account
- Build Graph
- Collect all Followers
- Collect Followers of Followers
- Collect Followers of Followers of Followers
- Collect …
- Compute Diameter
- Identify User with highest Centrality