Welcome Judes!

Time Remaining

Time's up!

The time limit for the assessment is up. A recruiter will be in touch if there is anything to follow up on.

The Problem

Many companies use HubSpot for its calling capabilities. Sales reps use the HubSpot product throughout the day to make phone calls to prospects.

We've found that certain customers using HubSpot have a large number of sales reps *concurrently* making calls with HubSpot, and this puts heavy load on our systems. In response to this, we'd like to bill our customers based on their peak calling load. In other words, we'd like to bill customers based on their maximum number of concurrent calls.

You're provided with an HTTP GET endpoint that returns phone call records represented as JSON: `https://candidate.hubteam.com/candidateTest/v3/problem/dataset?userKey=205013308a48dde8860b3bb696c9`

Each call looks something like this:

```
{
  "customerId": 123,
  "callId": "2c269d25-deb9-42cf-927c-543112f7a76b",
  "startTimestamp": 1707314726000,
  "endTimestamp": 1707317769000
}
```

- `customerId` is a unique identifier for one customer. One customer may have many sales reps concurrently making calls.
- `callId` is a unique identifier for a single phone call. No two phone calls will have the same `callId`.
- `startTimestamp` is when the call started. This value is given as a UNIX timestamp in milliseconds. In other words, it's the number of milliseconds that passed between the UNIX epoch (1970-01-01 00:00:00 UTC) and the start of the call.
- `endTimestamp` is when the call ended. This value is also given as a UNIX timestamp in milliseconds. `endTimestamp` will always be greater than `startTimestamp` for a given call record.

For the billing team to charge our customers correctly, they need to know the maximum number of concurrent calls for each customer *for each day*. The billing team has asked you to POST this information to the following endpoint: `https://candidate.hubteam.com/candidateTest/v3/problem/result?userKey=205013308a48dde8860b3bb696c9`. The POST body must be in the following format:

```
{
  "results": [
    {
      "customerId": 123,
      "date": "2024-02-07",
      "maxConcurrentCalls": 1,
      "timestamp": 1707314726000,
      "callIds": [
        "2c269d25-deb9-42cf-927c-543112f7a76b"
      ]
    }
  ]
}
```

- `date` is a UTC date in the format YYYY-MM-DD. So the example date refers to February 7th 2024.
- `maxConcurrentCalls` is the maximum number of simultaneous calls that occurred *at any time during the corresponding date* for this customer.
- `timestamp` is a UNIX timestamp (in milliseconds) at which `maxConcurrentCalls` was reached for this customer and date. There could be multiple time periods during this date where `maxConcurrentCalls` is reached. A `timestamp` during any of these time periods can be chosen.
- `callIds` is an array of calls that were happening for this customer at `timestamp`. The length of this array should equal `maxConcurrentCalls`. The order of `callIds` does not matter.

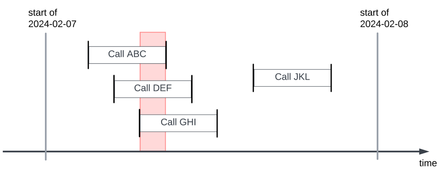
This example response only has one entry in the `results` array, but we expect the actual answer to have multiple results.

Note

- The `startTimestamp` of a call is inclusive, and the `endTimestamp` of a call is exclusive. This means that:
  - If call A has an `endTimestamp` of 123, and call B has a `startTimestamp` of 123, they never overlapped.
  - For a given `results` entry, the `timestamp` should always be less than the `endTimestamp` of each call in `callIds`.
- A single call may span multiple UTC dates, and calls can be arbitrarily long.
- The order of results posted does not matter.
- For a given `customerId` and `date`, if no phone calls occurred during that date, there should be no `results` entry with that `customerId` and `date` combination.
- No two entries in the `results` array should have identical values for both `customerId` and `date`. In other words, for a given `customerId` and `date` combination, there should be at most one entry in the array.

Example 1

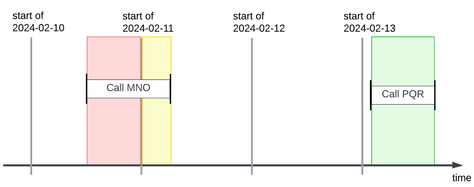
Let's say that for some customer, the diagram below is a visual representation of their call records.



For this customer, for the date 2024-02-07, `maxConcurrentCalls` should equal 3, since that's the largest number of calls that were concurrently happening on that date. The given `timestamp` for this customer on date 2024-02-07 can be any UNIX timestamp in the time period highlighted in red.

Example 2

Let's say that for some other customer, the diagram below is a visual representation of their call records.



Call MNO starts on February 10th and ends on February 11th. For this customer:

- For the date 2024-02-10, `maxConcurrentCalls` should equal 1. The given `timestamp` can be any UNIX timestamp in the time period highlighted in red.
- For the date 2024-02-11, `maxConcurrentCalls` should equal 1. The given `timestamp` can be any UNIX timestamp in the time period highlighted in yellow.
- For the date 2024-02-12, no result should be given.
- For the date 2024-02-13, `maxConcurrentCalls` should equal 1. The given `timestamp` can be any UNIX timestamp in the time period highlighted in green.

API Guidelines

If your answer is correct, the API will return `200 OK`. If the request is malformatted or incorrect, the API will return `400` along with a message indicating if the response is of the wrong structure or incorrect. If you submit an incorrect response that is close or on the right track, the error message will tell you so.

If you get a `5xx` response, let us know and we'll help you out.

The `candidate.hubteam.com` domain is set up with a permissive cross-origin policy, so you can `POST` to it from any location in a browser if you choose to implement in an in-browser JS solution.

Testing Guidelines

If you want help testing and debugging your solution, you can use the these endpoints:

`GET https://candidate.hubteam.com/candidateTest/v3/problem/test-dataset?userKey=205013308a48dde8860b3bb696c9`

- This endpoint above returns a small example dataset for testing.

`GET https://candidate.hubteam.com/candidateTest/v3/problem/test-dataset-answer?userKey=205013308a48dde8860b3bb696c9`

- This endpoint above returns a correct answer corresponding to the test dataset.

`POST https://candidate.hubteam.com/candidateTest/v3/problem/test-result?userKey=205013308a48dde8860b3bb696c9`

- This endpoint above accepts a request body with a JSON answer. A 200 status code is returned if the given answer is correct for the test dataset. A 400 status code is returned if the given answer is incorrect for the test dataset.
- If you use the response body from the `test-dataset-answer` endpoint as the body of your `test-result` request, a 200 status code will be returned. The `test-result` endpoint exists just to help you debug your solution. A 200 response from that endpoint does not mean you have passed the full assessment. You still need to POST a correct answer to the `https://candidate.hubteam.com/candidateTest/v3/problem/result?userKey=205013308a48dde8860b3bb696c9` endpoint to pass the full assessment.

Evaluation

When you're done, this page will update with a form to upload your code. We'll evaluate you based on two things:

- First and foremost, if you complete the project within three hours.
- Next, the time from when you started the assessment to the time you submit a correct solution.

Uploading your solution to the interview project

Please upload a **.zip archive of the complete program**. You do not need to upload any dependencies used, but it is suggested (although not required) to include a dependency declaration in the file you used. (e.g. a `package.json` for Node, `pom.xml` for Java, or a `requirements.txt` for Python)

If you have any questions, email the recruiter you've been working with.

Choose file Upload