talent.io homework

Hello there! Thank you for considering **talent.io** as your next company. We would like to know more about your experience with programming and problem solving. There are 4 independent parts to this assignment:

Part I - **Algorithmics**

Part II - **Backend**

Part III - **Frontend**

Part IV - **DataBase**

Complete **parts 1 & 2**, and pick **a third one out of parts 3 & 4**, depending on what you’re more comfortable working on.

You can code the assignment in **Ruby, Python, or JavaScript**. You can use any backend or frontend library that you deem useful. Just so you know, our stack at **talent.io** is Ruby 2.4.x, Rails 5, Redis, Resque, PostgreSQL and React.js.

*Tips:*

* ***Reach out to us*** *if you have any question! We will be happy to clarify anything if needed.*
* ***Comment your code*** *to let us know what choices you made and why.*
* *Don’t be afraid to ask* ***Google*** *for help.*
* *When you are done, please send us your solutions in a zip file.*
* *We expect you to spend between 1h and 4h on this assignment, depending on the number of questions you choose to tackle and your level of experience.*

# 

# 

# 

# Part I - Algorithmic

You are given an array of 10,000 random integers, positive or negative, in the file *tupples/unordered\_integers.json*. Your job is to determine all couples of elements summing to a given number. You can use the same element twice. We don’t double count couples if they are present more than once.

Example: In the array [-5, 13, 4, 9, -1, 13], the couples summing to 8 are (-5, 13), (4, 4), and (9, -1).

1. Write a function that, given an array and an integer s, returns the set of couples summing to s. Give the average time-complexity and memory-complexity of your algorithm. You can check that your solution is correct by comparing the output of your algorithm with *unordered\_couples.json* for the sum 8765.
2. Assuming that all the elements are all positive integers, and that the input array is sorted from smallest to largest, how can you optimize the solution? You can check your solution by comparing the output you get for *ordered\_positive\_integers.json* with *ordered\_positive\_couples.json*

# Part II - Backend

### Presentation

At talent.io, we do business in multiple countries: we hire people from all over the world, and our clients are based in France, Germany and the UK. We will open new countries in the upcoming months, so we need to know how to convert between currencies.

### Assignment

(1) Write a library that can convert an amount from one currency to another, using [fixer.io](https://fixer.io)’s API.

For instance, we will be able to ask your library to convert the amount 13,000 from USD to EUR, based on the exchange rate on 19/03/2018.

(2) When a candidate is hired by one of talent.io’s clients, we call it a **deal**. A **deal** has the following properties:

|  |  |
| --- | --- |
| **Field name** | **Field type** |
| id | bigint |
| amount | integer |
| currency | text (3-letters code) |
| created\_at | timestamp |

### 

Implement a function that returns the amount of a deal, whatever the currency is, in €.

**E.g**: Provide the amount in euros of deal **X.**

(3) Implement one or several functions that allow to express a global revenue for talent.io in euros for a specific week / month / year.

E.g: Provide the talent.io’s global revenue for March 2017, in €.

**Keep in mind:**

* We don't care about the time of day when converting.
* You can write as many classes / modules / methods as you want.
* You can use any library you want.
* You should be able to justify your choices.
* We expect clarity, optimizations and high-quality architecture.

# Part III - Frontend

Create a basic web page where each of the 4 corners of the screen displays a coloured button. You may pick any colour for each corner.

Next, create a display of 10 squares in the center of the page. When one of the 4 corner buttons is clicked, it automatically updates this list of 10 squares. The center squares always display the last 10 colours that were clicked.

On page refresh, the list is **not** emptied.

You can use any UI library such as React, Redux, jQuery, or use vanilla Javascript.

# Part IV - DataBase

You are given a PostgreSQL database with the following tables & columns:

* *talent\_advocates*:
  + id [Bigint]
  + name [String]
* *candidates*:
  + id [Bigint]
  + name [String]
  + talent\_advocate\_id [Bigint], foreign key for *talent\_advocates*
* *job\_offers*:
  + id [Bigint]
  + candidate\_id [Bigint], foreign key for *candidates*

Write a query that returns the leaderboard of Talent Advocates on the count of offers for their candidates, i.e. a table with the following information:

|  |  |  |  |
| --- | --- | --- | --- |
| talent\_advocate\_id | talent\_advocate\_name | nb\_offers | rank |
| 42 | John Doe | 764 | 1 |
| 12 | Jane Washington | 700 | 2 |
| 659 | Bob Dylan | 700 | 2 |
| 87645 | Roger Federer | 345 | 4 |
| ... | ... | ... | ... |
| 13 | Donald Trump | 0 | 23 |

**Note that:**

* The table is sorted by decreasing number of offers.
* Talent Advocates can share the same score. You can chose to give them the same rank, as in this example, or not.
* We want to include Talent Advocates with no job offers.