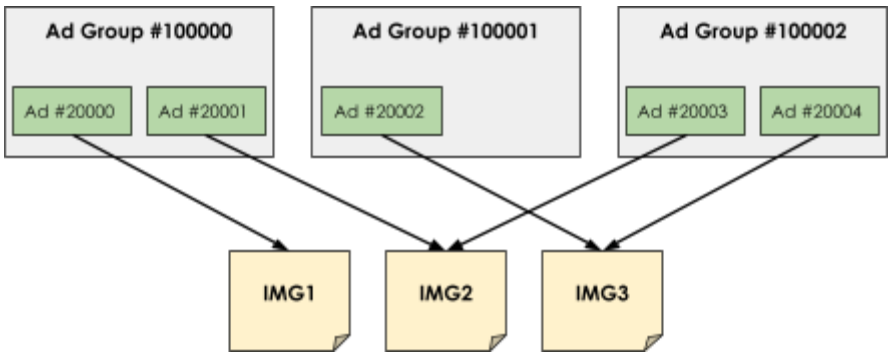
Albert: Coding Exercise

Background

Albert is an autonomous system for managing enterprise-scale marketing campaigns across different online advertising platforms, mainly Google AdWords and Facebook.

As such, Albert needs to constantly handle campaign objects - that is, build and modify campaigns, ads, targeting parameters, etc. - and analyze their performance to make choices. For example, understand which ad image works better and which worse, so the better performing images can be pushed stronger to the relevant audiences.

A standard advertising campaign is of a hierarchical structure which looks somewhat like this:



As you can see, the campaign consists of **ad groups**, which contain **ads**. Each ad has an **image** file (which stands for a "banner" or a Facebook image ad).

Ad groups and ads are unique (the same ad object cannot belong to different ad groups). Images, however, are not - the same image may be used in multiple ad objects, either under the same ad group or in different ad groups.

This structure is quite effective for managing campaigns. For example, you could take 3 ads, put them in the same ad group, and set up this ad group to target a specific audience. Other ads will be put in another ad group, and target different audiences. As the campaign runs and data is being collected, you can check and compare the performance of different ad groups, ads, and images, and thus make better choices to improve your campaign's effectiveness.

In order to run performance analysis, Albert first needs to determine which of the ads are using the same image. **That's the purpose of this exercise.**

Your Input

Your input is a JSON file (**image_data.json**) which contains a list of ads.
Each ad is a JSON object with the following fields:

- **ad_id**: unique system-wide
- **ad_group_id**: unique system-wide
- **url**: the url of the ad's image (**not** unique)
- **impressions**: the total number of ad views so far
- **clicks**: the total number of ad clicks so far
- **conversions**: the total number of purchases which came from this ad so far

Example:

```
{
  "ad_id": 2000008,
  "ad_group_id": 1000001,
  "url": "http://adwords.google.com/clientUploads/1707df73-19a4-42e7-981e-b151cbc0b3d0",
  "impressions": 76326,
  "clicks": 40058,
  "conversions": 5385
 }
```

For the purpose of this exercise, you can ignore **impressions**, **clicks**, and **conversions**.

Code you can use

For the exercise, you will be provided with the class **`ComparisonService`**.
You can use this class as is, and should **not** re-implement it or change it.

The `ComparisonService` class contains the following three functions which compare two image URLs and determine if they are referring to the same image:

- **`strings_are_equal():`** Two images with the same URL are obviously identical.
- **`urls_are_equal():`** Albert's storage service uses a proxy server. As a result, sometimes the same image file will get two different external URLs. This function checks this with the proxy server.
- **`images_are_equal():`** Sometimes the same image has been uploaded twice, only in different sizes, qualities, or formats, resulting in different files which in fact contain the same image. This function downloads the two files and checks them pixel-by-pixel to determine whether they can be considered identical or not.

All these functions get two URLs as arguments, and return **`True`** if the images are found identical.
And… the `ComparisonService` class doesn't really implement these functionalities - it's a mockup class :) But for the sake of the exercise, let's assume that it does.

Your Task: Regroup ads by image

In the input data, find all the ad objects which share the same image, and group them together. The output should be a list of lists of tuples, looking like this:

```
[
        [(1000000, 2000000), (1000001, 2000001)],
        [(1000002, 2000002), (1000003, 2000003), (1000004, 2000004)]
]
```

Each item in the result is a list of ads which are using the same image.
Each ad in the inner lists is represented by a tuple of **`(ad_group_id, ad_id)`**.

In the example above, ad 2000000 in ad group 1000000 and ad 1000001 in ad group 2000001 are both using the same image.

A few notes:

- Please keep your code as clean and readable as possible. Remember that it should reflect "how you usually write code".
- Try to make the code as efficient and fast as possible, so it could also handle much larger inputs - **but not at the expense of clarity and readability.**
- We also attached a **solution.json** file, where you can find the correct solution for the exercise, if you wish to compare and verify that you got it right.
  (The order of the items in the result lists is unimportant and can be ignored).

Thanks, and have fun! :)