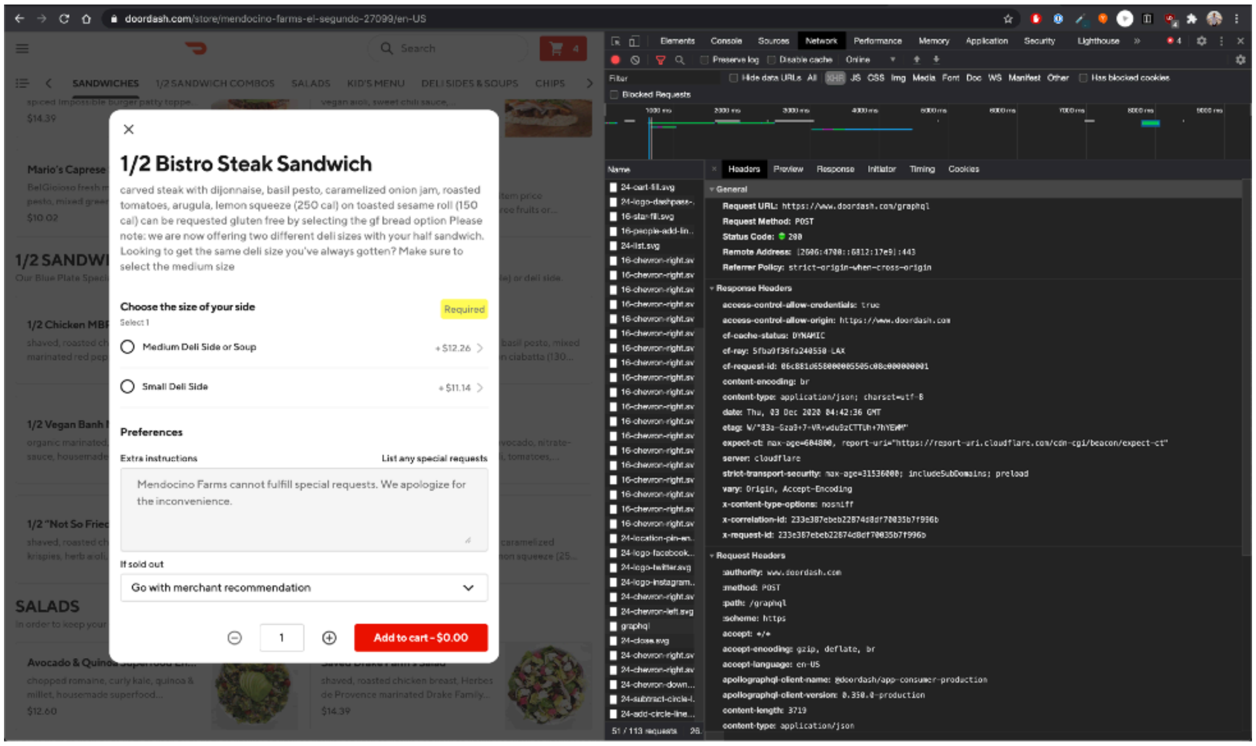
# Programming Challenge

We're scraping DoorDash, and we need to get the service fee for a particular restaurant. For a bit of background, service fees are the percentage that DoorDash charges the consumer for offering their service, and they can range anywhere from 0% to 15%. DoorDash's api unfortunately doesn't give us the service fee percentage directly, but they do show you the service fee at checkout if you have items in your cart.

So, we have to add a menu item to our cart. Menu items, though, have required options, and those options can have sub options. If we're going to send the "add this item to our cart" api request, how do we craft the body of that request with the required options? In this challenge, you'll be writing a function that takes JSON for a menu item, and outputs JSON for the minimum set of required menu item options.

For an example, head to https://www.doordash.com/store/mendocino-farms-el-segundo-27099/en-US, and, so you can inspect the api requests as we click around, open up the Chrome Dev Tools, and click the Network tab.

Then, click on a menu item, say, 1/2 Bistro Steak Sandwich because it has a lot of options. In the Network tab, check the call the frontend made to the `/graphql` endpoint. It should look something like this:



**1/2 Bistro Steak Sandwich**

carved steak with dijonnaise, basil pesto, caramelized onion jam, roasted tomatoes, arugula, lemon squeeze (250 cal) on toasted sesame roll (150 cal) can be requested gluten free by selecting the gf bread option Please note: we are now offering two different deli sizes with your half sandwich. Looking to get the same deli size you've always gotten? Make sure to select the medium size

Choose the size of your side — Required
Select 1

- Medium Deli Side or Soup +$12.26
- Small Deli Side +$11.14

Preferences

Extra instructions — List any special requests

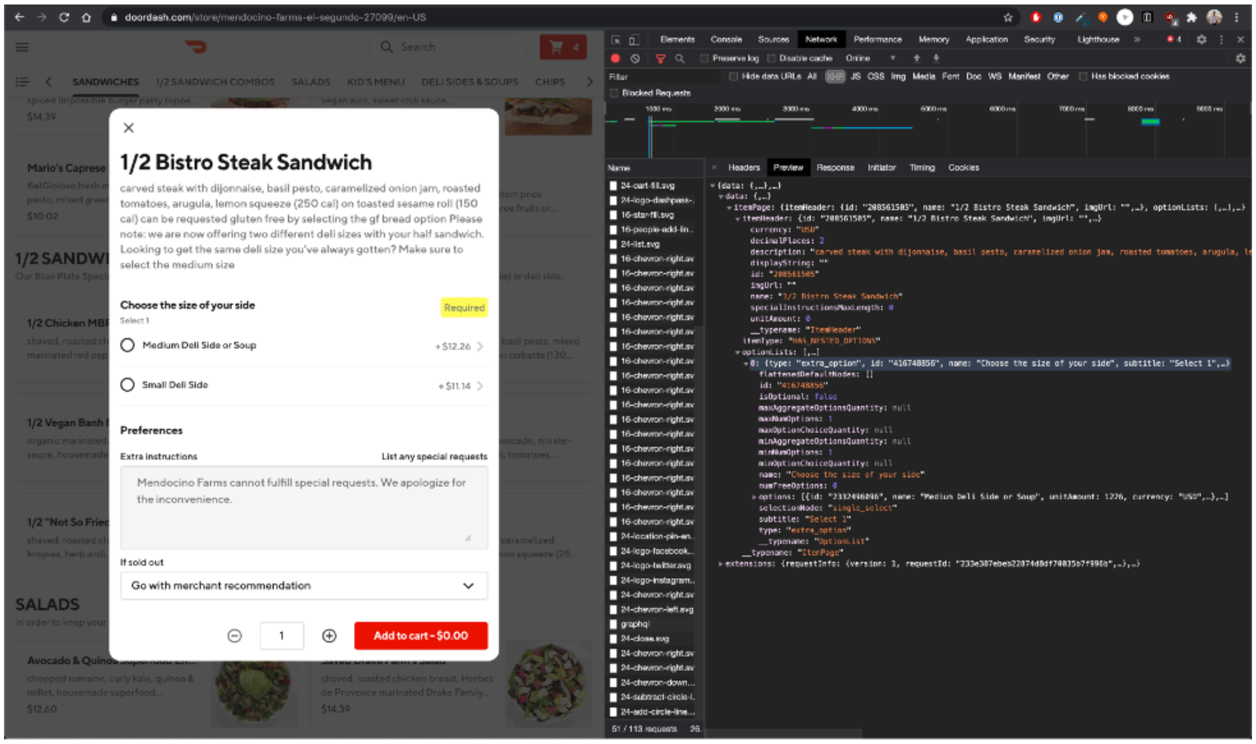
Mendocino Farms cannot fulfill special requests. We apologize for the inconvenience.

If sold out: Go with merchant recommendation

1 — Add to cart - $0.00

```
General
Request URL: https://www.doordash.com/graphql
Request Method: POST
Status Code: 200
Remote Address: [2606:4700::6812:17e9]:443
Referrer Policy: strict-origin-when-cross-origin
Response Headers
access-control-allow-credentials: true
access-control-allow-origin: https://www.doordash.com
cf-cache-status: DYNAMIC
cf-ray: 5fba0f36fa240550-LAX
cf-request-id: 06c881d6580000085505c08c000000001
content-encoding: br
content-type: application/json; charset=utf-8
date: Thu, 03 Dec 2020 04:42:36 GMT
etag: W/"83a-Gza9+7-VR+wdu9zCTTUh+7hYEWM"
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
server: cloudflare
strict-transport-security: max-age=31536000; includeSubDomains; preload
vary: Origin, Accept-Encoding
x-content-type-options: nosniff
x-correlation-id: 233e387ebeb22874d8df70035b7f996b
x-request-id: 233e387ebeb22874d8df70035b7f996b
Request Headers
:authority: www.doordash.com
:method: POST
:path: /graphql
:scheme: https
accept: */*
accept-encoding: gzip, deflate, br
accept-language: en-US
apollographql-client-name: @doordash/app-consumer-production
apollographql-client-version: 0.350.0-production
content-length: 3719
content-type: application/json
```

Now click on the “Preview” or “Response” tab for the request to get the JSON for the menu item. It should look like this:



1/2 Bistro Steak Sandwich

carved steak with dijonnaise, basil pesto, caramelized onion jam, roasted tomatoes, arugula, lemon squeeze (250 cal) on toasted sesame roll (150 cal) can be requested gluten free by selecting the gf bread option Please note: we are now offering two different deli sizes with your half sandwich. Looking to get the same deli size you've always gotten? Make sure to select the medium size

Choose the size of your side — Required — Select 1

- Medium Deli Side or Soup +$12.26
- Small Deli Side +$11.14

Preferences

Extra instructions — List any special requests

Mendocino Farms cannot fulfill special requests. We apologize for the inconvenience.

If sold out: Go with merchant recommendation

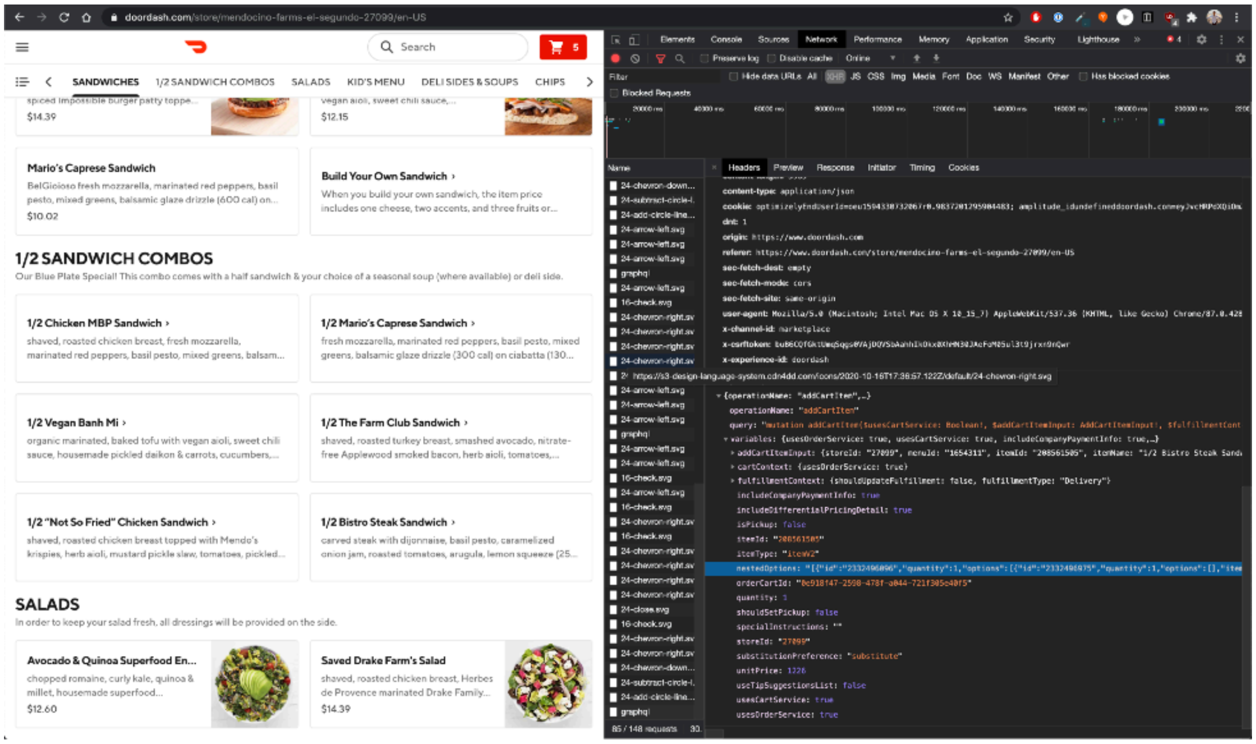
1 — Add to cart - $0.00

```
▾{data: {,…},…}
  ▾data: {,…}
   ▾itemPage: {itemHeader: {id: "288561505", name: "1/2 Bistro Steak Sandwich", imgUrl: "",…}, optionLists: [,…],…}
    ▾itemHeader: {id: "288561505", name: "1/2 Bistro Steak Sandwich", imgUrl: "",…}
       currency: "USD"
       decimalPlaces: 2
       description: "carved steak with dijonnaise, basil pesto, caramelized onion jam, roasted tomatoes, arugula, le
       displayString: ""
       id: "288561505"
       imgUrl: ""
       name: "1/2 Bistro Steak Sandwich"
       specialInstructionsMaxLength: 0
       unitAmount: 0
       __typename: "ItemHeader"
      itemType: "HAS_NESTED_OPTIONS"
    ▾optionLists: [,…]
     ▾0: {type: "extra_option", id: "416748856", name: "Choose the size of your side", subtitle: "Select 1",…}
        flattenedDefaultNodes: []
        id: "416748856"
        isOptional: false
        maxAggregateOptionsQuantity: null
        maxNumOptions: 1
        maxOptionChoiceQuantity: null
        minAggregateOptionsQuantity: null
        minNumOptions: 1
        minOptionChoiceQuantity: null
        name: "Choose the size of your side"
        numFreeOptions: 0
       ▸options: [{id: "2332496896", name: "Medium Deli Side or Soup", unitAmount: 1226, currency: "USD",…},…]
        selectionMode: "single_select"
        subtitle: "Select 1"
        type: "extra_option"
        __typename: "OptionList"
      __typename: "ItemPage"
  ▸extensions: {requestInfo: {version: 1, requestId: "233e387ebeb22874d0df78035b7f998b",…},…}
```

This is the JSON from their API describing the menu item you just clicked.

Now, add the item to your cart in the UI, selecting the fewest amount of menu items possible (so, only menu items that are “required”). If there are multiple options, select the first ones. Note the additional `/graphql` requests sent, that are fetching additional optionLists for the nested options.

Once you click “Add to cart,” note the `/graphql` request that was sent in the Network tab:



```
content-type: application/json
cookie: optimizelyEndUserId=oeu1594330732067r0.9837201295904483; amplitude_idundefineddoordash.com=eyJvcHRPdXQiOm
dnt: 1
origin: https://www.doordash.com
referer: https://www.doordash.com/store/mendocino-farms-el-segundo-27099/en-US
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: same-origin
user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.428
x-channel-id: marketplace
x-csrftoken: buB6CQfGktUmqSqgs0VAjDOVSbAahhIkOkx0XhHN30JAcFoM05ul3t9jrxn9nQwr
x-experience-id: doordash

▾ {operationName: "addCartItem",…}
   operationName: "addCartItem"
   query: "mutation addCartItem($usesCartService: Boolean!, $addCartItemInput: AddCartItemInput!, $fulfillmentCont
  ▾ variables: {usesOrderService: true, usesCartService: true, includeCompanyPaymentInfo: true,…}
    ▸ addCartItemInput: {storeId: "27099", menuId: "1654311", itemId: "208561505", itemName: "1/2 Bistro Steak Sand
    ▸ cartContext: {usesOrderService: true}
    ▸ fulfillmentContext: {shouldUpdateFulfillment: false, fulfillmentType: "Delivery"}
     includeCompanyPaymentInfo: true
     includeDifferentialPricingDetail: true
     isPickup: false
     itemId: "208561505"
     itemType: "itemV2"
     nestedOptions: "[{"id":"2332496096","quantity":1,"options":[{"id":"2332496975","quantity":1,"options":[],"item
     orderCartId: "0e918147-2598-478f-a044-721f305e40f5"
     quantity: 1
     shouldSetPickup: false
     specialInstructions: ""
     storeId: "27099"
     substitutionPreference: "substitute"
     unitPrice: 1226
     useTipSuggestionsList: false
     usesCartService: true
     usesOrderService: true
```

Now look at where the options we selected UI were described in the api request. They’re in “variables.nestedOptions.” Note the JSON in “nestedOptions”: this will be the output of the function we’re writing.

So, to keep this challenge as simple as possible, let’s assume we have all of the data we need: all of the menu item and options responses nested in one big JSON object (attached as `input.json`), with the subsequent “optionLists” as a property on their respective “option” objects. How would we take the JSON for that big menu item object and generate the minimum set of required options that you see in “nestedOptions” (attached as `output.json`) so ultimately, in this fictional scraping scenario, we can add any item to our cart with our scraping scripts, and eventually get the service fee for any restaurant?

For this programming challenge, make a function that can take the JSON for any menu item in DoorDash, and generate the minimum set of nestedOptions. As a test for the function, it should be able to take the attached `input.json`, and output the attached `output.json`. Also, it’s not necessary to spend more than 2 hours on this. A good explanation on how you would finish if you had the time would be great.

Thanks,
Tucker