Máximo de Árvore

Neste problema, exploraremos encontrar valores máximos em uma estrutura chamada *árvore*.

Para nossos propósitos, uma árvore é descrita como um dicionário com duas chaves (note que a definição é em si recursiva, já que depende na definição de árvore!):

- `value` mapeia a um número, e
- `children` mapeia a uma lista contendo algum número de filhos, cada um sendo uma árvore em si. Essas estruturas, que são bastante utilizadas para uma variedade de aplicações práticas, têm uma representação gráfica também, consistindo de *vértices/nódulos*, cada um contendo um valor numérico e algum número de "filhos". Aqui estão alguns exemplos de árvores, representados tanto quanto dicionários e graficamente:

O dicionário abaixo:

```
{'value': 9, 'children': []}
```

representa a seguinte árvore:

O dicionário abaixo:

```
{'value': 9,
 'children': [{'value': 2, 'children': []},
              {'value': 3, 'children': []},
              {'value': 7, 'children': []}]}
```

representa a seguinte árvore:

O dicionário abaixo:

```
{'value': 13,
 'children': [{'value': 7, 'children': []},
              {'value': 8,
               'children': [{'value': 99, 'children': []},
                            {'value': 16,
                             'children': [{'value': 77, 'children':
[]}]},
                            {'value': 42, 'children': []}]}]}
```

representa a seguinte árvore:

Neste problema, estamos interessados em encontrar o valor máximo em uma dada árvore.

1) Binary Tree Max

Para começar, limitaremos nossa atenção às chamadas *árvores binárias próprias* (árvores nas quais cada subárvore tem 0 filhos ou exatamente 2 filhos). Vamos definir uma função chamada `binary_tree_max(tree)`, que recebe uma única árvore da forma descrita acima como entrada, para encontrar o valor máximo em uma árvore dessa forma.

Conforme descrito nas leituras, podemos tentar pensar em um caso base e um caso recursivo. Aqui, consideraremos que o caso base é o caso de uma árvore não ter filhos (nesse caso, podemos retornar `value` da árvore diretamente).

E no caso recursivo, podemos chamar recursivamente `binary_tree_max` para encontrar o máximo da primeira árvore filha e fazer outra chamada recursiva para encontrar o máximo da segunda árvore filha. Então, sabemos que o máximo da árvore fornecida é o máximo de sua primeira árvore filha, o máximo de sua segunda árvore filha ou seu próprio valor (o que for mais alto dos três).

Escreva a função `binary_tree_max(tree)` usando esta estrutura.

2) Arbitrary Tree Max

Agora, escreva uma função chamada `tree_max(tree)` que tem o mesmo objetivo (encontrar o valor máximo em uma árvore), mas que pode realizar esta operação em uma árvore arbitrária (ou seja, uma árvore onde cada nódulo tem uma quantidade arbitrária de filhos, não necessariamente 0 ou 2).

Você pode achar útil começar com seu código da parte anterior, ao invés de começar do zero (o que precisa mudar em seu código para lidar com árvores arbitrárias?).

Submissão

Quando estiver pronto (depois de ter simulado manualmente e testado em sua própria máquina e estiver convencido de que seu programa fará a coisa certa), faça upload do seu arquivo Python no **Problema 3.6**