Krabičky

Created	@Nov 06, 2019 2:41 PM
Tags	Homework

Zadanie

Máme 2 krabičky s číslami (3, 11) a 4 krabičky s operátormi (+, +, *, *). Našou úlohou je nájsť všetky možnosti zapojenia krabičiek tak, aby OUTPUT bolo číslo 102

Spájanie krabičiek

Číslo

Krabička s číslom má nekonečne veľa výstupov. V praxi to znamená, že to isté číslo môžeme použiť viackrát.

Operátor

Krabička operátoru má práve 2 vstupy = dva čísla, ktoré v našom prípade buď sčíta alebo vynásobí a nekonečne veľa výstupov. To v praxi znamená, že číslo, ktoré je výsledkom nejakej operáce môže byť použité aj v ďalších operáciách. Príklad:

(0): (3, 11 | +, +, *, *) / 3 + 3

(1): (3, 6, 11 | +, *, *)

Sčítali sme čísla 3 a 3, a tak sme sa vlastne dostali do situácie, kedy pracujeme s troma číslami (3, 6, 11) a troma operátorami (+, *, *)

Všetky kombinácie

Začíname v stave (3, 11 | +, +, *, *).

V každej iterácii pracujeme s n číslami a k operátorami.
Ľubovoľne môžeme kombinovať:

{číslo} {operátor} {číslo}

Pričom čísla sa môžu opakovať. To nám v kažej iterácii dáva n * n * k kombinácií, ako sa dostať do ďalšieho stavu.
Toto opakujeme, až kým sa nedostaneme do stavu, kedy už nemáme k dispozícii žiadne operátory.

V prípade, že vyberieme kombináciu, ktorá vytvorí číslo 102, túto vetvu ukončíme a zapamätáme si postup, ako sme sa k tomuto číslu dostali.

Rôzne vetvy sa dostanú k rovnakej konečnej kombinácii (tj. 2 čísla, jeden operátor = 102). Toto je spôsobené tým, že nám niekedy stačia iba 3 kroky a teda jeden krok môže byť ľubovoľný.

JavaScritp code

Podotýkam, že táto funkcia nemá output vo formáte, ktorý by som použil v praxy. Nechcelo sa mi to nejako formátovať, pretože mi stačila schopnosť programubrute-force-núť všetky možnosti. Analýzu možností som už spravil ja podľa outputu v konzole.

```
const removeFirst = (arr, toRemove) => {
  const newArr = [];
  let removed = false;

  for (let i = 0; i < arr.length; i++) {
    if (removed || arr[i] != toRemove) {
      newArr.push(arr[i]);
    } else {
      removed = true;
    }
  }

  return newArr;
};

const unique = arr => Array.from(new Set(arr));
```

```
const run = (options, target, nums, operators, history = []) => {
  unique(operators).forEach(operator => {
    for (let i = 0; i < nums.length; i++) {
      for (let j = i; j < nums.length; j++) {
        const num1 = Math.min(nums[i], nums[j]);
        const num2 = Math.max(nums[i], nums[j]);

        const exp = [num1, num2].join(operator);
        const val = eval(exp);

        const newNums = [...nums, val];
        const newOperators = removeFirst(operators, operator);
        const newHistory = [...history, [exp + ' = ' + val]];

        if (newNums.includes(target)) {
          if (!options.hasOwnProperty(exp)) {
            options[exp] = [];
          }

          options[exp].push(newHistory);
        } else {
          run(options, target, newNums, newOperators, newHistory);
        }
      }
    }
  });

  return options;
};

const options = {};
run(options, 102, [3, 11], ['+', '+', '*', '*']);

Object.entries(options).forEach(([exp, combinations]) => {
  const min = combinations.reduce(
    (min, item) => Math.min(min, item.length),
    Infinity
  );

  // Remove all combinations with an arbitrary operation
  options[exp] = combinations
    .filter(item => item.length == min)
    .map(item => item.join(' | '));
});

console.log(options);
```

Output

```
[wiki@wiki skatule]$ node index.js
{
  '6*17': [ '3+3 = 6 | 6+11 = 17 | 6*17 = 102' ],
  '3+99': [
    '3*3 = 9 | 9*11 = 99 | 3+99 = 102',
    '3*11 = 33 | 3*33 = 99 | 3+99 = 102'
  ],
  '36+66': [
    '3+3 = 6 | 6*11 = 66 | 6*6 = 36 | 36+66 = 102',
    '3+3 = 6 | 6*6 = 36 | 6*11 = 66 | 36+66 = 102'
  ]
}
```

Riešenie

Existujú 3 finálne kominácie:

- 6 * 17
- 3 + 99
- 36 + 66

Pričom číslo 99 vo finálnej kombinácii 3 + 99 sa dá zostaviť dvoma spôsobmi:

- 99 = (3 + 3) * 11
- 99 = (3 * 11) * 3

Existujú teda 4 riešenia: