24 Game

You have 4 cards each containing a number from 1 to 9. You need to judge whether they could operated through `*`, `/`, `+`, `-`, `(`, `)` to get the value of 24.

Example 1:

```
Input: [4, 1, 8, 7]
Output: True
Explanation: (8-4) * (7-1) = 24
```

Example 2:

```
Input: [1, 2, 1, 2]
Output: False
```

Note:

1. The division operator `/` represents real division, not integer division. For example, 4 / (1 - 2/3) = 12.
2. Every operation done is between two numbers. In particular, we cannot use `-` as a unary operator. For example, with `[1, 1, 1, 1]` as input, the expression `-1 - 1 - 1 - 1` is not allowed.
3. You cannot concatenate numbers together. For example, if the input is `[1, 2, 1, 2]`, we cannot write this as 12 + 12.

Solution 1

```
def judgePoint24(self, nums):
    bad = '떢빗각걋냇갅갸꺚뵞숣읤봄뵞뤼갈값뤔떌옊메늪숭캾긶꺚옓갍뇐쪗곴듵갱국옖왹눞삸걃끗긕땉궬가쌀낐걊숝빼늘갠꽸숢걂갋갃쬧곈솣쩡섐쇷꿏뤡간빖쪤웨땬옉뤷갂뵳덦놈빖옜귒늪갰갗놥궬걚옌뼘뫅거갎긽낤겼'
    return chr(int(''.join(map(str, sorted(nums)))) + 42921) not in bad
```

There are really only 495 possible inputs, of which 404 are solvable and 91 aren't. The above is the shortest encoding of those 91 that I could think of. One character for each case. The +42921 is for getting all characters from the same unicode range (from the "Hangul Syllables" range) so that it looks good. For extra style points I shuffled them, otherwise they'd look somewhat sorted.

Edit: Then again, after a few iterations my "normal" solution ended up being *shorter* than this. But at least this is still much faster and imho more fun :-)

written by StefanPochmann original link here

Solution 2

```
class Solution {

    boolean res = false;
    final double eps = 0.001;

    public boolean judgePoint24(int[] nums) {
        List<Double> arr = new ArrayList<>();
        for(int n: nums) arr.add((double) n);
        helper(arr);
        return res;
    }

    private void helper(List<Double> arr){
        if(res) return;
        if(arr.size() == 1){
            if(Math.abs(arr.get(0) - 24.0) < eps)
                res = true;
            return;
        }
        for (int i = 0; i < arr.size(); i++) {
            for (int j = 0; j < i; j++) {
                List<Double> next = new ArrayList<>();
                Double p1 = arr.get(i), p2 = arr.get(j);
                next.addAll(Arrays.asList(p1+p2, p1-p2, p2-p1, p1*p2));
                if(Math.abs(p2) > eps)  next.add(p1/p2);
                if(Math.abs(p1) > eps)  next.add(p2/p1);

                arr.remove(i);
                arr.remove(j);
                for (Double n: next){
                    arr.add(n);
                    helper(arr);
                    arr.remove(arr.size()-1);
                }
                arr.add(j, p2);
                arr.add(i, p1);
            }
        }
    }
}
```

written by zhang00000 original link here

Solution 3

```
class Solution {
public:
    bool judgePoint24(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        do {
            if (valid(nums)) return true;
        } while(next_permutation(nums.begin(), nums.end()));
        return false;
    }
private:
    bool valid(vector<int>& nums) {
        double a = nums[0], b = nums[1], c = nums[2], d = nums[3];
        if (valid(a+b, c, d) || valid(a-b, c, d) || valid(a*b, c, d) || valid(a/b, c
, d)) return true;
        if (valid(a, b+c, d) || valid(a, b-c, d) || valid(a, b*c, d) || valid(a, b/c
, d)) return true;
        if (valid(a, b, c+d) || valid(a, b, c-d) || valid(a, b, c*d) || valid(a, b,
c/d)) return true;
        return false;
    }
    bool valid(double a, double b, double c) {
        if (valid(a+b, c) || valid(a-b, c) || valid(a*b, c) || b&&valid(a/b, c)) ret
urn true;
        if (valid(a, b+c) || valid(a, b-c) || valid(a, b*c) || c&&valid(a, b/c)) ret
urn true;
        return false;
    }
    bool valid(double a, double b) {
        if (abs(a+b-24.0) < 0.0001 || abs(a-b-24.0) < 0.0001 || abs(a*b-24.0) < 0.00
01 || b&&abs(a/b-24.0) < 0.0001)
            return true;
        return false;
    }
};
```

written by zestypanda original link here