


# GitHub commit log ( github.com )

 Search or jump to... / Pull requests Issues Marketplace Explore

pattasai-wsu / primefunctions Private

Unwatch 1 Star 0 Fork 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main

Commits on Dec 8, 2020

Adding my images of the testsuites to the git repo

pattasai-wsu committed 6 minutes ago

8c1ab6f <>

optimization finished

pattasai-wsu committed 22 minutes ago

0bfbcab <>

optimization 2

pattasai-wsu committed 33 minutes ago

babd441 <>

Commits on Dec 7, 2020

added some comments about optimizations 1 changes

pattasai-wsu committed 22 hours ago

8c1b0bf <>

optimization 1 - about 10ms better on 10000 and 500-1000ms better on ...

pattasai-wsu committed 23 hours ago

38fe408 <>

at about 67ms for 10000 and still timing out for 100000

pattasai-wsu committed yesterday

23da88c <>

Finished speed tests

pattasai-wsu committed yesterday

ac83605 <>

working on getting the timeout code correct

pattasai-wsu committed yesterday

f4c4c3e <>

uncomment then comment out the same statement

pattasai-wsu committed yesterday

07758a9 <>

Commits on Nov 25, 2020

took out testing for array length and cleaned up text output

pattasai-wsu committed 13 days ago

adbcc6a <>

finished maxPrimeSum tests

pattasai-wsu committed 13 days ago

f06615a <>

finished cumulativeSum tests

pattasai-wsu committed 13 days ago

77987a5 <>

finished primeGen tests

pattasai-wsu committed 13 days ago

b7d7a30 <>

commented out some code for testing purposes

pattasai-wsu committed 13 days ago

f6f3ecd <>

# GitHub commit log ( desktop app )

File Edit View Repository Branch Help

Current repository  
primefunctions

Current branch  
main

Fetch origin  
Last fetched 12 minutes ago

Changes

History

Select branch to compare...

Adding my images of the testsuites to the git repo  
Patrick Tsai • 12m

optimization finished  
Patrick Tsai • 27m

optimization 2  
Patrick Tsai • 38m

added some comments about optimizations 1 changes  
Patrick Tsai • 22h

optimization 1 - about 10ms better on 10000 and 500-1000ms better on 100000  
Patrick Tsai • 23h

at about 67ms for 10000 and still timing out for 100000  
Patrick Tsai • 1d

Finished speed tests  
Patrick Tsai • 1d

working on getting the timeout code correct  
Patrick Tsai • 1d

uncomment then comment out the same statement  
Patrick Tsai • 1d

took out testing for array length and cleaned up text output  
Patrick Tsai • Nov 25, 2020

finished maxPrimeSum tests  
Patrick Tsai • Nov 25, 2020

finished cumulativeSum tests  
Patrick Tsai • Nov 25, 2020

finished primeGen tests  
Patrick Tsai • Nov 25, 2020

commented out some code for testing purposes  
Patrick Tsai • Nov 25, 2020

Finished assignment - fixed cumulativeSum output  
Patrick Tsai • Nov 1, 2020

Finished maxPrimeSum  
Patrick Tsai • Oct 29, 2020

update to maxPrimeSum but not finished  
Patrick Tsai • Oct 28, 2020

optimization finished

Patrick Tsai • 1 changed file

New

primefunctions2.js

69

69

70

70

71

71

72

72

73

73

74

74

75

75

76

76

77

77

78

78

79

79

80

80

81

81

82

82

83

83

84

84

85

85

86

86

87

87

88

88

89

89

90

90

91

91

92

92

93

93

94

94

95

95

96

96

97

97

136

136

137

137

138

138

139

139

116

116

117

117

118

118

119

119

120

120

121

121

122

122

123

123

@@ -69,29 +69,6 @@ function cumulativeSum(numListIn) {  
 return (this.numList);  
}  
  
/\*  
-I have changed just about everything from my first NON-Optimized code  
-My first optimization was about as good, if not a little better, than this code  
-However, I like this code because it's a little more straight forward  
-Both are anywhere from .5 to 2 seconds faster than the original code,  
-I'm not sure why, but sometimes it's much faster, and when I rerun the code  
-it slower.  
-  
-Changes from HW4:  
-1. HW 4 I used a large array to keep track of every [prime\_value, consecutive\_sum]  
- This code, I use an array but only have one entry for the largest [value, consec]  
- and then I replace if a larger one occurs  
-2. In HW5 maxPrimeSum code I use the cumulativeSum function, I did not in HW4  
-3. HW5 I only check odd values from the cumulativeSum array, against the array from primeGen  
- HW4 I checked every value  
-4. HW5 code, I only check numbers in my cumulativeSum array that are smaller than the  
- largest prime number in my primeGen array  
-5. HW5, I use an offset to not check primeGen values I've already checked.  
-  
-Honestly, I thought these improvements would be much faster. I feel like 10,000 and below  
-my code is fast, but when I go for 100,000 it feels sluggish.  
- \*/  
-  
// eslint-disable-next-line no-unused-vars  
function maxPrimeSum(n) {  
 this.array1 = primeGen(n);  
  
 @@ -136,4 +113,26 @@ function maxPrimeSum(n) {  
 return (this.keeper[0]);  
 }  
 -console.log(maxPrimeSum(100000)); @+  
+  
+/\*  
+I have changed just about everything from my first NON-Optimized code  
+My first optimization was about as good, if not a little better, than this code  
+However, I like this code because it's a little more straight forward  
+Both are anywhere from .5 to 2 seconds faster than the original code,  
+I'm not sure why, but sometimes it's much faster, and when I rerun the code  
+it slower.  
+ \*/  
+ \*/