

2396. Strictly Palindromic Number

Medium

Brainteaser

Math

Two Pointers

Leetcode Link

Problem Description

An integer is classified as **strictly palindromic** if it adheres to a special rule regarding its representation in different number bases. Specifically, a number n is strictly palindromic if, for every base b starting from 2 up to $n - 2$, its representation in that base is a palindrome. It's important to note that a palindrome reads the same forward and backward. For instance, the string "121" is a palindrome because it remains "121" even when reversed.

The task is to create a function that takes an integer n and returns `true` if n is strictly palindromic or `false` otherwise. This means the function needs to verify if n is a palindrome in all the bases from 2 to $n - 2$. The challenge is to figure out a way to evaluate this condition effectively or to determine if such a number can exist.

Intuition

Upon initial consideration, you might think that to determine if a number is strictly palindromic, you'd need to convert the number into all the base representations from 2 to $n-2$ and check if it's a palindrome in each. This would indeed be the brute-force approach.

However, a closer examination of the problem can lead to the conclusion that such a number cannot exist. Intuitively, the notion of a strictly palindromic number is flawed because when a number n is represented in base $n-1$, the result would always be `11` (since n base $n-1$ is the same as saying $n-1$ times `1` with a remainder of `1`, which is how base conversion works). This representation of the number n as `11` in base $n-1$ is palindromic, but that is just for this single base.

When you get to base $n-2$, the hypothesis falls apart. Regardless of what the number n is, it cannot continue to be palindromic in each and every base leading down to 2 from $n-2$. This is due to the way numbers scale in different bases, which inevitably leads to different least significant digits as you go down in base, disrupting the palindromic nature.

Given these considerations, it turns out that the solution to the problem is quite simple: no number greater than `3` can satisfy the condition of being strictly palindromic because it will fail the test when n is expressed in base $n-2$. The function, therefore, correctly always returns `False`, simplifying the solution significantly.

Solution Approach

The implementation of the solution does not require complex algorithms, data structures, or patterns. It is based on the mathematical proof that there are no strictly palindromic numbers beyond `3` due to the nature of base conversion, particularly when n is represented in base $n - 2$.

Exploring different bases:

- When converting a number to a base that is one less than the number itself (base $n - 1$), the number is always represented as `11`. This is palindromic, but it only works for that specific base.
- Reducing the base further to $n - 2$, the representation changes and cannot maintain the palindromic property across all lower bases down to `2`.

Given this understanding, the reference solution's approach is straightforward:

The `isStrictlyPalindromic` function in Python is defined with a single line of code:

```
1 class Solution:
2     def isStrictlyPalindromic(self, n: int) -> bool:
3         return False
```

This function does not require any loops, checks, or base conversions because it leverages the knowledge that no number $n > 3$ can ever be strictly palindromic. Thus, the solution is exceptionally efficient with a constant time complexity of $O(1)$ - it will always return `False` immediately, regardless of the value of n .

Example Walkthrough

Let's pick an integer n and walk through the solution approach using this example. Suppose we choose $n = 10$. We need to determine if 10 is strictly palindromic, which means it must be a palindrome from base 2 up to base $n - 2$, which would be base `8` in this case.

- For base `2`, the representation of 10 is `1010`, which is not palindromic.
- For base `3` to base `7`, the representation will change for each base, and while we might find it palindromic in one or a few bases, that is irrelevant because we need it to be palindromic in all bases from `2` to $n - 2$.
- For base $n - 1$, which is base `9` for our example, the representation of `10` would be `11`, as `10` in base `9` equals `9 + 1`. This is indeed palindromic but again does not fulfill the strict condition required.
- Finally, for base $n - 2$, which is base `8` in our case, the representation of `10` would no longer be `11`. In base `8`, `10` is equal to `12`, because `10` in base `8` equals `8 + 2`. This violates the condition to be strictly palindromic as it is not the same forward and backward.

From this example, we can observe that as we change the base, the representation of the number changes and does not remain palindromic in all bases leading up to $n - 2$. Hence, by similar logic, we can generalize that for any number $n > 3$, it will not remain strictly palindromic across all the bases from 2 to $n - 2$. The solution below reflects this understanding:

```
1 class Solution:
2     def isStrictlyPalindromic(self, n: int) -> bool:
3         return False
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

Java Solution

```
1 class Solution {
2
3     // This method checks if an integer n is "strictly palindromic"
4     // According to the current definition provided, it returns false for all inputs
5     public boolean isStrictlyPalindromic(int n) {
6         // Always returns false as no integer can be strictly palindromic by the problem's original statement
7         return false;
8     }
9 }
10
```

C++ Solution

```
1 class Solution {
2 public:
3     // Function to determine if a number is strictly palindromic
4     // Since the problem states to convert the number into all bases
5     // between 2 and n - 2 and check if they are palindromes, and no number
6     // can satisfy this condition, the function always returns false.
7     bool isStrictlyPalindromic(int n) {
8         // The implementation of the logic that eventually leads to the conclusion that
9         // no number can satisfy the strictly palindromic condition is omitted.
10        // Therefore, the function simply returns false.
11        return false;
12    }
13 };
14
```

Typescript Solution

```
1 /**
2  * Checks if a number is strictly palindromic.
3  * A number is strictly palindromic if it forms a palindrome in every base from 2 up to n - 1
4  *
5  * @param {number} n - The number to check.
6  * @returns {boolean} - A boolean indicating if the number is strictly palindromic or not.
7  */
8 function isStrictlyPalindromic(n: number): boolean {
9     // Iterate through all the bases from 2 to n - 1
10    for (let base = 2; base < n; base++) {
11        // Convert the number to the current base
12        const numberInBase = n.toString(base);
13
14        // Check if the converted number is a palindrome
15        if (!isPalindrome(numberInBase)) {
16            return false;
17        }
18    }
19
20    // If number n is palindromic in all bases from 2 to n - 1, return true
21    return true;
22 }
23
24 /**
25  * Checks if a string is a palindrome.
26  *
27  * @param {string} str - The string to check.
28  * @returns {boolean} - A boolean indicating whether the string is a palindrome or not.
29  */
30 function isPalindrome(str: string): boolean {
31    // Define pointers for the start and end of the string
32    let start = 0;
33    let end = str.length - 1;
34
35    // Check if the string is a palindrome by comparing characters from both ends
36    while (start < end) {
37        if (str[start] !== str[end]) {
38            return false;
39        }
40        start++;
41        end--;
42    }
43
44    // If all characters match, the string is a palindrome
45    return true;
46 }
47
```

Time and Space Complexity

The given code is a Python method within a class `Solution` that always returns `False` regardless of the input n .

Time Complexity: $O(1)$

The time complexity of the method `isStrictlyPalindromic` is constant time $O(1)$, because the function performs a single operation (returning `False`) regardless of the size or value of the input parameter n .

Space Complexity: $O(1)$

The space complexity of the method is also constant $O(1)$. No additional space is used that grows with the input size; the only space used is for the input itself and the return value, which do not depend on the value or size of n .