Task MEDIAN

The 10 test cases for MEDIAN have been designed to detect performance differences as exhibited by 16 different algorithms (also see below):

- OPE = Onion Peeling Eliminiation
- LISF = Linear Insertion Sort Using Full List
- LISH = Linear Insertion Sort Using Half List
- LISZ = Linear Insertion Sort Using Zoom List
- BISF = Binary Insertion Sort Using Full List
- BISH = Binary Insertion Sort Using Half List
- BISZ = Binary Insertion Sort Using Zoom List
- TISF = Ternary Insertion Sort Using Full List
- TISH = Ternary Insertion Sort Using Half List
- TISZ = Ternary Insertion Sort Using Zoom List
- TPFS = Ternary Partioning Find Using Straddled Pivots
- TPFF = Ternary Partioning Find Using First Pivots
- TPFP = Ternary Partioning Find Using Proportional Pivots
- TPFR = Ternary Partioning Find Using Random Pivots
- SLSB = Sorted List of Sorted Buckets
- HTSB = Heap-like Tree of Sorted Buckets

The next table shows how many calls each algorithm made for each test case solved within the bound of 7777 calls. The rightmost column shows the score.

Case #	1	2	3	4	5	6	7	8	9	10	
N	5	177	577	975	1087	1267	1357	1415	1415	1499	
Cat	M	R	N	R	R	R	R	R	A	R	Pts
Alg											
OPE	4	7744									20
LISF	4	4062	619								30
LISH	3	2590	598								30
LISZ	3	2160	598								30
BISF	4	861	4175	7051							40
BISH	5	843	4108	6803							40
BISZ	4	730	3621	6269	7078						50
TISF	3	712	2918	5415	6143	7376					60
TISH	3	669	2707	5349	6011	7103	7642				70
TISZ	3	609	2537	4889	5540	6641	7191	7511	7572		90
TPFS	3	517		1525	2842	3257	3531	2231		3218	80
TPFF	4	395		2205	2378	3635	3601	2663		2493	80
TPFP	5	331	848	3512	1705	2291	3093	2860	2863	2985	100

Alg	1	2	3	4	5	6	7	8	9	10	1499
TPFR	4	372	1778	2201	2507	2981	3377	3987	3279	3540	100
SLSB	4	491	1954	3242	3605	4258	4578	4824	4149	5147	100
HTSB	4	508	2218	3184	3902	4517	4862	5074	4389	5354	100

The 10 test cases belong to 4 categories:

- M = Manually designed
- R = Randomly generated
- N = Nearly sorted
- A = Alternating outside-to-inside (1 3 5 ... 6 4 2)

Here is a similar table showing the number of calls for cases where the algorithm FAILS (does not stay within the bound). When the number of calls exceeds 9999, only an approximate value in "scientific notation" is given, where >XeY means that the number of calls exceeds X*10^Y, but does not exceed (X+1)*10^Y. One extra column has been added on the right. It indicates whether for N = 1499 and under worst-case conditions (W), the algorithm stays within the bound of 7777 (shown as <=) or not (shown as >). The library, however, is not able to create such worst-case conditions dynamically.

Case #	1	2	3	4	5	6	7	8	9	10	
N	5	177	577	975	1087	1267	1357	1415	1415	1499	1499
Cat	M	R	N	R	R	R	R	R	A	R	W
Alg											
OPE			>8e4	>2e5	>2e5	>4e5	>4e5	>4e5	>4e5	>5e5	>
LISF				>1e5	>1e5	>1e5	>2e5	>2e5	>2e5	>2e5	>
LISH				>7e4	>8e4	>1e5	>1e5	>1e5	>1e5	>1e5	>
LISZ				>5e4	>7e4	>1e5	>1e5	>1e5	>6e4	>1e5	>
BISF					7791	9532	>1e4	>1e4	>1e4	>1e4	>
BISH					7811	9414	>1e4	>1e4	>1e4	>1e4	>
BISZ						8537	9299	9803	>1e4	>1e4	>
TISF							7981	8386	8339	8993	>
TISH								7980	7946	8519	>
TISZ										8032	>
TPFS			>7e4						>1e4		>
TPFF			>5e4						>1e4		>
TPFP											>
TPFR											>
SLSB											<=

```
----   + ----  ----  ----  ----  ----  ----  ----  ----  ----  ---- + ---
HTSB   |                                                            |  <=
----   + ----  ----  ----  ----  ----  ----  ----  ----  ----  ---- + ---
```

Information about the algorithms

OPE: Repeatedly eliminate the two extremes (min and max strength). This takes (N-1)^2 / 4 calls.

All Insertion Sort methods: Maintain a sorted list of objects investigated so far (sorted modulo up or down) and repeatedly insert a next object. The location to insert can be found by linear, binary, or ternary search. Linear insertion is quadratic in both worst and average cases, and linear in best cases. Binary and ternary insertion have N*log N complexity (ternary has smaller constant factor).

Instead of maintaining the Full list (LISF, BISF, TISF) containing all the objects in the end, it is enough to limit the list to contain no more than half the number of objects (Half List: LISH, BISH, TISH). Reason: after having considered (N+1)/2 objects, the element at the end of the list cannot be the median, because more than (N-1)/2 objects are stronger/weaker than this object.

In fact, both extremes in the sorted list can be eliminated once (N+1)/2 objects have investigated (Zoom List: LISZ, BISZ, TISZ). That way, the list increases in length during the first half, and descreases in length during the second half, until only one candidate remains (which then must be the median); it zooms out and then in on the median.

All Partitioning Find methods: Compare to median selection by partitioning (as in QuickSort, discarding the segment that is known not to contain the median). Only partitioning into three parts (based on choosing two pivot objects) have been considered. In general these methods are quadratic in worst case, but linear in average and best case. There are various ways to choose the pivots: one at each end (Straddled: TPFS), both at one end (First: TPFF), at one third and two thirds in the list (Proportional: TPFP), and Random (TPFR). For TPFS and TPFF, the sorted input is bad, but for TPFP and TPFR it is (very) good. TPFR has no specific worst case inputs. Worst case input for TPFP depends on details of rounding when choosing the proportional pivots.

SLSB: Maintains buckets of at most K objects (for some K; K=8 is a good choice). Each bucket is sorted (with respect to the order of two reference objects), and the list of buckets is sorted on the minimum of the buckets (w.r.t. the same reference objects). Compared to insertion sort into a list of single objects (Full, Half, or Zoom) this saves calls (over 2000 in the worst case of the task), because only a partial order instead of a total order is constructed. You can calculate the number of calls in the worst case for N=1499, and it is just below 7777. Average case behavior is better than worst case.

HTSB: This takes the idea of SLSB one step further by maintaining the buckets in a heap-like leaftree. The data structure is more complicated, and this method is not needed to obtain a perfect score. It shows that more advanced data structures can do even better, not only on average but also in the worst case. Note that the advantage is not visible for "small" N (such as 1499) on random cases.