# Increasing sequence

Short formulation. The number sequence is given. Your task is to construct the increasing sequence that approximates the given one in the best way. The best approximating sequence is the sequence with the least total deviation from the given sequence.

More precisely. Let $t_1, t_2, \ldots, t_N$ is the given number sequence. Your task is to construct the number sequence $z_1, z_2, \ldots, z_N$ satisfying to the next conditions:

1. $z_1 < z_2 < \ldots < z_N$
2. The sum $|t_1 - z_1| + |t_2 - z_2| + \ldots + |t_N - z_N|$ should be a minimal feasible.

**Input format.** There is the integer N (1<=N<=1000000) in the input file first line. Each of the next N lines contains single integer – the given sequence element. There is $t_K$ in the (K+1)-th line. Any element is satisfying to relation $0 <= t_K <= 2000000000$.

**Output format.** The first line must contain the single integer – the minimal possible total deviation. Each of the next N lines contains single integer – the recurrent element of the best approximating sequence.
You must output any one from the sequences with a least total deviation.

**Sample.**

| Input file | Output file |
|---|---|
| 7 | 13 |
| 9 | 6 |
| 4 | 7 |
| 8 | 8 |
| 20 | 13 |
| 14 | 14 |
| 15 | 15 |
| 18 | 18 |

# Solution

1.

Call the sequence ($y_1$, $y_2$, … $y_N$) *nondecreasing* if its elements satisfying to condition $y_1 \le y_2 \le \ldots \le y_N$.

Let determine the one-to-one correspondence between increasing and nondecreasing sequences by the next way:

$z_i = y_i + i$, i= 1, 2, …, N,

and correspondently,

$y_i = z_i - i$, i= 1, 2, …, N

It's clear if $(z_i)_{i=1\div N}$ is an increasing sequence, then $(y_i)_{i=1\div N}$ is a nondecreasing sequence and vice versa.

Reduce the source problem to the problem of the best nondecreasing sequence construction. Note $\sum_{i=1}^{N} |t_i - z_i| = \sum_{i=1}^{N} |t_i - (y_i + i)| = \sum_{i=1}^{N} |(t_i - i) - y_i|$.

Therefore, we can operate in this way:

A. change the sequence $t_i$ onto the sequence $x_i = t_i - i$, i = 1, 2, …, N
B. search the nondecreasing sequence $(y_i)_{i=1\div N}$ so that the sum $\sum_{i=1}^{N} |x_i - y_i|$ should be a least feasible
C. compute $z_i = y_i + i$, i= 1, 2, …, N

Definition. The *decision sequence* is any of such sequences $(y_i)_{i=1\div N}$.

2.

Lemma 1. Let $a \le b$. Then $\min\limits_{y} (|y-a| + |y-b|) = |b-a|$ and it is reached for all $a \le y \le b$.

The proof is trivial.

Lemma 2. Let $a_1 \le a_2 \le \ldots \le a_n$. Then the sum $\sum_{i=1}^{n} |y - a_i|$ is reached to its minimum for

a). all $a_k \le y \le a_{k+1}$ if n=2k b). $y = a_k$ if n=2k-1

Proof.

$$\sum_{i=1}^{n} |y-a_i| = (|y-a_1| + |y-a_n|) + (|y-a_2| + |y-a_{n-1}|) + (|y-a_3| + |y-a_{n-2}|) + \ldots$$

The conclusion of the Lemma follows from it and the Lemma 1 immediately.

Definition. The *median* of the numbers sequence ($x_1$, $x_2$, … $x_n$) is the value of the member placed in the center in the nondecreasing order.

More precisely, let ($a_1$, $a_2$, … $a_n$ ) is the permutation of the sequence ($x_1$, $x_2$, … $x_n$) with $a_1 \le a_2 \le \ldots \le a_n$; the median of the sequence ($x_1$, $x_2$, … $x_n$) is the value of the $a_{(n+1) \operatorname{div} 2}$ : median($x_1$, $x_2$, … $x_n$) = $a_{(n+1) \operatorname{div} 2}$ .

<u>Corollary.</u> The sum $\sum_{i=1}^{n}|y-x_i|$ is reached to its minimum by y = median($x_1$, $x_2$, … $x_n$ ).

3.
Analyze the sequence $(y_i)_{i=1\div N}$ . Choose the sequence segment $y_p$, $y_{p+1}$, …, $y_q$ so that $y_{p-1} < y_p = y_{p+1} = … = y_q < y_{q+1}$ (let $y_0=-\infty$, $y_{N+1}=+\infty$).

Designate $y_p$ = y. Then the sum $\sum_{i=p}^{q}|y_i-x_i|$ = $\sum_{i=p}^{q}|y-x_i|$ is reached to its minimum by

y = median($x_p$, $x_{p+1}$, … $x_q$).

Hence, we have two statements:

<u>Lemma 3.</u> It's possible to construct the decision sequence that the each its element is equal to some element of the sequence $(x_i)_{i=1\div N}$.

<u>Lemma 4.</u> It's possible to construct the decision sequence $(y_i)_{i=1\div N}$ of the such kind $y_1 = y_2 = \ldots = y_{q_1} < y_{q_1+1} = y_{q_1+2} = \ldots = y_{q_2} < \ldots < y_{q_{R-1}+1} = y_{q_{R-1}+2} = \ldots = y_{q_R}$ ($q_R = N$) with $y_{q_i} = median(x_{q_{i-1}+1},\ x_{q_{i-1}+2}, \ldots, x_{q_i})$ .

<u>Definition.</u> The decision sequence of this kind is called by *canonical*.

4. First solution (mathematical induction style).

A. Assume the canonical decision sequence for the sequence ($x_1$, $x_2$, … $x_k$) is constructed, i. e. the numbers $q_1 < q_2 <… q_{S-1} < q_S = k$ and $y_{q_1} < y_{q_2} < \ldots < y_{q_S}$ are obtained.

B. Let analyze the sequence ($x_{k+1}$, $x_{k+2}$, … $x_m$). Assume the corresponding canonical decision sequence consists of one segment – all its members are equal to each other and equal to *Y=median ($x_{k+1}$, $x_{k+2}$, … $x_m$)*.

C.

- If $y_{q_S} < Y$, then set $q_{S+1}$=m, $y_{q_{S+1}} = Y$ and canonical decision sequence for ($x_1$, $x_2$, … $x_m$) with m>k is obtained.
- If $y_{q_S} = Y$, then substitute $q_S$ on m, and canonical decision sequence for ($x_1$, $x_2$, … $x_m$) with m>k is obtained too.
- If $y_{q_S} > Y$, then merge two last segments: suppose $Y = median(x_{q_{S-1}+1}, \ldots, x_k, x_{k+1}, \ldots, x_m)$ and decrease S on 1. Simultaneously rearrange the numbers $x_{q_{S-1}+1}, \ldots, x_k, x_{k+1}, \ldots, x_m$ in the nondecreasing order. It makes possible to calculate the median Y. Each segment of the constructed canonical decision sequence has been sorting; therefore, the MergeSort procedure is applied to merge two segments. The obtained modification of the elements order doesn't influence onto the next steps.
- If S>0, then we turn out to item B. – jump to B.;
  if S=0, then canonical decision sequence for ($x_1$, $x_2$, … $x_m$) is constructed – all

its members are equal to each other and equal to median($x_1, x_2, \dots x_m$).

In all, the present algorithm is obtained:

Algorithm 1.

```
for k = 1, 2, …, (N-1) sequentially
  1. setting m=k+1,
  2. construct the canonical decision sequence
     for ($x_1$, $x_2$, … $x_m$) as described above.
```

The complexity of the presented algorithm is $O(N^2)$ with memory requirement $O(N)$.

## 5. Second solution (Quick-approach based).

Set

$L(b_1, b_2, \dots, b_P)$ the number of the elements $b_i$ of the array $(b_1, b_2, \dots, b_P)$ such that $b_i <$ *median* $(b_1, b_2, \dots, b_P)$;
$LE(b_1, b_2, \dots, b_P)$ the number of the elements $b_i$ of the array $(b_1, b_2, \dots, b_P)$ such that $b_i \leqslant$ *median* $(b_1, b_2, \dots, b_P)$;
$G(b_1, b_2, \dots, b_P)$ the number of the elements $b_i$ of the array $(b_1, b_2, \dots, b_P)$ such that $b_i >$ *median* $(b_1, b_2, \dots, b_P)$;
$GE(b_1, b_2, \dots, b_P)$ the number of the elements $b_i$ of the array $(b_1, b_2, \dots, b_P)$ such that $b_i \geqslant$ *median* $(b_1, b_2, \dots, b_P)$;

Designate $Y$ = median($x_1, x_2, \dots, x_N$).

Lemma 5. Let the array $(x_1, x_2, \dots, x_N)$ could be split on two segments $(x_1, x_2, \dots, x_K)$ and $(x_{K+1}, x_{K+2}, \dots, x_N)$ so that

i. $L(x_P, x_{P+1}, \dots, x_K) > GE(x_P, x_{P+1}, \dots, x_K)$ for all P, $1 \leqslant P \leqslant K$;
ii. $L(x_{K+1}, x_{K+2}, \dots, x_Q) \leqslant GE(x_{K+1}, x_{K+2}, \dots, x_Q)$ for all Q, $K < Q \leqslant N$.

Then

$y_K < Y$ for any decision sequence $(y_1, y_2, \dots y_N)$,
and
we can find the decision sequence with $y_{K+1} \geqslant Y$.

Proof.
Let us assume that $(y_1, y_2, \dots y_N)$ is the some decision sequence with $y_K > Y$.
Assume $y_{K-1} \geqslant Y, y_{K-2} \geqslant Y, \dots, y_P \geqslant Y$, and $y_{P-1} < Y$ (supposing $y_0 = -\infty$).
Let decrease by 1 all numbers $y_K, y_{K-1}, \dots, y_P$.
If $x_i < Y$ $(P \leqslant i \leqslant K)$, then $|y_i - x_i| = y_i - x_i$ is decreasing by 1,
if $x_i \geqslant Y$ $(P \leqslant i \leqslant K)$, then $|y_i - x_i|$ is either increasing or decreasing by 1.
Therefore, the increment of the sum $\sum_{i=P}^{K} |y_i - x_i|$ is at most

$$GE(x_P, x_{P+1}, \dots, x_K) - L(x_P, x_{P+1}, \dots, x_K),$$

and $GE(x_P, x_{P+1}, \dots, x_K) - L(x_P, x_{P+1}, \dots, x_K) < 0$ because of condition i.

If $y_K = Y$ now, then the process is stopped, in case $y_K > Y$, we are repeating the process – find P such as $y_{K-1} \geqslant Y, y_{K-2} \geqslant Y, \dots, y_P \geqslant Y$, and $y_{P-1} < Y$ once more, and decrease by 1 all numbers $y_K, y_{K-1}, \dots, y_P$. It is clear that process shall be stopping after a few steps. The sum $\sum_{i=1}^{N}|y_i - x_i|$ is non-increasing in this process, so the new sequence ($y_1$, $y_2$, … $y_N$) is the decision sequence too.

There is $y_K = y_{K-1} = \ldots = y_P = Y > y_{P-1}$ in the anew constructed decision sequence. Let decrease by 1 all the numbers $y_K$, $y_{K-1}$, …, $y_P$ once more. After that the sum $\sum_{i=P}^{K}|y_i - x_i|$ is decreasing by $L(x_P, x_{P+1}, \ldots, x_K) - GE(x_P, x_{P+1}, \ldots, x_K) > 0$. It contradicts to the assumption that the initial sequence is the decision sequence. The obtained contradiction proves the first conclusion of the Lemma.

The second conclusion is proved analogously. Write this proof for accuracy.
Let we have some decision sequence ($y_1$, $y_2$, … $y_N$) with $y_{K+1}<Y$.
Assume $y_{K+2}<Y$, …, $y_Q<Y$, and $y_{Q+1}\geqslant Y$.
Let increase by 1 all numbers $y_{K+1}$, $y_{K+2}$, …, $y_Q$.
If $x_i\geqslant Y$ ($K<i\leqslant Q$), then $|y_i - x_i| = y_i - x_i$ will decrease by 1,
if $x_i<Y$ ($K<i\leqslant Q$), then $|y_i - x_i|$ either increase or decrease by 1.
Therefore, the increment of the sum $\sum_{i=P}^{K}|y_i - x_i|$ is at most

$$L(x_{K+1}, x_{K+2}, \ldots, x_Q) - GE(x_{K+1}, x_{K+2}, \ldots, x_Q).$$

This difference is nonpositive because of condition ii.

If $y_{K+1}\geqslant Y$ now, then the process is stopped, in case $y_{K+1}<Y$, we are repeating the process – find Q such as $y_{K+1}<Y$, $y_{K+2}<Y$, …, $y_Q<Y$, and $y_{Q+1}\geqslant Y$ once more, and increase by 1 all numbers $y_{K+1}$, $y_{K+2}$, …, $y_Q$. It is clear that process shall be stopping after a few steps. The sum $\sum_{i=1}^{N}|y_i - x_i|$ is nonincreasing in this process, so the new sequence ($y_1$, $y_2$, … $y_N$) is the decision sequence too.

Lemma 6. Set $D(P) = L(x_1, x_2, \ldots, x_P) - GE(x_1, x_2, \ldots, x_P)$, $1\leqslant P\leqslant N$.
Let $P_0$ is the minimal P such as D(P) achieves to its maximal value: $D(P_0)\geqslant D(P)$, $1\leqslant P\leqslant N$ and $D(P_0) = D(P) \Rightarrow P_0 < P$.
Then the array ($x_1$, $x_2$, …, $x_N$) can be split on two segments in the same way as described in the Lemma 5 conditions if and only if $D(P_0) > 0$ where $K= P_0$.

Proof.
If $L(x_P, x_{P+1}, \ldots, x_K)\leqslant GE(x_P, x_{P+1}, \ldots, x_K)$ for some P, $1<P\leqslant K$, then
$D(P-1)=D(K) - (L(x_P, x_{P+1}, \ldots, x_K)-GE(x_P, x_{P+1}, \ldots, x_K)) \geqslant D(K)$ and $P-1<K$.
Moreover, $L(x_P, x_{P+1}, \ldots, x_K) > GE(x_P, x_{P+1}, \ldots, x_K)$ for P=1 because of
$L(x_1, x_2, \ldots, x_K) - GE(x_1, x_2, \ldots, x_K) = D(K) >0$.

If $L(x_{K+1}, x_{K+2}, \ldots, x_Q) > GE(x_{K+1}, x_{K+2}, \ldots, x_Q)$ for some Q, $K<Q\leqslant N$, then
$D(Q) = D(K) + (L(x_{K+1}, x_{K+2}, \ldots, x_Q) - GE(x_{K+1}, x_{K+2}, \ldots, x_Q)) > D(K)$.

Corollary. If the array ($x_1$, $x_2$, …, $x_N$) could be split on two segments in the same way as described in the Lemma 5 conditions, then $K\leqslant N-2$.

Proof.
$D(N) = L(x_1, x_2, \ldots, x_N) - GE(x_1, x_2, \ldots, x_N) < 0$ by the median definition.
It is clear that $|D(N) - D(N-1)| = 1$, hence $D(N-1) \leqslant 0$.

Lemma 7. If the array $(x_1, x_2, \ldots, x_N)$ couldn't be split in the way described in the Lemma 5 conditions, and
the array $(x_1, x_2, \ldots, x_N)$ could be split on to two segments $(x_1, x_2, \ldots, x_M)$ and $(x_{M+1}, x_{M+2}, \ldots, x_N)$, $M \leqslant N$, so that

iii. $LE(x_P, x_{P+1}, \ldots, x_M) \geqslant G(x_P, x_{P+1}, \ldots, x_M)$ for all P, $1 \leqslant P \leqslant M$;
iv. $LE(x_{M+1}, x_{M+2}, \ldots, x_Q) \leqslant G(x_{M+1}, x_{M+2}, \ldots, x_Q)$ for all Q, $M < Q \leqslant N$

then we can construct the decision sequence with $y_1 = y_2 = \ldots = y_M = Y$.

Proof.
First prove the existence of the decision sequence with $y_{M+1} \geqslant Y$ and $y_M \leqslant Y$ similarly to Lemma 5.

Moreover, we know that the array $(x_1, x_2, \ldots, x_N)$ couldn't be split in the way described in the Lemma 5 conditions. It means (corresponding to Lemma 6) that for all P, $1 \leqslant P \leqslant N$, $D(P) \leqslant 0$, i.e. $L(x_1, x_2, \ldots, x_P) \leqslant GE(x_1, x_2, \ldots, x_P)$.

Suppose $y_1 < Y$, $y_2 < Y$, …, $y_R < Y$ and $y_{R+1} \geqslant Y$ .
Let increase by 1 all numbers $y_1, y_2, \ldots, y_R$.

If $x_i < Y$ $(1 \leqslant i \leqslant R)$ then $|y_i - x_i| = y_i - x_i$ either increase or decrease by 1,
if $x_i \geqslant Y$ $(1 \leqslant i \leqslant R)$ then $|y_i - x_i|$ will decrease by 1.
Therefore, the increment of the sum $\sum_{i=1}^{R} |y_i - x_i|$ is at most

$$L(x_1, x_2, \ldots, x_R) - GE(x_1, x_2, \ldots, x_R) \leqslant 0.$$

As above repeat this process until $y_1 = Y$.
We are construct the decision sequence with $y_1 = y_2 = \ldots = y_R = Y$ and $y_M \leqslant Y$. Since the sequence is nondecreasing hence $y_1 = y_2 = \ldots = y_M = Y$.

Lemma 8. Any array $(x_1, x_2, \ldots, x_N)$ can be split in the way described in the Lemma 7 conditions.
Proof.
Similarly to Lemma 6.
Designate $DE(P) = LE(x_1, x_2, \ldots, x_P) - G(x_1, x_2, \ldots, x_P)$, $1 \leqslant P \leqslant N$.
Let $DE(P)$ achieves to its maximal value with $P = P_0$ ( $DE(P_0) \geqslant DE(P)$, $1 \leqslant P \leqslant N$).
Note $DE(N) \geqslant 0$ by the median definition, therefore $DE(P_0) \geqslant 0$.
Then we can split the array $(x_1, x_2, \ldots, x_N)$ by desired manner with $M = P_0$.

Altogether, formulate the solution algorithm.
Algorithm 2.

1. Split (if it's possible) the array $(x_1, x_2, \ldots, x_N)$ so that the conditions of the Lemma 5 should be fulfilled. The method of the partition search is described in the Lemma 6.
2. In case this partition exists then apply process of the decision sequence construction recursively to the both obtained parts of the sequence.

3. In case of this partition is impossible, then split the array ($x_1$, $x_2$, …, $x_N$) so that the conditions of the Lemma 7 should be fulfilled (by the method described in the Lemma 8). Set $y_1 = y_2 = … = y_M =$ *median* ($x_1$, $x_2$, …, $x_N$) and apply recursively the whole process of the decision sequence construction to the right partition part.

Complexity analysis.
In the worst case, the algorithm 2 complexity is O(N·T(N)) where T(N) is the complexity of the median search. The median can be obtained with O(N) time expenditure (see, D.E.Knuth, *The art of computer programming, Vol. 3. Sorting and searching.* Second edition, 1998. Theorem L in chapter 5.3.3). The program `quick2.pas` is realizing this approach. However, the numerical experiment has showed that this program operates relatively slow in consequence of the realization complexity. The other program variant `quick1.pas` uses the "reduced" QuickSort to the median search and operates much faster than any other from the presented programs. "Reduced" QuickSort means we continue search only in one from two parts obtained because of array partition. This procedure complexity is O($N^2$) in the worst case and O(N) on the average.

On the average, the algorithm 2 complexity is O(log N ·T(N)). We can prove this fact in just the same way as in QuickSort analysis [D.E.Knuth, *The art of computer programming, Vol. 3. Sorting and searching.* Second edition, 1998. Algorithm Q in chapter 5.2.2].

In abstract

| Program | Complexity | |
|---|---|---|
| | In the worst case | On the average |
| **quick1** | **O($N^3$)** | **O(N·log N)** |
| **quick2** | **O($N^2$)** | **O(N·log N)** |

In spite of the fact that `quick1` has a bad theoretical evaluation of the complexity it operates much faster than `quick2` (and any other of the considered programs) – see Appendix. Quick in Quick is really quick! ☺

## 6. Third solution (dynamical programming).

Put the array ($x_1$, $x_2$, …, $x_N$) in the nondecreasing order and remove all repeating numbers from it. As a result we have obtained the array ($a_1$, $a_2$, …, $a_m$): $a_1 < a_2 < … < a_m$, moreover each of the array ($x_1$, $x_2$, …, $x_N$) elements is encountered in the array ($a_1$, $a_2$, …, $a_m$).

Correspondently to Lemma 3 we can construct the decision sequence ($y_1$, $y_2$, …, $y_N$) from elements of the array ($a_1$, $a_2$, …, $a_m$).

Let denote by $aim(k, j)$ the least possible value of the sum $\sum_{i=1}^{k} |y_i - x_i|$ provided that $y_1 \le y_2 \le \ldots \le y_k = a_j$.

Algorithm 3.
Direct motion – array *aim* filling.

$$aim(1, j) = |x_1 - a_j|, \text{ j=1, 2, …, m}$$

$$aim(k,j) = |x_k - a_j| + \min_{1 \le p \le j} aim(k-1,p)$$, j=1, 2, …, m; k=2, 3, …, N

It is clear that $\min \sum_{i=1}^{n} |y_i - x_i| = \min_{1 \le j \le m} aim(N,j)$

<u>Reverse motion</u> – the sequence ($y_1$, $y_2$, … $y_N$) construction.

Let $\min_{1 \le j \le m} aim(N,j) = aim(N,j_0)$. Then $y_N = a_{j_0}$.

Find $y_{N-1}$ from relation $y_{N-1} = a_{j_1}$, with $j_1 \le j_0$ and $aim(N-1,j_1) + |y_N - x_N| = aim(N,j_0)$.

Then find sequentially $y_{N-2}$, $y_{N-3}$, … $y_2$, $y_1$ in the similar way.

The complexity of the presented algorithm 3 is O($N^2$) with memory requirement O($N^2$). There is its realization in the program `dynamic1.pas`.

We can manage without 2-dimensional array `aim` for decision sequence constructing (on the reverse motion). In that case, the reverse motion requires to recalculate one-dimensional array `aim(m,*)` every time. Such model needs only O(N) memory and its complexity is O($N^3$). It has realized in the program `dynamic2.pas`.

## Appendix. Comparison of the presented programs speed. A plan for generating the test data.

| | | Time (sec.) | | | | |
|---|---|---|---|---|---|---|
| *Test description* | **Test size (N)** | **dynamic2** | **dynamic1** | **induct** | **quick1** | **quick2** |
| Increasing sequence $t_{i+1}$-$t_i$ > 1 | 1000 | 63 | <0.1 | <0.1 | <0.1 | <0.1 |
| | 2000 | 173 | 0.1 | <0.1 | <0.1 | <0.1 |
| | 50000 | – | – | 0.1 | 0.15 | 6.1 |
| | 100000 | – | – | 0.3 | 0.25 | 8.7 |
| | 500000 | – | – | 1.3 | 1.5 | 14.3 |
| | 1000000 | – | – | 2.7 | 3.3 | 32.0 |
| Increasing sequence $t_{i+1}$-$t_i$ = 1 | 1000 | 50 | <0.1 | <0.1 | <0.1 | <0.1 |
| | 2000 | 100 | <0.1 | <0.1 | <0.1 | <0.1 |
| | 500000 | – | – | 1.2 | 1.1 | 2.6 |
| | 1000000 | – | – | 2.4 | 2.3 | 5.1 |
| Decreasing sequence | 1000 | 58 | <0.1 | 0.2 | <0.1 | <0.1 |
| | 2000 | 157 | 0.2 | 0.35 | <0.1 | <0.1 |
| | 10000 | – | – | 2.1 | <0.1 | 0.15 |
| | 30000 | – | – | 9.9 | <0.1 | 0.3 |
| | 50000 | – | – | 22.3 | 0.12 | 0.35 |
| | 100000 | – | – | 116.0 | 0.2 | 1.0 |
| | 500000 | – | – | – | 1.2 | 4.9 |
| | 1000000 | – | – | – | 2.5 | 9.9 |
| Constant sequence (equivalent to decreasing sequence) | 1000 | 58 | <0.1 | 0.2 | <0.1 | <0.1 |
| | 2000 | 157 | <0.1 | 0.35 | <0.1 | <0.1 |
| | 10000 | – | – | 2.1 | <0.1 | 0.15 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 30000 | – | – | 9.8 | <0.1 | 0.3 |
| | 50000 | – | – | 22.3 | 0.12 | 0.35 |
| | 100000 | – | – | 116.0 | 0.2 | 1.0 |
| | 500000 | – | – | – | 1.2 | 4.9 |
| | 1000000 | – | – | – | 2.3 | 9.8 |
| 1,1,2,2,3,3,4,4,5,5, … | 2000 | 136 | 0.1 | 0.35 | <0.1 | <0.1 |
| | 100000 | – | – | 116.0 | 0.2 | 1.0 |
| | 1000000 | – | – | – | 2.1 | 9.5 |
| 1,1,1,2,2,2,3,3,3,4,4,4,5,5,5, … | 2000 | 139 | 0.1 | 0.35 | <0.1 | <0.1 |
| | 100000 | – | – | 116.0 | 0.2 | 0.95 |
| | 1000000 | – | – | – | 2.1 | 9.6 |
| 1,2,1,2,1,2,… | 2000 | 131 | <0.1 | 0.35 | <0.1 | 0.1 |
| | 100000 | – | – | 116.0 | 0.15 | 0.95 |
| | 1000000 | – | – | – | 1.6 | 9.2 |
| 1,1,3,3,5,5,7,7,9,9, … | 2000 | 106 | <0.1 | <0.1 | <0.1 | <0.1 |
| | 100000 | – | – | 7.6 | 0.2 | 0.7 |
| | 1000000 | – | – | 17.2 | 2.1 | 7.2 |
| 1,1,1,4,4,4,7,7,7,10,10,10, … | 2000 | 102 | <0.1 | 0.2 | <0.1 | <0.1 |
| | 100000 | – | – | 10.2 | 0.2 | 0.5 |
| | 1000000 | – | – | 102 | 2.1 | 5.3 |
| Random sequence, $0 \leqslant t_i \leqslant 100$ | 1000 | 56 | <0.1 | 0.2 | <0.1 | <0.1 |
| Random sequence, $0 \leqslant t_i \leqslant 100$ | 2000 | 139 | 0.1 | 0.35 | <0.1 | <0.1 |
| Random sequence, $0 \leqslant t_i \leqslant 10^6$ | 2000 | 158 | 0.15 | 0.3 | <0.1 | 0.12 |
| Random sequence, $0 \leqslant t_i \leqslant 100$ | 10000 | – | – | 2.1 | <0.1 | 0.15 |
| Random sequence, $0 \leqslant t_i \leqslant 10^6$ | 10000 | – | – | 1.4–1.6 | <0.1 | 0.3–0.4 |
| Random sequence, $0 \leqslant t_i \leqslant 100$ | 30000 | – | – | 9.8 | <0.1 | 0.3–0.35 |
| Random sequence, $0 \leqslant t_i \leqslant 10^6$ | 30000 | – | – | 4.8–5.3 | 0.1 | 0.7–0.8 |
| Random sequence, $0 \leqslant t_i \leqslant 500$ | 50000 | – | – | 22.3 | 0.12 | 0.45–0.5 |
| Random sequence, $0 \leqslant t_i \leqslant 10^5$ | 50000 | – | – | 19.3 | 0.12 | 0.5–0.55 |
| Random sequence, $0 \leqslant t_i \leqslant 10^8$ | 50000 | – | – | 7.9–8.0 | 0.15 | 1.4–1.7 |
| Random sequence, $0 \leqslant t_i \leqslant 10^5$ | 100000 | – | – | 135 | 0.2 | 1.0 |
| Random sequence, $0 \leqslant t_i \leqslant 10^9$ | 100000 | – | – | 15–16 | 0.3 | 2.9–3.2 |
| Random sequence, $0 \leqslant t_i \leqslant 10^6$ | 300000 | – | – | 740–790 | 0.6–0.8 | 3.5–7 |
| Random sequence, $0 \leqslant t_i \leqslant 10^9$ | 300000 | – | – | 55–60 | 0.9–1.0 | 8.5–9.5 |
| Random sequence, $0 \leqslant t_i \leqslant 10^6$ | 500000 | – | – | – | 1.25–1.3 | 4.8–4.9 |
| Random sequence, $0 \leqslant t_i \leqslant 5 \cdot 10^7$ | 500000 | – | – | 160 | 1.6 | 12.5–13 |
| Random sequence, $0 \leqslant t_i \leqslant 2 \cdot 10^9$ | 500000 | – | – | 120–140 | 1.7–1.8 | 13–14 |
| Random sequence, $0 \leqslant t_i \leqslant 10^6$ | 1000000 | – | – | – | 2.2–2.3 | 9.7–10.0 |
| Random sequence, $0 \leqslant t_i \leqslant 5 \cdot 10^8$ | 1000000 | – | – | 240–280 | 3.2–3.4 | 26–28 |
| Random sequence, $0 \leqslant t_i \leqslant 2 \cdot 10^9$ | 1000000 | – | – | 240–280 | 3.3–3.6 | 26–28 |