Anti-Quicksort

Bookmark this page

Anti-Quicksort

2.0 points possible (graded)

Input file:	antiqs.in
Output file:	antiqs.out
Time limit:	2 seconds
Memory limit:	256 megabytes

For sorting integer sequences, the Quicksort algorithm is widely used. Below we give a program which sorts an array a using this algorithm.

```
var a : array [1..N] of integer;

procedure QSort(left, right : integer);
var i, j, key, buf : integer;
begin
  key := a[(left + right) div 2];
  i := left;
  j := right;
  repeat
    while a[i] < key do
      inc(i);
    while key < a[j] do
      dec(j);
    if i <= j then begin
      buf := a[i];
      a[i] := a[j];
      a[j] := buf;
      inc(i);
      dec(j);
    end;
  until i > j;
  if left < j then QSort(left, j);
  if i < right then QSort(i, right);
end;
```

```
begin
  ...
  QSort(1, N);
end.
```

Although Quicksort is very fast on average, there exist integer sequences, which take quite a long time to be sorted using this algorithm. We will measure the running time of the algorithm by the number of comparisons where array elements participate (that is, the total number of comparisons in the first and the second while loop).

Your task is to generate a test, which forces the given Quicksort implementation to perform the maximum number of these comparisons.

Input

The first line of the input file contains a single integer number n ($1 \leq n \leq 10^6$).

Output

Output a permutation of numbers from 1 to n, which forces forces the given Quicksort implementation to perform the maximum number of these comparisons (among all permutations of numbers from 1 to n). If there are several such permutations, print any of them.

Example

antiqs.in	antiqs.out
3	1 3 2
Download	Download

Choose Files No file chosen

Submit You have used 0 of 200 attempts

Discussion

Topic: 06: 3rd Week Problems / Anti-Quicksort

Show Discussion