MMZ Sort

Proposta de um novo algoritmo de ordenação

0	1	2	3	4	5	6	7
68	19	33	7	22	13	46	88

Cria-se 2 arrays, um com os valores pares e outro com os valores ímpares.

0	1	2	3
68	22	46	88

0	1	2	3
19	33	7	13

Ordena-se ambos usando Insertion Sort em cada.

0	1	2	3
22	46	68	88

0	1	2	3
7	13	19	33

Para unificar o array par com o array ímpar em um array ordenado, será feito o seguinte processo:

Comparar o valor da primeira posição do array ímpar com o da primeira posição do array par e adicionar o menor na primeira posição do novo array de mesmo tamanho do original.

0	1	2	3
22	46	68	88

0	1	2	3
7	13	19	33

0	1	2	3	4	5	6	7
7							

Como o 7 era o menor, sendo ele a primeira posição do array ímpar, a primeira posição do array par vai ser comparada com a segunda posição do array ímpar, ambas disputando a segunda posição do novo array.

0	1	2	3
22	46	68	88

0	1	2	3
7	13	19	33

0	1	2	3	4	5	6	7

7	13						

Como o 13 era menor que o 22 também, o 22 agora vai ser comparado com o 29 para disputar pela 3° posição do array novo.

0	1	2	3
22	46	68	88

0	1	2	3
7	13	19	33

0	1	2	3	4	5	6	7
7	13	19					

A primeira posição do array par ainda não encontrou um valor maior que ela no array ímpar, então ela vai ser comparada com o 33, disputando a 4° posição do array novo.

0	1	2	3
22	46	68	88

0	1	2	3
7	13	19	33

0	1	2	3	4	5	6	7
7	13	19	22				

Como o 22 era menor que o 33, o 33 agora vai ser comparado com o valor da segunda posição do array par, disputando pela 5° posição do array novo.

0	1	2	3
22	46	68	88

0	1	2	3
7	13	19	33

0	1	2	3	4	5	6	7
7	13	19	22	33			

Como terminou o array ímpar, basta adicionar o resto do array par no novo array.

0	1	2	3
22	46	68	88

0	1	2	3
7	13	19	33

0	1	2	3	4	5	6	7
7	13	19	22	33	46	68	88

Algoritmo da solução criada acima:

```
811  void MMZSort(int arrayA[], int n) {
812    int I[n];
813    int P[n];
814    int contI = 0;
815    int contP = 0;
816
817    for(int i = 0; i < n; i++) {
818      if(arrayA[i]%2 == 0) {
819        P[contP] = arrayA[i];
820        contP++;
821      } else {
822        I[contI] = arrayA[i];
823        contI++;
824      }
825    }
826
827    insertionSort(P, contP);
828    insertionSort(I, contI);
829
830    int posicaoI = 0;
831    int posicaoP = 0;
832
833    for(int i = 0; i < n; i++) {
834      if(P[posicaoP] < I[posicaoI]) {
835        arrayA[i] = P[posicaoP];
836        posicaoP++;
837      } else {
838        arrayA[i] = I[posicaoI];
839        posicaoI++;
840      }
841    }
842  }
```

Comprovante do funcionamento:

```
Ordenando o Array - MMZ Sort:

 55 33 75 86 91 15 6 1 8 85
 1 6 8 15 33 55 75 85 86 91
```

Cálculo da complexidade no pior caso: Array par e ímpar decrescentes

Linha	812 à 815	817	818 à 820	827	828	830-831	833	834 à 836
T(n)	4	1+n+(n-1)	3	$contP^2$	$contI^2$	2	1+n+(n-1)	3

Somando tudo: $T(n) = 14 + n + (n-1) + n + (n-1) + contP^2 + contI^2$

$T(n) = 14 + 2n + 2(n-1) + contP^2 + contI^2$

$T(n) = 14 + 2n + 2n - 2 + contP^2 + contI^2$

$T(n) = 12 + 4n + contP^2 + contI^2 = O(n + contP^2 + contI^2)$