Zadanie 8.
Zaprojektuj algorytm, który sortuje n liczb całkowitych z przedziału od 1 do n^2 w czasie $O(n)$

Algorytmem, który proponuję użyć pochodzi z książki Wstęp do Algorytmów autorstwa Cormena. Jest to mianowicie Radix-Sort, poniżej umieszczony został pseudokod tego algorytmu.

W następującej procedurze przyjmujemy, że każdy element w n-elementowej tablicy A ma d cyfr, gdzie cyfra na pozycji 1 jest najmniej znacząca, a cyfra na pozycji d jest najbardziej znacząca.

```
RADIX-SORT(A, d)
for i ← 1 to d
    do posortuj stabilnie tablicę A według cyfry i
```

Do sortowania opisanego w tym algorytmie, można posłużyć się Couning-Sort, którego złożoność obliczeniowa wynosi $\Theta(n+k)$, gdzie n to ilość kluczy do posortowania, a k to zakres kluczy, z którego sortujemy.
Jednak nie możemy zastosować tego algorytmu bezpośrednio na kluczach z zakresu od 1 do n^2 i zostać przy złożoności $O(n)$ co wynika właśnie z zakresu $k = n^2$.
Pojawia się pytanie jak temu zaradzić, otóż możemy posłużyć się Radix-Sortem, ale korzystając z systemu pozycyjnego liczb o podstawie n. W zakresie 1 do n^2 mamy $n^2 - 1$ różnych liczb, możemy to "przeskalować" na przedział od 0 do $n^2 - 1$.
Dowolną liczbę M z tego zakresu jesteśmy w stanie zakodować jako parę liczb (k_1, k_2) w systemie pozycyjnym o podstawie n. To znaczy każda z liczb $k_1, k_2 < n$ oraz $M = (n * k_1 + k_2)$
Używamy 2 razy Counting-Sort na tych "cyfrach" - uwaga - "cyfra" oznacza tu liczbę z zakresu 0 do $n - 1$. A Counting Sort dla kluczy z tego zakresu działa w czasie $2 * \Theta(n)$. A to implikuje, że Radix-Sort działa w czasie $O(n)$.
Ponadto każdą z liczb jesteśmy w stanie zapisać w systemie o podstawie n w czasie stałym (możemy założyć, że pojedyncza operacja dzielenia i odejmowanie zajmują stały czas