CSI 5165 Combinatorial Algorithms
Assignment 1
Wei Li 0300113733 report date: 2/9/2021

Question 1

Give successor, ranking and unranking algorithms to Generalized Lexicographical n-tuples for mixed basis.
Let us denote the basis of n-tuples as $M = (m_0, m1, \ldots, m_{n-1})$ and $M[i]$ is the basis m_i. $T = (t_1, t_2, \ldots, t_{n-1})$ is a n-tuple generated from the lexicographical algorithm. r denotes the rank.

Ranking

Algorithm 1: tupleRanking(n, M, T)

Result: r
$r \leftarrow 0$;
$u \leftarrow 1$;
for $i \leftarrow$ *n-1 to 1* **do**
  $r \leftarrow r + t_i * u$;
  $u \leftarrow u * M[i]$;
end
return r

Unranking

Algorithm 2: tupleUnranking(n, M, r)

Result: T
$T \leftarrow \emptyset$;
$u \leftarrow \prod_{i=1}^{n-1}$ M[i];
for $i \leftarrow$ *0 to n-2* **do**
  $t_i \leftarrow \lfloor r/u \rfloor$;
  $T \leftarrow T \cup t_i$;
  $r \leftarrow r - t_i * u$;
  $u \leftarrow u/M[i+1]$;
end
$t_{n-1} \leftarrow r$;
$T \leftarrow T \cup t_{n-1}$;
return T

Successor

Algorithm 3: tupleSuccessor(n, M, T)

```
Result: T
for i ← n-1 to 0 do
    if t_i + 1 ≤ M[i] − 1 then
        t_i ← t_i + 1;
        return T;
    else
        t_i ← 0;
    end
end
```

Question 2

Correctness of Successor algorithm for Graycodes G^n: Prove Theorem 2.2 of the textbook which states that Algorithm GrayCodeSucessor(n,T) (slide 15) correctly computes successor for the binary reflected Gray code.
G^n will denote the binary reflected Gray code (BRGC) for the 2^n binary n-tuples

$$\mathrm{G}^n = \left[\mathrm{G}_0^n, \mathrm{G}_1^n, \ldots, \mathrm{G}_{2^n-1}^n\right]$$

By recursive definition, G^1 is defined to be

$$G^1 = [0, 1]$$

Given G^{n-1}, the Gray code G^n is defined to be

$$G^n = \left[0G_0^{n-1}, \ldots, 0G_{2^{n-1}-1}^{n-1}, 1G_{2^{n-1}-1}^{n-1}, \ldots, 1G_0^{n-1}\right] = \left[0G^n, 1(G^n)^R\right]$$

we use $(G^n)^R$ to denote the reverse of (G^n).
Property without proof:
1. G^n always has even number of elements (since the recursive definition double the set every time).
2. Consecutive elements in G^n ($A = (a_{n-1}, an-2, \ldots, a_1, a_0)$) only differ in 1 element. (the definition of gray code)

Prove 1: To get a successor, If $w(A)$ is even, then the last bit of A (namely a_0) is flipped.
Proof: we proof two properties,
1) the parity of weight in G^n is a repeating sequence of even and odd, starting with even weight.
2) the elements in G^n can be group to a group of 2 starting with even weight (even weight then odd weight), and consecutive elements in a group only differ from the last bit.

1) by induction,
basis: in the G^0, the weight is repeating sequence of even and odd, starting with even (0) weight.
induction hypothesis: in G^n the weight is repeating sequence of even and odd, starting with even weight.
The G^{n+1} is

$$G^{n+1} = \left[0G^n, 1(G^n)^R\right]$$

In the first half, adding 0 to G^n does not change its parity. In the second half, the parity sequence of weight in the reversed G^n is reversed (odd, even,...), but adding 1 to the begining reverse the sequence again, so it is still repeating sequence of even and odd starting with even weight.

2)also by induction,
basis: obviously G^0 and G^1 satisfy the property that consecutive elements in a 2-group start with even weight only differ from the last bit.
induction hypothesis: in G^n consecutive elements in a 2-group start with even weight only differ from the last bit.
In G^{n+1}, in the first half adding 0 to the head does not change the property. In the second half, even thought G^n is reversed, elements in each of the 2-group remain the same, and by property 1), the even-odd order is also the same. Thus in G^{n+1} consecutive elements in a 2-group start with a even weight still differ from the last bit.

By property 1) and 2), we can get the successor of a element in even weight position by flip the last bit.

Prove 2: To get a successor, If $w(A)$ is odd, then we find the first "1" from the right, and flip the next bit (to the left).
Proof: we proof 2 properties,
1) the first element of G^n for any $n \geq 1$ is all '0'; the last element of G^n for any $n \geq 1$ is a '1' following n-1 number of '0's. We denote the characteristic as a '[10] sequence'.
2) element in an group of two starting with an odd weight element always differ from the successor in the digit before the first '1' .

1)by induction,
basis: G^1 and G^2 satisfy the property.
induction hypothesis: assume G^n satisfies that the first element is all '0' and the last element is '[10] sequence'.
In G^{n+1}, the first element is adding 0 to the head of the first element of G^n, which is still all '0'. The last element is the adding '1' to the first element of G^n, which is a '[10] sequence'.

2)by induction,
basis: in G^2, group ('01', '11') satisfy the property. in G^3, group (001, '011), (010, 100), (111, 101) satisfy the property
*induction hypothesis:*in G^n element in an group of 2 starting with an odd weight element always differ from the successor the digit before the first '1' .
In G^{n+1},

- case1.in the first half, $[0G^n]$ does not change the parity order, thus does not change the property of G^n.

- case2.in the middle, the first element of the 2-group (starting from odd position) is the last element of the first half, which is a 0 adding to a [10] sequence ([010...]). The second element (even position) is the first element in the second half, which is a 1 adding to a [10] sequence ([110...]). The difference of the two is the first digit right before the first '1'.

- case3. the second half is $[1(G^n)^R]$, since the parity position has not changed, the property remain the same, like case1.

By property 2), we can get the successor of a odd weight position by flipping the next bit (to the left) before the first '1'.

Question 3

We define the tuple C^n of subsets according to non-increasing cardinality order by concatenating the tuples of subsets given by $A^{n,k}, k = n, n-1, \ldots, 1$ as follows:

$$C^n = A^{n,n} \left\| A^{n,n-1} \right\| \cdots \left\| A^{n,1} \right\| A^{n,0}$$

Give successor, ranking and unranking algorithms (pseudocode is fine).

It is obvious that C^n contains $\sum_{k=n}^{0} \binom{n}{k}$ entries. Each $\binom{n}{k}$ consists of a block a k-subset corresponds to $A^{n,k}$.

Assume we have the successor, ranking and unranking algorithm of revolving door ordering k-subset as *kSubsetRevDoorSuccessor(T, k, n)*, *kSubsetRevDoorRank(T, k)* and *kSubsetRevDoorUnrank(r, k, n)* respectively, assume we can use those algorithm directly.

We refer the length of tuple T by *T.length*.

Ranking

Algorithm 4: Ranking(T, n)

Result: r
if *T.length is n* **then**
  $r \leftarrow 0$;
else
  $r \leftarrow \sum_{k=n}^{T.length+1} \binom{n}{k}$;
  $k \leftarrow T.length$;
  $r \leftarrow r + kSubsetRevDoorRank(T, k)$;
end
return r;

Unranking

Algorithm 5: Unranking(r, n)

Result: T
for $i \leftarrow n$ *to 0* **do**
  if $0 \leq r < \binom{n}{i}$ **then**
    $T \leftarrow kSubsetRevDoorUnrank(r, i, n)$;
    return T;
  else
    $r \leftarrow r - \binom{n}{i}$
  end
end

The *kSubsetRevDoorUnrank(r, k, n)* should be able to account for the case of empty set (kSubsetRevDoorUnrank(0, 0, n)) by initiate T with ϕ.

Successor

Algorithm 6: Successor(T, n)

Result: T
$k \leftarrow T.length$;
if *T is k-1 natural numbers sequence +* $\{n\}$ **then**
  $T \leftarrow \{1, 2, 3, ..., k-1\}$;
else
  $T \leftarrow kSubsetRevDoorSuccessor(T, k, n)$;
end
return T;

Question 4

A signed permutation is a permutation with optional signs attached to the elements; therefore, there are $2^n n!$ such signed permutations of n elements.

- Give an algorithm that generates all signed permutations of $1, 2, ..., n$ where each step either interchanges two adjacent elements or negates the first element.
- Demonstrate that your algorithm works by implementing it, and showing the results for $n = 1, 2, 3, 4$. Please do a compact printout with $2^{n-1}(n1)!$ lines with $2n$ permutations per line.

solution:
In the algorithm, we assign each integer in the permutation with a direction, and we define two action: negation and swap. We also define a *mobile integer*, which is: *the largest integer in the permutation who is greater than the neighborhood its direction points to; when no such integer can be found, mobile integer is '1'.* In each iteration, there is only one mobile integer and it is the only one that takes action like negation or swap.

In the algorithm we set two vectors $\overline{neg}$ and $\overline{dir}$ of size n for permutation of size n. The $\overline{neg}$ stores 0/1 value, when 1 is stores, the next time a integer becomes mobile, its action is negation, and the entry resume to 0. $\overline{dir}$ stores the direction (left or right) an integer points to.

When an integer becomes mobile and the action is not negation, the integer swap with the next integer it points to. Each entry in the two vectors correspond to the integer in the permutation, for example: dir[0] store the direction of number '1' or '-1', neg[n-1] store negate property of number 'n' or '-n', notes that one integer and its negation share the same entry, so to access the entry we only need to take an absolute number.

The algorithm is described as follow:

1. get the mobile integer of current permutation
2. if mobile integer is '1', negate '1' (become -1), and flip the direction of all integers. go back to 1.
3. if mobile integer is not '1', filp the direction of all integers who is greater than the mobile integer, and go to 4 or 5.
4. if the mobile integer's $\overline{neg}$ condition is 1, then negate the mobile integer and set its $\overline{neg}$ condition is 0, go back to 1.
5. else, swap the mobile integer with the neighborhood its direction points to. If the swap led the mobile integer to the head of the permutation, change its direction immediately and set its $\overline{neg}$ condition to '1'. go back to 1.

0.1 result

Figure 1: n = 1

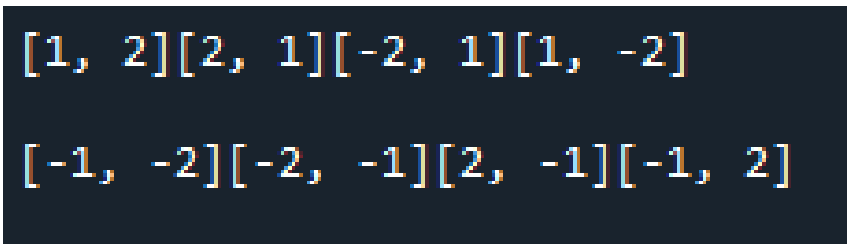

Figure 2: n = 2

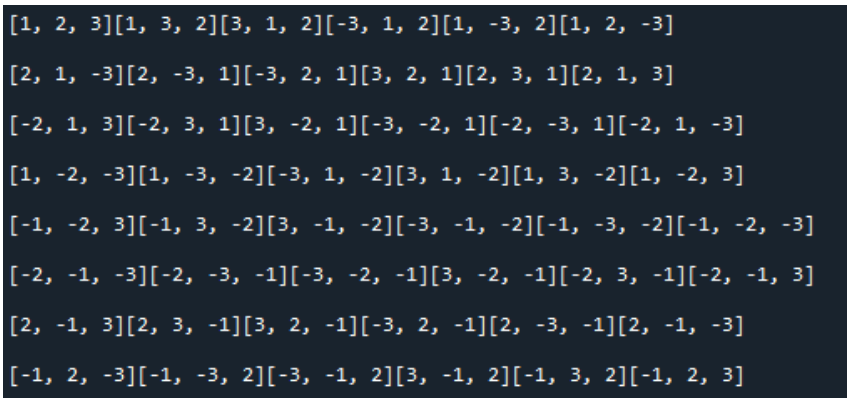

Figure 3: n = 3

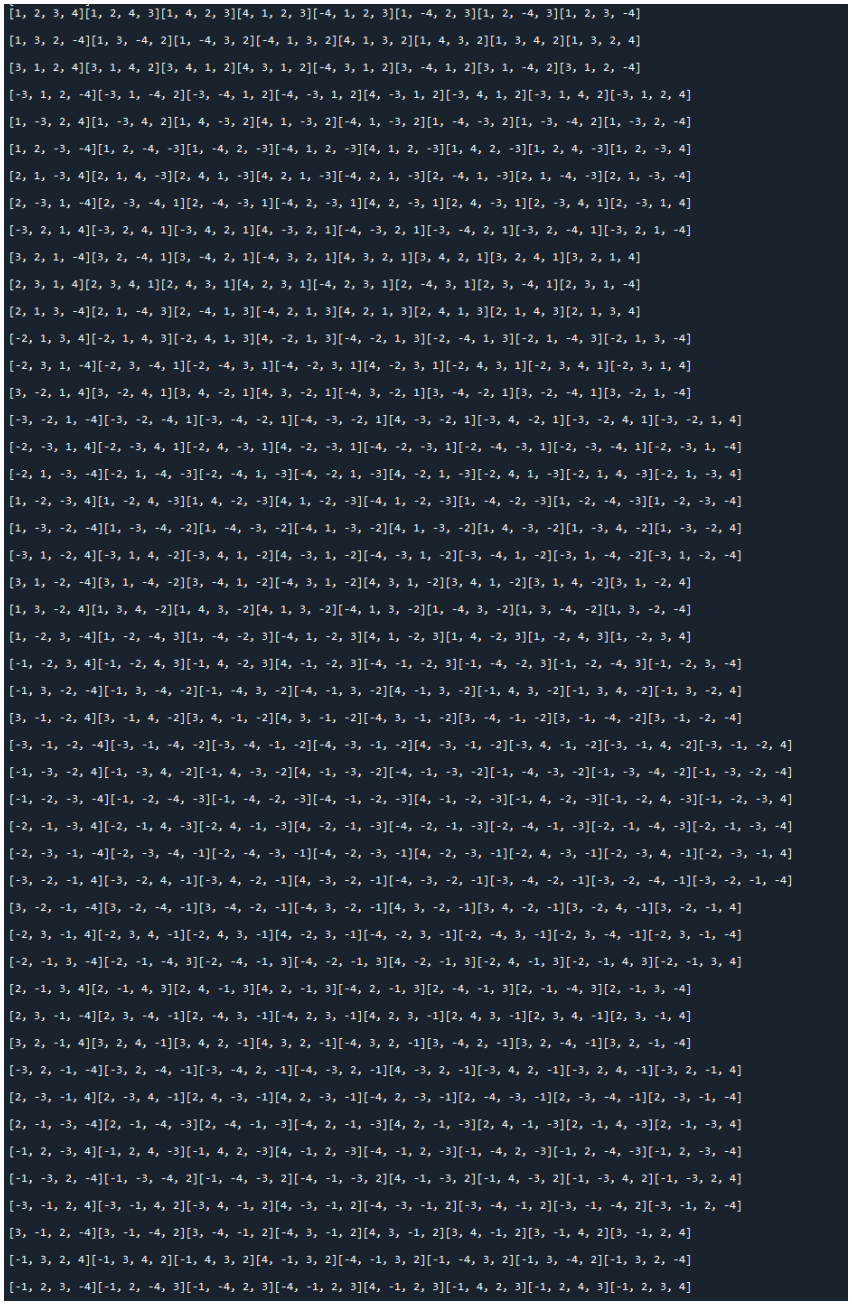
[1, 2, 3, 4][1, 2, 4, 3][1, 4, 2, 3][4, 1, 2, 3][-4, 1, 2, 3][1, -4, 2, 3][1, 2, -4, 3][1, 2, 3, -4]
[1, 3, 2, -4][1, 3, -4, 2][1, -4, 3, 2][-4, 1, 3, 2][4, 1, 3, 2][1, 4, 3, 2][1, 3, 4, 2][1, 3, 2, 4]
[3, 1, 2, 4][3, 1, 4, 2][3, 4, 1, 2][4, 3, 1, 2][-4, 3, 1, 2][3, -4, 1, 2][3, 1, -4, 2][3, 1, 2, -4]
[-3, 1, 2, -4][-3, 1, -4, 2][-3, -4, 1, 2][-4, -3, 1, 2][4, -3, 1, 2][-3, 4, 1, 2][-3, 1, 4, 2][-3, 1, 2, 4]
[1, -3, 2, 4][1, -3, 4, 2][1, 4, -3, 2][4, 1, -3, 2][-4, 1, -3, 2][1, -4, -3, 2][1, -3, -4, 2][1, -3, 2, -4]
[1, 2, -3, -4][1, 2, -4, -3][1, -4, 2, -3][-4, 1, 2, -3][4, 1, 2, -3][1, 4, 2, -3][1, 2, 4, -3][1, 2, -3, 4]
[2, 1, -3, 4][2, 1, 4, -3][2, 4, 1, -3][4, 2, 1, -3][-4, 2, 1, -3][2, -4, 1, -3][2, 1, -4, -3][2, 1, -3, -4]
[2, -3, 1, -4][2, -3, -4, 1][2, -4, -3, 1][-4, 2, -3, 1][4, 2, -3, 1][2, 4, -3, 1][2, -3, 4, 1][2, -3, 1, 4]
[-3, 2, 1, 4][-3, 2, 4, 1][-3, 4, 2, 1][4, -3, 2, 1][-4, -3, 2, 1][-3, -4, 2, 1][-3, 2, -4, 1][-3, 2, 1, -4]
[3, 2, 1, -4][3, 2, -4, 1][3, -4, 2, 1][-4, 3, 2, 1][4, 3, 2, 1][3, 4, 2, 1][3, 2, 4, 1][3, 2, 1, 4]
[2, 3, 1, 4][2, 3, 4, 1][2, 4, 3, 1][4, 2, 3, 1][-4, 2, 3, 1][2, -4, 3, 1][2, 3, -4, 1][2, 3, 1, -4]
[2, 1, 3, -4][2, 1, -4, 3][2, -4, 1, 3][-4, 2, 1, 3][4, 2, 1, 3][2, 4, 1, 3][2, 1, 4, 3][2, 1, 3, 4]
[-2, 1, 3, 4][-2, 1, 4, 3][-2, 4, 1, 3][4, -2, 1, 3][-4, -2, 1, 3][-2, -4, 1, 3][-2, 1, -4, 3][-2, 1, 3, -4]
[-2, 3, 1, -4][-2, 3, -4, 1][-2, -4, 3, 1][-4, -2, 3, 1][4, -2, 3, 1][-2, 4, 3, 1][-2, 3, 4, 1][-2, 3, 1, 4]
[3, -2, 1, 4][3, -2, 4, 1][3, 4, -2, 1][4, 3, -2, 1][-4, 3, -2, 1][3, -4, -2, 1][3, -2, -4, 1][3, -2, 1, -4]
[-3, -2, 1, -4][-3, -2, -4, 1][-3, -4, -2, 1][-4, -3, -2, 1][4, -3, -2, 1][-3, 4, -2, 1][-3, -2, 4, 1][-3, -2, 1, 4]
[-2, -3, 1, 4][-2, -3, 4, 1][-2, 4, -3, 1][4, -2, -3, 1][-4, -2, -3, 1][-2, -4, -3, 1][-2, -3, -4, 1][-2, -3, 1, -4]
[-2, 1, -3, -4][-2, 1, -4, -3][-2, -4, 1, -3][-4, -2, 1, -3][4, -2, 1, -3][-2, 4, 1, -3][-2, 1, 4, -3][-2, 1, -3, 4]
[1, -2, -3, 4][1, -2, 4, -3][1, 4, -2, -3][4, 1, -2, -3][-4, 1, -2, -3][1, -4, -2, -3][1, -2, -4, -3][1, -2, -3, -4]
[1, -3, -2, -4][1, -3, -4, -2][1, -4, -3, -2][-4, 1, -3, -2][4, 1, -3, -2][1, 4, -3, -2][1, -3, 4, -2][1, -3, -2, 4]
[-3, 1, -2, 4][-3, 1, 4, -2][-3, 4, 1, -2][4, -3, 1, -2][-4, -3, 1, -2][-3, -4, 1, -2][-3, 1, -4, -2][-3, 1, -2, -4]
[3, 1, -2, -4][3, 1, -4, -2][3, -4, 1, -2][-4, 3, 1, -2][4, 3, 1, -2][3, 4, 1, -2][3, 1, 4, -2][3, 1, -2, 4]
[1, 3, -2, 4][1, 3, 4, -2][1, 4, 3, -2][4, 1, 3, -2][-4, 1, 3, -2][1, -4, 3, -2][1, 3, -4, -2][1, 3, -2, -4]
[1, -2, 3, -4][1, -2, -4, 3][1, -4, -2, 3][-4, 1, -2, 3][4, 1, -2, 3][1, 4, -2, 3][1, -2, 4, 3][1, -2, 3, 4]
[-1, -2, 3, 4][-1, -2, 4, 3][-1, 4, -2, 3][4, -1, -2, 3][-4, -1, -2, 3][-1, -4, -2, 3][-1, -2, -4, 3][-1, -2, 3, -4]
[-1, 3, -2, -4][-1, 3, -4, -2][-1, -4, 3, -2][-4, -1, 3, -2][4, -1, 3, -2][-1, 4, 3, -2][-1, 3, 4, -2][-1, 3, -2, 4]
[3, -1, -2, 4][3, -1, 4, -2][3, 4, -1, -2][4, 3, -1, -2][-4, 3, -1, -2][3, -4, -1, -2][3, -1, -4, -2][3, -1, -2, -4]
[-3, -1, -2, -4][-3, -1, -4, -2][-3, -4, -1, -2][-4, -3, -1, -2][4, -3, -1, -2][-3, 4, -1, -2][-3, -1, 4, -2][-3, -1, -2, 4]
[-1, -3, -2, 4][-1, -3, 4, -2][-1, 4, -3, -2][4, -1, -3, -2][-4, -1, -3, -2][-1, -4, -3, -2][-1, -3, -4, -2][-1, -3, -2, -4]
[-1, -2, -3, -4][-1, -2, -4, -3][-1, -4, -2, -3][-4, -1, -2, -3][4, -1, -2, -3][-1, 4, -2, -3][-1, -2, 4, -3][-1, -2, -3, 4]
[-2, -1, -3, 4][-2, -1, 4, -3][-2, 4, -1, -3][4, -2, -1, -3][-4, -2, -1, -3][-2, -4, -1, -3][-2, -1, -4, -3][-2, -1, -3, -4]
[-2, -3, -1, -4][-2, -3, -4, -1][-2, -4, -3, -1][-4, -2, -3, -1][4, -2, -3, -1][-2, 4, -3, -1][-2, -3, 4, -1][-2, -3, -1, 4]
[-3, -2, -1, 4][-3, -2, 4, -1][-3, 4, -2, -1][4, -3, -2, -1][-4, -3, -2, -1][-3, -4, -2, -1][-3, -2, -4, -1][-3, -2, -1, -4]
[3, -2, -1, -4][3, -2, -4, -1][3, -4, -2, -1][-4, 3, -2, -1][4, 3, -2, -1][3, 4, -2, -1][3, -2, 4, -1][3, -2, -1, 4]
[-2, 3, -1, 4][-2, 3, 4, -1][-2, 4, 3, -1][4, -2, 3, -1][-4, -2, 3, -1][-2, -4, 3, -1][-2, 3, -4, -1][-2, 3, -1, -4]
[-2, -1, 3, -4][-2, -1, -4, 3][-2, -4, -1, 3][-4, -2, -1, 3][4, -2, -1, 3][-2, 4, -1, 3][-2, -1, 4, 3][-2, -1, 3, 4]
[2, -1, 3, 4][2, -1, 4, 3][2, 4, -1, 3][4, 2, -1, 3][-4, 2, -1, 3][2, -4, -1, 3][2, -1, -4, 3][2, -1, 3, -4]
[2, 3, -1, -4][2, 3, -4, -1][2, -4, 3, -1][-4, 2, 3, -1][4, 2, 3, -1][2, 4, 3, -1][2, 3, 4, -1][2, 3, -1, 4]
[3, 2, -1, 4][3, 2, 4, -1][3, 4, 2, -1][4, 3, 2, -1][-4, 3, 2, -1][3, -4, 2, -1][3, 2, -4, -1][3, 2, -1, -4]
[-3, 2, -1, -4][-3, 2, -4, -1][-3, -4, 2, -1][-4, -3, 2, -1][4, -3, 2, -1][-3, 4, 2, -1][-3, 2, 4, -1][-3, 2, -1, 4]
[2, -3, -1, 4][2, -3, 4, -1][2, 4, -3, -1][4, 2, -3, -1][-4, 2, -3, -1][2, -4, -3, -1][2, -3, -4, -1][2, -3, -1, -4]
[2, -1, -3, -4][2, -1, -4, -3][2, -4, -1, -3][-4, 2, -1, -3][4, 2, -1, -3][2, 4, -1, -3][2, -1, 4, -3][2, -1, -3, 4]
[-1, 2, -3, 4][-1, 2, 4, -3][-1, 4, 2, -3][4, -1, 2, -3][-4, -1, 2, -3][-1, -4, 2, -3][-1, 2, -4, -3][-1, 2, -3, -4]
[-1, -3, 2, -4][-1, -3, -4, 2][-1, -4, -3, 2][-4, -1, -3, 2][4, -1, -3, 2][-1, 4, -3, 2][-1, -3, 4, 2][-1, -3, 2, 4]
[-3, -1, 2, 4][-3, -1, 4, 2][-3, 4, -1, 2][4, -3, -1, 2][-4, -3, -1, 2][-3, -4, -1, 2][-3, -1, -4, 2][-3, -1, 2, -4]
[3, -1, 2, -4][3, -1, -4, 2][3, -4, -1, 2][-4, 3, -1, 2][4, 3, -1, 2][3, 4, -1, 2][3, -1, 4, 2][3, -1, 2, 4]
[-1, 3, 2, 4][-1, 3, 4, 2][-1, 4, 3, 2][4, -1, 3, 2][-4, -1, 3, 2][-1, -4, 3, 2][-1, 3, -4, 2][-1, 3, 2, -4]
[-1, 2, 3, -4][-1, 2, -4, 3][-1, -4, 2, 3][-4, -1, 2, 3][4, -1, 2, 3][-1, 4, 2, 3][-1, 2, 4, 3][-1, 2, 3, 4]

Figure 4: n = 4