irred of Sn 2

```
# Look in the help of SymmetricGroupAlgebra and there is a comment
that there are two types of multiplication
# in order to ensure that the multiplication agrees with the formulas
in the explanation
# you can specify the multiplication is done "right-to-left" or 'r2l'
(the default is "left-to-right" or 'l2r'
sage.combinat.permutation.Permutations.global_options(mult = 'r2l')
```

```
def YFLO(T1, T2):
    """
    Test if T1 is (strictly) less than T2 in Youngs first letter order

    EXAMPLES::

        sage: YFLO([[1,3,5],[2,4]],[[1,2,3],[4,5]])
        True
        sage: YFLO([[1,2],[3]],[[1,3],[2]])
        False
    """
    T1 = StandardTableau(T1)
    T2 = StandardTableau(T2)
    for i in range(T1.size()):
        c1 = T1.cells_containing(i+1)[0]
        c2 = T2.cells_containing(i+1)[0]
        if c1!=c2:
            return c1[0]>c2[0]
    return False

def my_bubble_sort( L, cmp ):
    """
    this sorts a list L using the cmp function

    EXAMPLES::

        sage: my_bubble_sort( range(10), lambda x,y: x>y )
        [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
        sage: my_bubble_sort( range(10), lambda x,y: x<y )
        [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
        sage: my_bubble_sort( StandardTableaux([3,1]).list(), YFLO )
        [[[1, 2, 3], [4]], [[1, 2, 4], [3]], [[1, 3, 4], [2]]]
    """
    for i in range(len(L)):
        for j in range(i+1,len(L)):
            if cmp(L[i],L[j]):
                t = L[i]
                L[i]=L[j]
                L[j] = t
    return L
```

```
def wedge_tableaux( T1, T2 ):
    """
    find the wedge of two tableaux T1 and T2

    EXAMPLES::

        sage: wedge_tableaux([[1,2],[3]],[[1,3],[2]])
        [(0, 0), (0, 0), (1, 1)]
        sage: wedge_tableaux([[1,3],[2]],[[2,3],[1]])
        [(0, 0), (1, 0), (0, 1)]
    """
    T1 = Tableau(T1)
    T2 = Tableau(T2)
    return [(T1.cells_containing(i+1)[0][0], T2.cells_containing(i+1)
[0][1]) for i in range(T1.size())]

def test_good( T1, T2 ):
    """
    return True if the number of coordinates in wedge_tableaux
    is equal to the size of the tableau, otherwise return False
    this is the definition of "good" on page 2 of the notes.

    EXAMPLES::

        sage: test_good([[1,2],[3]],[[1,3],[2]])
        False
        sage: test_good([[1,3],[2]],[[2,3],[1]])
        True
    """
    return len(Set(wedge_tableaux(T1,T2)))==Tableau(T1).size()
```

```
def Cee( T1, T2 ):
    """
    This is the function in defintion 1.1 of the notes
    EXAMPLES::

        sage: Cee([[1,2],[3]],[[1,3],[2]])
        0
        sage: Cee([[1,3],[2]],[[2,3],[1]])
        -1
    """
    if test_good(T1, T2):
        # find the sign of the permtuation beta
        wc = wedge_tableaux(T1,T2)
        entries=flatten(T2)
        count = 0 # the number of inversions of beta
        for i in range(len(wc)):
            for j in range(i+1, len(wc)):
                if wc[entries[i]-1][1]==wc[entries[j]-1][1] and
wc[entries[i]-1][0]>wc[entries[j]-1][0]:
                    count = count + 1
        return (-1)^count
    else:
        return 0

def action( sig, Tab ):
    """
    function which takes a permutation and acts on a tableau
    EXAMPLES::
        sage: all(action( sig*tau, [[1,2,3]]) == action( sig,
action(tau, [[1,2,3]])) for sig in Permutations(3) for tau in
Permutations(3))  # this uses the 'r2l' action!!!
        True
        sage: action( [3, 1, 2], [[1,2],[3]] )
        [[3, 1], [2]]
    """
    return [[sig[v-1] for v in row] for row in Tab]

def tab_in_YFLO( la ):
    """
    return a list of standard tableaux in YFLO

    EXAMPLES::

        sage: tab_in_YFLO([3,1])
        [[[1, 2, 3], [4]], [[1, 2, 4], [3]], [[1, 3, 4], [2]]]
    """
    return my_bubble_sort( StandardTableaux(la).list(), YFLO)

def Cmat( la, sig ):
    """
    la is a partition
    sig is a permtuation which acts on the tableaux
    create a matrix indexed by standard tableaux in YFLO
    the i,j entry is Cee(T_i, action(sig, T_j))
    Equation 1.4 in the notes

    EXAMPLES::
        sage: Cmat([3,2],[1,2,3,4,5])
        [ 1  0  0  0  0]
        [ 0  1  0  0  0]
        [ 0  0  1  0  0]
        [ 0  0  0  1  0]
        [-1  0  0  0  1]
        sage: Cmat([3,1],[2,1,3,4])
        [ 1  0  0]
        [ 0  1  0]
        [-1 -1 -1]
    """
    return matrix([[Cee(T1, action(sig, T2)) for T2 in tab_in_YFLO( la
)] for T1 in tab_in_YFLO( la )])

def Amat( la, sig ):
    """
    This is equation 1.5 of the notes
    This is what Young proved was the irreducible representation of
S_n
    indexed by the partition la
    EXAMPLES::
        sage: Amat([2,2], [1,2,3,4])
        [1 0]
        [0 1]
        sage: Amat([3,2], [2,1,3,4,5])
        [ 1  0  0  0  0]
        [ 0  1  0  0  0]
        [ 0  0  1  0  0]
        [-1 -1  0 -1  0]
        [ 1  0 -1  0 -1]

    Prove that [2,2] is a representation but this will only return
True if we use the 'r2l' multiplication::

        sage: all( Amat([2,2],g)*Amat([2,2],h) == Amat([2,2], g*h) for
g in Permutations(4) for h in Permutations(4))
        True
    """
    return Cmat(la, range(1, len(sig)+1))^(-1)*Cmat( la, sig )
```

```
def Pee( T ):
    """
    the element P(T) in the symmetric group algebra
    defined on page 4 equation 2.1

    EXAMPLES::
        sage: Pee([[1,2],[3,4]])
        [1, 2, 3, 4] + [1, 2, 4, 3] + [2, 1, 3, 4] + [2, 1, 4, 3]
    """
    RT = Tableau(T).row_stabilizer()
    A = SymmetricGroupAlgebra(QQ, Tableau(T).size())
    return sum(A(Permutation(sig)) for sig in RT)

def eNN( T ):
    """
    the element N(T) in the symmetric group algebra
    defined on page 4 equation 2.1

    EXAMPLES::

        sage: eNN([[1,2],[3,4]])
        [1, 2, 3, 4] - [1, 4, 3, 2] - [3, 2, 1, 4] + [3, 4, 1, 2]
    """
    CT = Tableau(T).column_stabilizer()
    A = SymmetricGroupAlgebra(QQ, Tableau(T).size())
    return sum(sig.sign()*A(Permutation(sig)) for sig in CT)

def sig_T1_T2( T1, T2 ):
    """
    This returns the permutation which transforms T1 into T2

    EXAMPLES::

        sage: sig_T1_T2([[1,3],[2,4]], [[2,1],[4,3]])
        [2, 4, 1, 3]
        sage: sig_T1_T2([[1,3],[2,4,5]], [[1,3],[2,4,5]])
        [1, 2, 3, 4, 5]
    """
    T1 = Tableau(T1)
    return Permutation([T2[cx][cy] for i in range(T1.size()) for
(cx,cy) in T1.cells_containing(i+1)])
```

```
sig_T1_T2([[1,3],[2,4]], [[2,1],[4,3]])
```

```
    [2, 4, 1, 3]
```

```
eNN( [[1,2],[3,4]] )
```

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

```
# on the bottom of page 4, there are 3 equations
# the following tests if equation 2.2 (a) (first part) is correct
T = [[1,2,3],[4,5]]
A = SymmetricGroupAlgebra(QQ,5)
all( Pee( action( sigma, T ) ) == A(sigma)*Pee(T)*A(sigma.inverse())
for sigma in Permutations(5))
```

```
    True
```

```
action( [2,1,3,4,5], T )
```

```
    [[2, 1, 3], [4, 5]]
```

```
sig_T1_T2( [[1,2],[3]], [[1,3],[2]])
```

```
    [1, 3, 2]
```

```
eNN( [[1,2],[3]])
```

```
    [1, 2, 3] - [3, 2, 1]
```

```
Pee([[1,2],[3]])
```

```
    [1, 2, 3] + [2, 1, 3]
```

```
T = StandardTableau([[1,2],[3]])
```

```
T.row_stabilizer()
```

```
    Permutation Group with generators [(), (1,2)]
```

```
# test if Amat is a representation for la = [2,2]
# it must satisfy the following identity
# NOTE: this didn't work in class because we had to specify that the
SymmetricGroupAlgebra
# and permutation multiplication used 'r2l' ("right-to-left"
multiplication)
# see the first line of this worksheet
all( Amat([2,2],g)*Amat([2,2],h)==Amat([2,2],g*h) for g in
Permutations(4) for h in Permutations(4))
```

```
    True
```

```
Amat([3,2], [2,1,3,4,5])*Amat([3,2],[1,2,4,3,5])
```

```
    [ 0  1  0  0  0]
    [ 1  0  0  0  0]
    [ 0  0  1  0  0]
    [-1 -1  0 -1  0]
    [ 0  1  0  1  1]
```

```
Amat([3,2], [2,1,4,3,5])
```

```
    [ 0  1  0  0  0]
    [ 1  0  0  0  0]
    [ 0  0  1  0  0]
    [-1 -1  0 -1  0]
    [ 0  1  0  1  1]
```

```
action([3,2,1],[[1,2],[3]])
```

```
    [[3, 2], [1]]
```

```
Cee([[1,4,5],[6,2,3],[7]],[[6,4,2,3,5],[1,7]])
```

```
    -1
```

```
test_good([[1,4,5],[6,2,3],[7]],[[1,4,2,6,7],[5,3]])
```

```
    False
```

```
Set(wedge_tableaux([[1,4,5],[6,2,3],[7]],[[1,4,2,6,7],[5,3]]))
```

```
    {(1, 2), (0, 1), (0, 0), (1, 3), (2, 4), (1, 1)}
```

```
test_good([[1,4,5],[6,2,3],[7]],[[6,4,2,3,5],[1,7]])
```

```
    True
```

```
Set(wedge_tableaux([[1,4,5],[6,2,3],[7]],[[6,4,2,3,5],[1,7]]))
```

```
    {(1, 2), (0, 1), (0, 0), (0, 4), (1, 3), (1, 0), (2, 1)}
```

```
my_bubble_sort( StandardTableaux([3,2]).list(), YFLO )
```

```
    [[[1, 2, 3], [4, 5]], [[1, 2, 4], [3, 5]], [[1, 2, 5], [3, 4]], [[1,
    3, 4], [2, 5]], [[1, 3, 5], [2, 4]]]
```

```
YFLO([[1,2],[3]],[[1,3],[2]])
```

```
    False
```

```
YFLO([[1,3,5],[2,4]],[[1,2,3],[4,5]])
```

```
    True
```