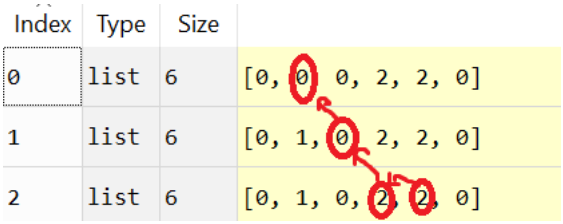
```
for i in range(len(em[0])):

    res[i][0]=s[i]+em[0][i]
pp=[]
f=[]
for i in range(1,len(em)):
    for j in range(len(tran)):

        for k in range(len(tran)):

            f.append(em[i][j]+tran[k][j]+res[k][i-1])


        res[j][i]=max(f)
        ind=f.index(max(f))
        pp.append(ind)
        back[j][i]=(ind)
#        print(j,k,'111')
#        for p in range(len(f)):
#
#            res[p][i]=f[p]
##        print(res,'res')
        f=[]

sd=[]
for i in range(2,len(pp)-1,2):
    sd.append(min(pp[i],pp[i+1]))


for i in range(len(em[0])):

    res[i][-1]=e[i]+res[i][-2]
path=-100000
for i in range(len(res)):
    if res[i][-1]>path:
        nnn=i
        path=res[i][-1]
final=[]
final.append(nnn)

for i in range(len(em)-1,0,-1):
    final.append(back[nnn][i])
    nnn=back[nnn][i]

final=final[::-1]
```

First of all we need to define two array for having results(res) so far and also one for back tracking(back). We need to add starting score to our res list and then start to find best tag for our sequence of word so far. By saying that it means we need to find all possible combination of coming from different tag to different tags of each word. Then each turn we just select maximum score and put it in our res list, we do that until the end of the sequence. After that we add ending score to the maximum score so far and return it. For finding the path, we need to remember what the tag with each maximum score was and then store

it to our back track list. Our back track list dimension is the same as res. Finally, we backtrack from our back list row by row backward. For example for first test in our Viterbi test case the backtrack list is like this:



Index	Type	Size	
0	list	6	[0, 0, 0, 2, 2, 0]
1	list	6	[0, 1, 0, 2, 2, 0]
2	list	6	[0, 1, 0, 2, 2, 0]

The last column is zero so it's for ending score tag, after adding that to our path_list(in this example it's 2) then we have to go to row number two and column(end -1) and read the tag which is 2 here and then like the picture we go row by row and column by column(in backward) to find the path. Finally we add starting tag to our list and return the final path.