Daily Coding Problem

Good morning! Here's your coding interview problem for today.

This problem was asked by Facebook.

Boggle is a game played on a `4 x 4` grid of letters. The goal is to find as many words as possible that can be formed by a sequence of adjacent letters in the grid, using each cell at most once. Given a game board and a dictionary of valid words, implement a Boggle solver.

Arranging the board

Straight away, it can be seen that without even performing calculations that is a permutational exercise. I will ensure I clarify this.
This is because the order of the letters on the board affects the adjacent connectivity to chain the letters and generate words. I am also certain since I am not from a Mathematics background, there will be lots calculations which are incorrect. But I will try to demonstrate my rationale.

And also it would be without replacement since from each 16 dice, only one side can be up facing at a time.
If the dice had same value on all sides, this is number way arranging each letter in each position on the boggle (4x4). **20,922,789,888,000** . I am certain this calculation is correct.

Permutations Calculator nPr

Permutations nPr Calculator

$$P(n,r) = \frac{n!}{(n-r)!}$$

n *(objects)* = 16

r *(sample)* = 16

Clear Calculate

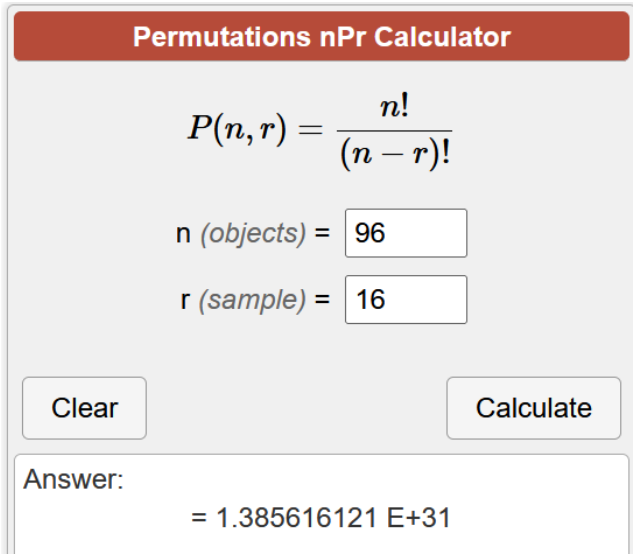
Answer:

= 2.092278988 E+13

It also potentially should be bigger since there are 6 values per dice. So the total available letters (‘Qu’ is taken to be a letter at this moment is 6 x 16 = 96

Now the following would cover all possible permutations of selecting 16 letters from an available 96. I am fairly certain that this calculation is correct.

P(96,16) =



So for this exercise, it will just perform a randomisation of each dice and input it onto the board. Perhaps in a similar way to suduko board. Since the value is too computationally high. However a more positive outcome is that it will suffice just to perform operation row by row in filling the Boggle (4x4 grid) since there are no validation rules in regards to how letters appear. This will be just as valid as randomisation.

Traversing the board

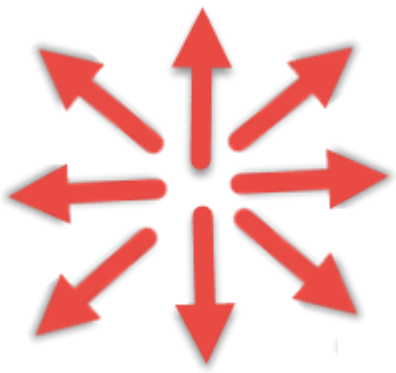
It can be traversed similar to Zillow coin collecting example.

However there are vast more number of moves possible (8 in total). It can still use the technique to modify method signature of the constructor in order to perform correct direction.

It can still be focussed on performing a maximum movement through 16 squares (start=> end) excluding the start (similar to Zillow). This would be 15 selected moves.

However the start position would need to be completed for each square.

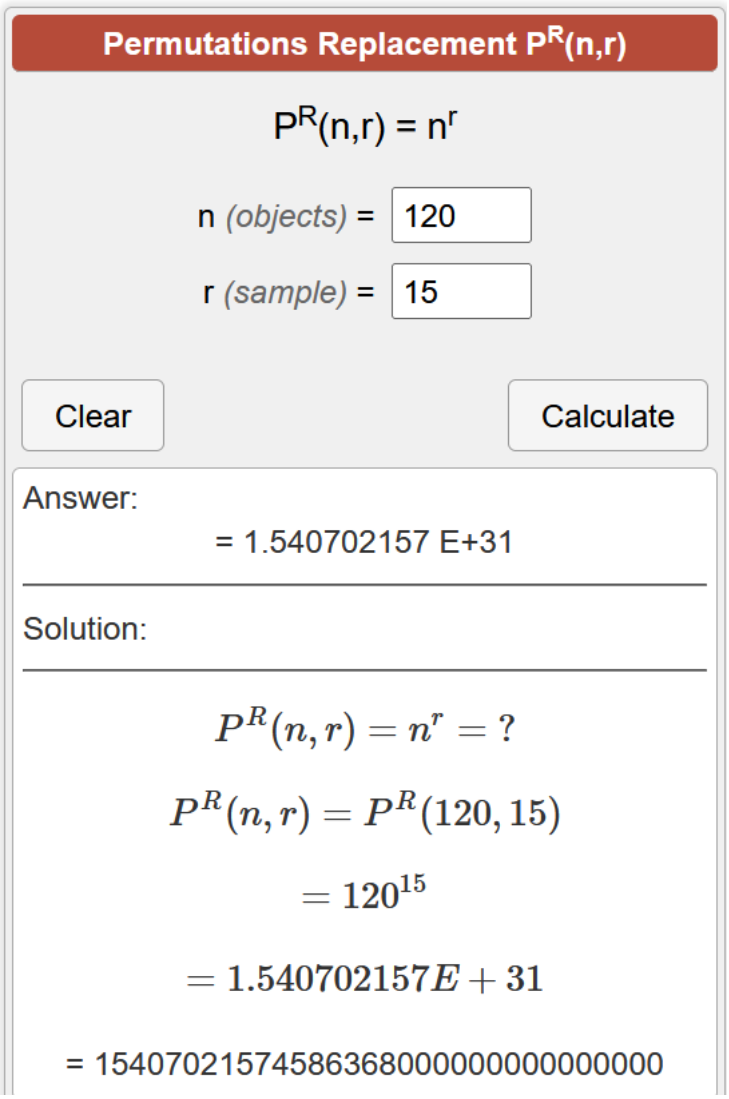
ALL AVAILABLE DIRECTIONS



Also it would need to ensure that the path does not move across any existing squares already passed through. It can be avoided by marking the square on 2D String array with "**" or similar.
Note, it is a String array since once of the values is 'Qu'.

If every single movement was possible in each direction from each square (we know its not possible), it would consider all 8 movements from 15 squares in which it has to make a decision... = 120. This would be P(120,15). But this is not the nature of the question since the directions might not necessarily link all 16 squares. At same time, we will not know if they are valid unless it generates all these permutations for 15 directions…

This most definitely creates a tricky situation since value is too high



To keep this solution computational, it will consider all single moves and cascade.
Since **this is the nature of the problem.**

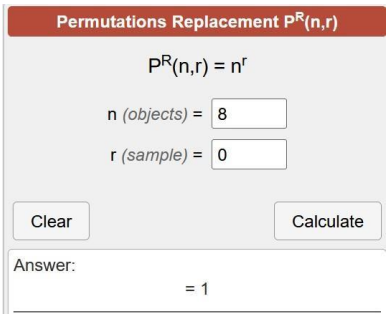
The starting point can be anywhere on the 4 x 4 grid.
Since we know there are 8 directions and it is required to perform maximum 15 directional changes(worst case scenario, however unlikely this is possible).

This is the extreme testing in which superseding aspect **cannot** prevail at all.
Since it is changing direction each time!
In practice this can be fulfilled if P(8,15) is computational since it will then perform single move between the squares.

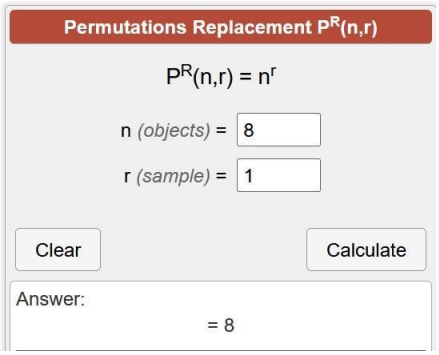
At this point, it needs to be understood the challenge are vectors (it has size and direction). The sizes of direction would be difficult to ascertain unlike the Zillow coin collecting example since there is no official start or finish.

So it will perform a for loop (0=>15) in the Permutation class. This is in reference to initiation at each square on 4 x 4

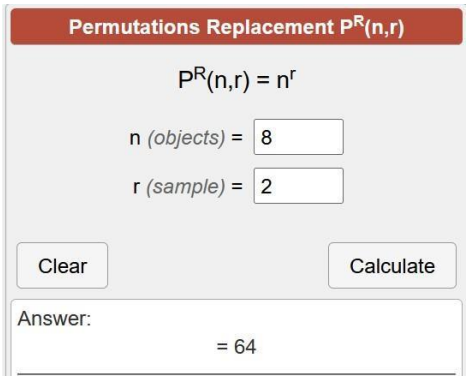
Inside for loop it will perform
this loop would simply create chain such as
DOWN=> RIGHT=> DIAGONAL RIGHT (when r=3)
(for r=0 , r<=15, r++)
P(8 directions, 0) = seek all outcomes 1



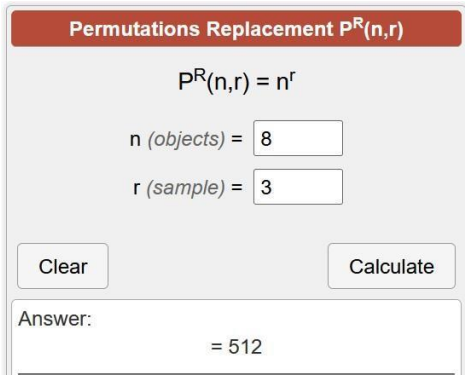
P(8 directions, 1) = seek all outcomes 8



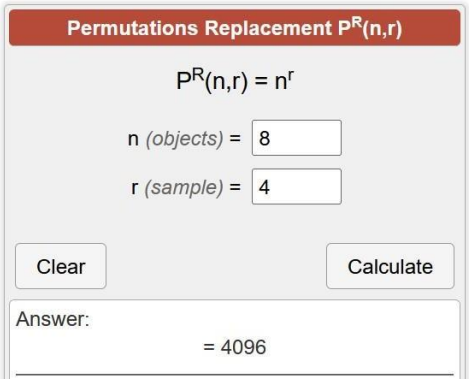
P(8 directions, 2)



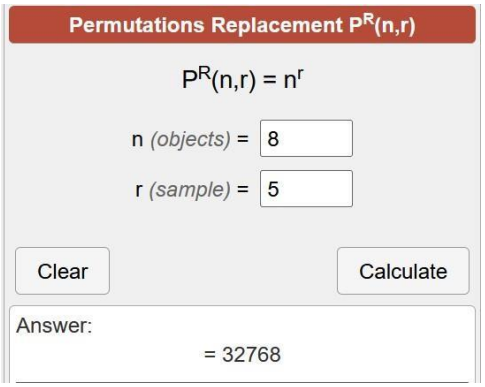
P(8 directions, 3)



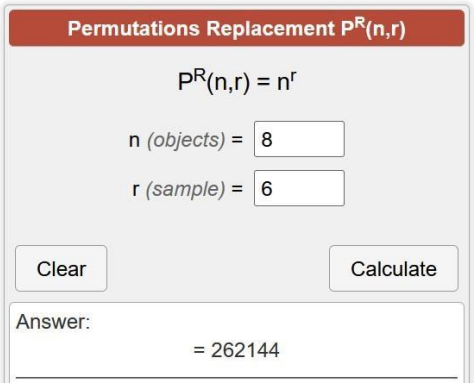
P(8 directions, 4)



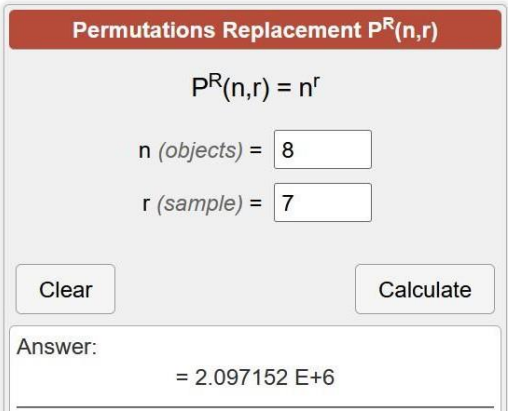
P(8 directions, 5)



P(8 directions, 6)



P(8 directions, 7)



It can be seen that this will stretch the limit of the computation.

So I will be able to complete 6 directional changes, which will give traversal of 7 squares. This is equivalent to a 8 letter word (given that one of the cubes has Qu).

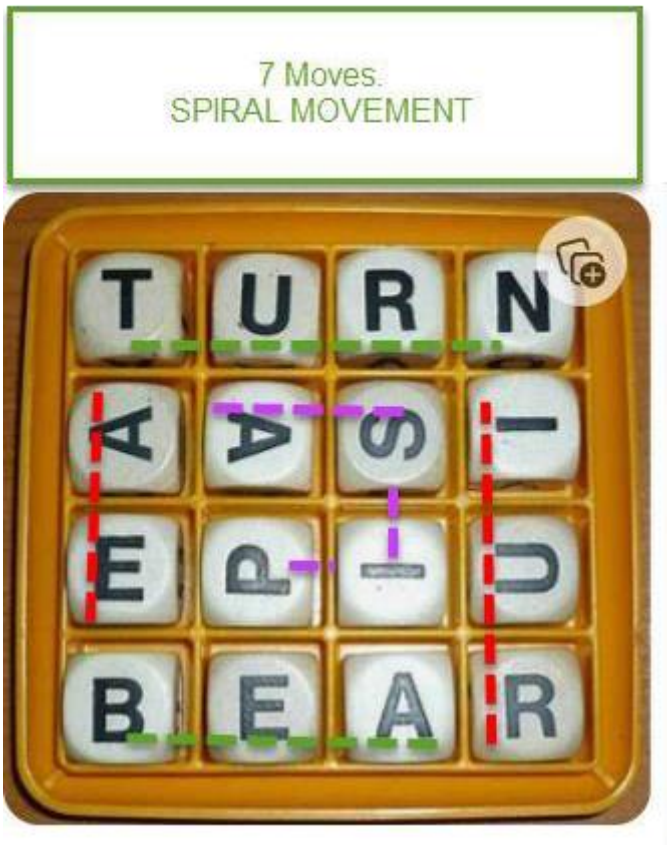
We know maximum move in any direction is 3. So I would need to repeat the above (in red) from each square (outer for loop) individually for direction sizes of 1,2, 3.
Let's say when processing: P(8,6) it might generate following:

DOWN=>RIGHT=>DIAGONAL LEFT => DIAGONAL RIGHT => UP => LEFT
It also needs to consider each size either being 1,2,3 and number permutations with these movements. **THIS IS THE ONLY WAY THE TRAVERSAL CAN SUCCEED ON THE CONSTRAINED**

4x4 arangment.

Areas like this I am not entirely unsure of the mathematical calculation since it has been extreme long time since I used this logic. However number is guaranteed to be much higher and not computational.

The problem with this approach (in red)and in fact across all my given approaches due to limitations and improvisations is that it will be capped at moving through 7 squares. So even the simple approach such as performing a spiral might be missed out if I plan too strategically.



CONCLUSION:
Only way is to vary the move sizes… but no strategy can be applied.

So as far as this challenge is concerned, I can either perform this challenge with r=0 to r=6 P (8,6) with all ways of combining directions such as and using direction size =1 (Maximum word length = 8 letters)

Down
Down => Right
Down =>Diagonal Right => Diagonal Left
Down=>Left=>Down=>Right
Down=>Diagonal right => Diagonal left => Right => Up
Down => Right => Up=> Diagonal right => Left => Down

OR

I can adapt above approach and find improvised technique to modify the size. I would not be able to reach 17 letter word, but its likely to be larger than 8 letters. The direction undertaken would total 15 from each start position.