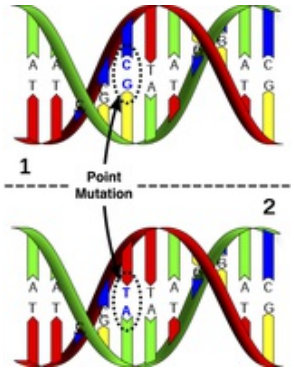
Consensus sequence

A **mutation** is simply a mistake that occurs during the creation or copying of a nucleic acid, in particular DNA. Because nucleic acids are vital to cellular functions, mutations tend to cause a ripple effect throughout the cell. Although mutations are technically mistakes, a very rare mutation may equip the cell with a beneficial attribute. In fact, the macro effects of evolution are attributable to the accumulated result of beneficial microscopic mutations over many generations.



The simplest and most common type of nucleic acid mutation is a **point mutation**, which replaces one base with another at a single nucleotide. In the case of DNA a point mutation must change the complementary base accordingly, as in the figure where a C-G pair is changed into an A-T pair.

DNA strands taken from different organism or species genomes are **homologous** if they share a recent ancestor. In comparing several homologous DNA strands, it might be helpful to compute their **consensus sequence**. After all, according to the biological principle of *parsimony* — which demands that evolutionary histories should be as simply explained as possible — this sequence represents the most likely ancestor of the given DNA strands.

Assignment

A **matrix** is a rectangular table of values divided into rows and columns. An $m \times n$ matrix has m rows and n columns. Given a matrix A, we write $A_{i,j}$ ($0 \leq i < m; 0 \leq j < n$) to indicate the value at the intersection of row i and column j.

Say that we have a series of DNA sequences, all having the same length n. Their **profile matrix** is a $4 \times n$ matrix P in which $P_{0, j}$ represents the number of times the base A occurs in the j-th position of the given sequences, $P_{1, j}$ represents the number of times the base C occurs in the j-th position of the given sequences, and so on (see table below).

The **consensus sequence** c is a string of length n formed from the series of DNA sequences by taking the most common base at each position. The j-th character of c therefore corresponds to the base having the maximal value in the j-th column of the profile matrix of the DNA sequences. If there is more than one maximal value in the j-th column of the profile matrix, the letter N is used as the j-th character of c.

	G C A A A A C G
	G C G A A A C T
	T A C C T T C A
sequences	T A T G T T C A
	G C C T T A G G
	G A C T T A T A

		T	C	G	G	A	T	C	C
	A	0	3	1	2	3	4	0	3
profile	**C**	0	4	3	1	0	0	5	1
	G	4	0	2	2	0	0	1	2
	T	3	0	1	2	4	3	1	1
consensus		G	C	C	N	T	A	C	A

Your task:

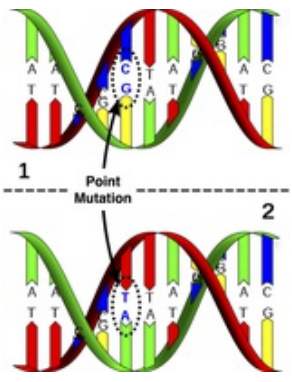
- Write a function `profile` that takes a series (a list or a tuple) of DNA sequences as its argument. In this assignment, DNA sequences are represented as strings that only contain the uppercase letters `A`, `C`, `G` and `T`. In case not all DNA sequences have equal length, the function must raise an `AssertionError` with the message `sequences should have equal length`. In case all DNA sequences have equal length n, the function must return the profile matrix of the sequences. The profile matrix is represented as a dictionary that maps each of the bases `A`, `C`, `G` and `T` onto a list of n integers, where the integer at position j indicates how often that base occurs in the j-th position of the given DNA sequences.
- Write a function `consensus` that takes a profile matrix as its argument. This profile matrix must be a dictionary that is formatted as the return values of the function `profile`. The function `consensus` must return the consensus sequence that corresponds to the given profile matrix.

Example

```
>>> seqs = ['GCAAAACG', 'GCGAAACT', 'TACCTTCA', 'TATGTTCA', 'GCCTTAGG', 'GACTTATA', 'TCGGATCC']
>>> profile(seqs)
{'A': [0, 3, 1, 2, 3, 4, 0, 3], 'C': [0, 4, 3, 1, 0, 0, 5, 1], 'T': [3, 0, 1, 2, 4, 3, 1, 1], 'G': [4, 0, 2, 2, 0, 0, 1, 2]}
>>> consensus(profile(seqs))
'GCCNTACA'

>>> seqs = ['GGTATCTTTA', 'TTGTCGTCTTAGA', 'GGATCCAGAC', 'ATTCAATCGA', 'TGATCTGGAA', 'AGAGTCATGC']
>>> profile(seqs)
Traceback (most recent call last):
AssertionError: sequences should have equal length
```

Een **mutatie** is niets anders dan een fout die optreedt tijdens het maken of kopiëren van een nucleïnezuur, in het bijzonder DNA. Omdat nucleïnezuren een vitale rol spelen bij cellulaire functies, hebben mutaties de neiging om een cel grondig dooreen te schudden. Maar hoewel mutaties technisch gezien als fouten kunnen aanzien worden, kunnen ze in zeer zeldzame gevallen een voordeel opleveren voor de cel. De macro-effecten van evolutie zijn dan ook toe te schrijven aan het cumulatieve resultaat van voordelige microscopische mutaties die optreden doorheen een groot aantal generaties.



Puntmutaties zijn de eenvoudigste een meest voorkomende vorm van nucleïnezuurmutaties, waarbij in één enkele nucleotide een base wordt vervangen door een andere base. In het geval van DNA verandert een puntmutatie ook de complementaire base, zoals in de figuur waarbij een C-G paar verandert in een A-T paar.

Stukjes DNA die afkomstig zijn uit het genoom van verschillende organismen of soorten worden **homoloog** genoemd als ze een recente voorouder hebben. Bij het vergelijken van meerdere homologe DNA-sequenties kan het handig zijn om hun **consensussequentie** te berekenen. Volgens het biologische principe van *parsimony* — de natuur neigt naar het kortst mogelijke evolutionaire pad — stelt deze sequentie immers de meest waarschijnlijke voorouder van de homologe DNA-sequenties voor.

Opgave

Een **matrix** is een rechthoekig rooster van waarden die gerangschikt zijn in rijen en kolommen. Een $m \times n$ matrix heeft m rijen en n kolommen. Gegeven een matrix A, dan schrijven we $A_{i,j}$ ($0 \leq i < m; 0 \leq j < n$) om de waarde aan te duiden die gevonden wordt op de intersectie van rij i en kolom j.

Veronderstel dat we beschikken over een reeks DNA-sequenties die allemaal dezelfde lengte n hebben. Hun **profielmatrix** is een $4 \times n$ matrix P waarin $P_{0, j}$ uitdrukt hoeveel keer de base A voorkomt op de j-de positie van de gegeven sequenties, $P_{1, j}$ uitdrukt hoeveel keer de base C voorkomt op de j-de positie van de gegeven sequenties, enzoverder (zie onderstaande tabel).

De **consensussequentie** c is een string van lengte n die gevormd wordt door in de reeks DNA-sequenties op elke positie de meest voorkomende base te kiezen. Het j-de karakter van c correspondeert dus met de base die de grootste waarde heeft in de j-de kolom van de profielmatrix van de DNA-sequenties. Indien er meerdere basen zijn die de grootste waarde hebben in de j-de kolom van de profielmatrix, dan wordt de letter N gebruikt als j-de karakter van c.

sequenties		G C A A A A C G
		G C G A A A C T
		T A C C T T C A
		T A T G T T C A
		G C C T T A G G
		G A C T T A T A
		T C G G A T C C
profiel	**A**	0 3 1 2 3 4 0 3
	C	0 4 3 1 0 0 5 1
	G	4 0 2 2 0 0 1 2
	T	3 0 1 2 4 3 1 1
consensus		G C C N T A C A

Gevraagd wordt:

- Schrijf een functie profiel waaraan een reeks (een lijst of een tuple) DNA-sequenties moet doorgegeven worden. In deze opgaven stellen we een DNA-sequentie voor als een string die

enkel bestaat uit de hoofdletters A, C, G en T. Indien niet alle DNA-sequenties dezelfde lengte hebben, dan moet de functie een AssertionError opwerpen met de boodschap sequenties moeten even lang zijn. Indien alle DNA-sequenties wel dezelfde lengte n hebben, dan moet de functie de profielmatrix van de sequenties teruggeven. De profielmatrix wordt voorgesteld als een dictionary die elk van de basen A, C, G en T afbeeldt op een lijst van n natuurlijke getallen, waarbij het getal op de j-de positie aangeeft hoeveel keer die base voorkomt op de j-de positie van de gegeven DNA-sequenties.

- Schrijf een functie consensus waaraan een profielmatrix moet doorgegeven worden. De profielmatrix moet een dictionary zijn die opgebouwd is zoals de waarden die door de functie profiel teruggegeven worden. De functie consensus moet de consensussequentie teruggeven die correspondeert met de gegeven profielmatrix.

Voorbeeld

```
>>> seqs = ['GCAAAACG', 'GCGAAACT', 'TACCTTCA', 'TATGTTCA', 'GCCTTAGG', 'GACTTATA', 'TCGGATCC']
>>> profiel(seqs)
{'A': [0, 3, 1, 2, 3, 4, 0, 3], 'C': [0, 4, 3, 1, 0, 0, 5, 1], 'T': [3, 0, 1, 2, 4, 3, 1, 1], 'G': [4, 0, 2, 2, 0, 0, 1, 2]}
>>> consensus(profiel(seqs))
'GCCNTACA'

>>> seqs = ['GGTATCTTTA', 'TTGTCGTCTTAGA', 'GGATCCAGAC', 'ATTCAATCGA', 'TGATCTGGAA', 'AGAGTCATGC']
>>> profiel(seqs)
Traceback (most recent call last):
AssertionError: sequenties moeten even lang zijn
```