Secuencia de ADN

∗ Escribe una clase `DNA` que represente una secuencia de ADN ∗

Implementa, *al menos*, los siguientes **métodos**:

```
def __init__(self, sequence: str):
```

- Crea el atributo `sequence` donde se almacena la secuencia de *bases nitrogenadas*.
- Las bases nitrogenadas se representarán por sus iniciales (como *string*):

Adenina	`'A'`
Timina	`'T'`
Guanina	`'G'`
Citosina	`'C'`

```
def __len__(self) -> int:
```

- Devuelve la longitud de la secuencia `self`.

```
def __str__(self) -> str:
```

- Devuelve la representación *string* de la secuencia `self`.

```
def adenines(self) -> int:
```

- Es una **propiedad**.
- Devuelve el **número de adeninas** de la secuencia `self`.

```
def cytosines(self) -> int:
```

- Es una **propiedad**.
- Devuelve el **número de citosinas** de la secuencia `self`.

```
def guanines(self) -> int:
```

- Es una **propiedad**.
- Devuelve el **número de guaninas** de la secuencia `self`.

```
def thymines(self) -> int:
```

- Es una **propiedad**.
- Devuelve el **número de timinas** de la secuencia `self`.

```
def __add__(self, other: DNA) -> DNA:
```

- Devuelve una *nueva secuencia* como suma de `self` y `other`.
- Para obtener la suma de dos secuencias habrá que ir recorriendo las secuencias simultáneamente y quedarse con la mayor base (*lexicográficamente*).
- Si las secuencias tienen tamaños diferentes, habrá que añadir al final de la secuencia resultante el "trozo" que falte de la secuencias más larga.

```
def stats(self) -> dict[str, float]:
```

- Devuelve un **diccionario** donde las *claves* son las bases nitrogenadas y los *valores* son el porcentaje de aparición de cada base en la secuencia `self`.

```
def __mul__(self, other: DNA) -> DNA:
```

- Devuelve la "multiplicación" de dos secuencias.
- La secuencia resultante contendrá aquellas bases comunes (*posición a posición*) de `self` y `other`.

```
def build_from_file(cls, path: str) -> DNA:
```

- Es un **método de clase**.
- Devuelve una nueva secuencia a partir del fichero con ruta `path`.
- El fichero sólo contendrá una única línea con una sucesión de *iniciales de bases nitrogenadas*.

```
def dump_to_file(self, path: str) -> None:
```

- Es un **método de clase**.
- Vuelca la secuencia `self` al fichero `path` en una única línea.
- El volcado sólo incluye las **iniciales** de cada base nitrogenada.

```
def __getitem__(self, index: int) -> str:
```

- Devuelve la *base nitrogenada* que ocupa la posición `index` en la secuencia.

```
def __setitem__(self, index: int, base: str) -> None:
```

- Asigna la *base nitrogenada* `base` a la posición `index` de la secuencia.
- Si la base que se va a asignar **no existe** entonces se asignará la **adenina**.

