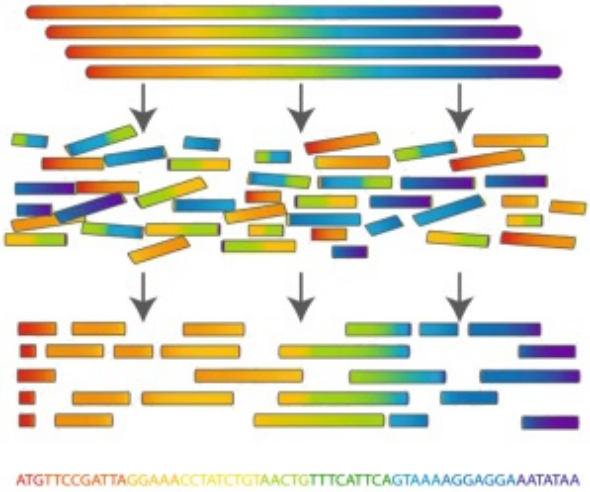
N50

Determining an organism's complete genome (called **genome sequencing**) forms a central task of bioinformatics. Unfortunately, we still don't possess the microscope technology to zoom into the nucleotide level and determine the sequence of a genome's nucleotides, one at a time. However, researchers can apply chemical methods to generate and identify much smaller snippets of DNA, called **reads**. After obtaining a large collection of reads from multiple copies of the same genome, the aim is to reconstruct the desired genome from these small pieces of DNA. This process is called **genome assembly**.



Genome assembly works by blasting many copies of the same genome into smaller, identifiable reads, which are then used to computationally assemble one copy of the genome.

From a computational perspective, genome assembly is an extremely difficult problem. It becomes even harder, because many of the genomes contain large numbers of identical sequences that are repeated at various locations in the genome. Such **repetitions** often stretch over thousands of nucleotides, and some are found at thousands of different positions across the genome. This especially occurs in the large genomes of plants and animals. As a result, it is often impossible to reconstruct the complete genome and the assembly process ends with a number of large pieces of the genome which are called **contigs** in this context.

Assignment

To determine the quality of a genome assembly (for example, when comparing two different assembly algorithms that separately have computed contigs starting from the same set of reads), the **N50 statistic** is often used.

Suppose that after assembly of a genome of length g we obtain a sequence of contigs, and that we sort the lengths of these contigs in decreasing order. We then start adding the lengths (in decreasing order). The first length that causes the sum to become larger or equal to $\frac{g}{2}$ (half the length of the genome) gives the first approximation of the N50 statistic. A second approximation is obtained by applying a similar procedure, where the lengths of the contigs are added in an increasing order. The N50 statistic is then computed as the mean of both approximations. In some cases the length of the genome is unknown, in which case the sum of all contig lengths is used as an estimate for the genome length g.

For example, consider the list of contig lengths $[2, 2, 2, 3, 3, 4, 8, 8]$. We can estimate the

genome length as $2 + 2 + 2 + 3 + 3 + 4 + 8 + 8 = 32$. For this example, the first approximation of the N50 statistic (decreasing order) gives the value 8, because $8 + 8 = 16 \geq \frac{32}{2}$. The second approximation (increasing order) gives the value 4, since $2 + 2 + 2 + 3 + 3 + 4 = 16 \geq \frac{32}{2}$. Based on both approximations, we can calculate the N50 statistic as $\frac{8 + 4}{2}$ = 6.

Your task:

- Write a function `N50_decreasing` that takes a collection (list, tuple, collection, …) of integers. These integers represent the lengths of the contigs obtained after genome assembly. The function also has a second optional parameter `size` to which the genome size can be passed. If no value is passed to this parameter, the sum of the given contig lengths is used as an estimate of the genome size. The function must return the approximation of the N50 statistic (as an integer) that is obtained if the procedure described above is applied with the contig lengths added in decreasing order.
- Write a function `N50_increasing` that takes same parameters as the function `N50_decreasing`, with the same meaning. The function must return the approximation of the N50 statistic (as an integer) that is obtained if the procedure described above is applied with the contig lengths added in increasing order.
- Use the functions `N50_decreasing` and `N50_increasing` to write a function `N50` that has the same parameters as the previous two functions, with the same meaning. The function must return the N50 statistic of the given collection of contig lengths as a *floating point* number.

Example

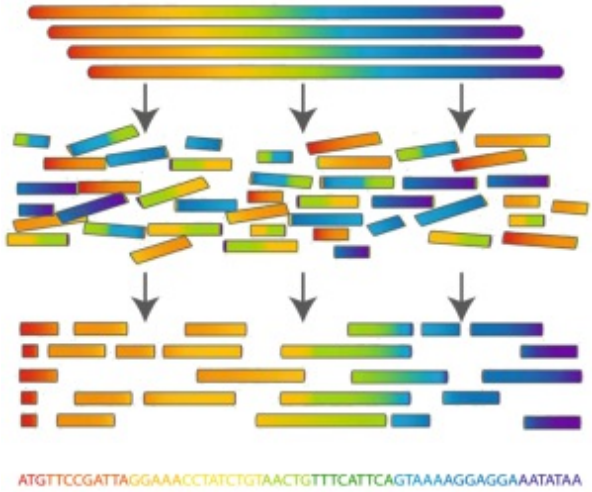
```
>>> contigs = (2, 2, 2, 3, 3, 4, 8, 8)
>>> N50_decreasing(contigs)
8
>>> N50_increasing(contigs)
4
>>> N50(contigs)
6.0

>>> contigs = (5, 8, 6, 4, 6, 1, 1)
>>> N50_decreasing(contigs, size=40)
6
>>> N50_increasing(contigs, size=40)
6
>>> N50(contigs, 40)
6.0
```

Sources

- **Miller JR, Koren S, Sutton G (2010)**. Assembly algorithms for next-generation sequencing data. *Genomics* **95(6)**, 315-327.

Het bepalen van het volledige genoom van een organisme (**genoomsequenering**) is één van de hoekstenen van de bioinformatica. Helaas beschikken we nog steeds niet over microscopische technologie die toelaat om tot op nucleotideniveau in te zoomen en zo de reeks nucleotiden van een genoom één voor één af te lezen. Onderzoekers beschikken echter wel over chemische methoden die kunnen toegepast worden voor het genereren en identificeren van kortere DNA fragmenten. In deze context worden de korte fragmenten **reads** genoemd. Nadat we

een grote collectie van reads geïdentificeerd hebben uit meerdere kopieën van hetzelfde genoom, blijft de uitdaging om het genoom te reconstrueren uit deze korte stukjes DNA. Dit proces wordt **genoomassemblage** genoemd.



Genoomassemblage gebeurt door een groot aantal kopieën van hetzelfde genoom op te breken in kleinere, identificeerbare reads, die dan gebruikt worden om het genoom computationeel terug samen te stellen.

Genoomassemblage is computationeel gezien een extreem moeilijk probleem. Het wordt nog moeilijker doordat heel veel genomen een groot aantal identieke sequenties bevatten die op verschillende plaatsen in het genoom herhaald worden. Dergelijke **herhalingen** zijn vaak duizenden nucleotiden lang en sommige kunnen op duizenden verschillende plaatsen in het genoom voorkomen. Dit komt vooral voor in de grote genomen van planten en dieren. Hierdoor is het vaak niet mogelijk om het volledige genoom te reconstrueren, en eindigt het assemblageproces met een aantal grote stukken van het genoom die in deze context **contigs** genoemd worden.

Opgave

Om de kwaliteit van een genoomassemblage te bepalen (bijvoorbeeld om twee verschillende assemblage-algoritmen te vergelijken, die elk afzonderlijk contigs bepaald hebben vertrekkend van dezelfde verzameling reads) wordt vaak gebruik gemaakt van de **N50 statistiek**.

Stel dat we voor een genoom van lengte g na assemblage een reeks contigs hebben bekomen, en dat we de lengtes van die contigs sorteren van groot naar klein. Daarna beginnen we de lengtes bij elkaar op te tellen (in dalende volgorde). De eerste lengte die ervoor zorgt dat de som groter of gelijk wordt aan $\frac{g}{2}$ (de helft van de lengte van het genoom) levert ons een eerste benadering van de N50 statistiek op. Een tweede benadering wordt bekomen door dezelfde procedure toe te passen, maar dan door de contiglengtes in stijgende volgorde bij elkaar op te tellen. De N50 statistiek is dan het gemiddelde van deze twee benaderingen. Soms is de lengte van het genoom niet gekend. In dat geval neemt men doorgaans de som van alle contiglengtes als schatting van de genoomlengte g.

Stel bijvoorbeeld dat de lijst van contiglengtes gelijk is aan $[2, 2, 2, 3, 3, 4, 8, 8]$. We kunnen de genoomlengte schatten als $2 + 2 + 2 + 3 + 3 + 4 + 8 + 8 = 32$. In dit geval levert een eerste benadering van de N50 statistiek (dalende volgorde) de waarde 8 op, want $8 + 8 = 16 \geq \frac{32}{2}$. Een tweede benadering (stijgende volgorde) levert de waarde 4 op, want $2 + 2 + 2 + 3 + 3 + 4 = 16 \geq \frac{32}{2}$. Op basis van deze twee benaderingen kunnen we de N50

statistiek dus berekenen als $\frac{8 + 4}{2}$ = 6.

Gevraagd wordt:

- Schrijf een functie `N50_dalend` waaraan een collectie (een lijst, tuple, verzameling, …) van natuurlijke getallen moet doorgegeven worden. Deze getallen stellen de lengtes voor van de contigs die bekomen worden na genoomassemblage. De functie heeft een tweede optionele parameter `grootte` waaraan de genoomgrootte kan doorgegeven worden. Indien geen waarde wordt doorgegeven aan deze parameter, dan wordt de som van de contiglengtes gebruikt als schatting voor de genoomgrootte. De functie moet de benadering van de N50 statistiek teruggeven (als natuurlijk getal), die resulteert als de procedure wordt toegepast die hierboven staat beschreven en waarbij een lopende som gebruikt wordt over de contiglengtes die in **dalende volgorde** gesorteerd zijn.
- Schrijf een functie `N50_stijgend` die dezelfde parameters heeft als de functie `N50_dalend`, met dezelfde betekenis. De functie moet de benadering van de N50 statistiek teruggeven (als natuurlijk getal), die resulteert als de procedure wordt toegepast die hierboven staat beschreven en waarbij een lopende som gebruikt wordt over de contiglengtes die in **stijgende volgorde** gesorteerd zijn.
- Gebruik de functies `N50_dalend` en `N50_stijgend` om een functie `N50` te schrijven die dezelfde parameters heeft als de twee vorige functies, met dezelfde betekenis. De functie moet de N50 statistiek voor de gegeven collectie van contiglengtes teruggeven als een *floating point* getal.

Voorbeeld

```
>>> contigs = (2, 2, 2, 3, 3, 4, 8, 8)
>>> N50_dalend(contigs)
8
>>> N50_stijgend(contigs)
4
>>> N50(contigs)
6.0

>>> contigs = (5, 8, 6, 4, 6, 1, 1)
>>> N50_dalend(contigs, grootte=40)
6
>>> N50_stijgend(contigs, grootte=40)
6
>>> N50(contigs, 40)
6.0
```

Bronnen

- **Miller JR, Koren S, Sutton G (2010)**. Assembly algorithms for next-generation sequencing data. *Genomics* **95(6)**, 315-327.