I have a problem that needs to be solved

here is my attempt of describing the problem

input: Canonical Smiles code of comb-like polymer with only Carbon. it will have a backbone, and sidechains, where each node of the backbone only produce at most 1 sidechain. The length of the backbone and the sidechains are not limited.

EX: CCCC(CCCC)CCCC

this is a saturated comb polymer with only Carbon, the backbone length is 8 and the side chain is at location 4 and the length of the sidechain excluding the backbone node is 4

After i input ther canonical smiles, here is what we are working with:

Atomic Properties (Atom, Atomic Mass, Charge, Ring, Edge Node):

['C', 12.011, 0, False, True]

['C', 12.011, 0, False, False]

['C', 12.011, 0, False, False]

['C', 12.011, 0, False, False]

['C', 12.011, 0, False, False]

['C', 12.011, 0, False, False]

['C', 12.011, 0, False, False]

['C', 12.011, 0, False, True]

['C', 12.011, 0, False, False]

['C', 12.011, 0, False, False]

['C', 12.011, 0, False, False]

['C', 12.011, 0, False, True]

[[0 1 0 0 0 0 0 0 0 0 0 0]

[1 0 1 0 0 0 0 0 0 0 0 0]

[0 1 0 1 0 0 0 0 0 0 0 0]

[0 0 1 0 1 0 0 0 1 0 0 0]

[0 0 0 1 0 1 0 0 0 0 0 0]

[0 0 0 0 1 0 1 0 0 0 0 0]

[0 0 0 0 0 1 0 1 0 0 0 0]

[0 0 0 0 0 0 1 0 0 0 0 0]

[0 0 0 1 0 0 0 0 0 1 0 0]

[0 0 0 0 0 0 0 0 1 0 1 0]

[0 0 0 0 0 0 0 0 0 1 0 1]

[0 0 0 0 0 0 0 0 0 0 1 0]]

I have 2 arrays. The 1st array is an array of arrays that describe each big atom, here we only have C. it will have the Atom, Atomic Mass, Charge, is it part of a Ring, and is it an edge node.

The second array is a connectivity matrix.

The function is create these arrays are already made on the following code:

```
from rdkit import Chem

import numpy as np

from setup import get_atom_properties, connectivity_matrix

#Input SMILES string

smiles = input("Please enter your Smiles String: ")

mol = Chem.MolFromSmiles(smiles)

atom_properties = get_atom_properties(mol)
```

connectivity_matrix = connectivity_matrix(mol, len(atom_properties))

print("\nAtomic Properties (Atom, Atomic Mass, Charge, Ring, Edge Node):")

for i in range(len(atom_properties)):

print(atom_properties[i])

print(connectivity_matrix)

So now, technically, we have 3 different inputs that we can use.

Task 1: Create a text file that visually represent this comb polymer. Here is the details:

The text file will show all of the connections (and the amount) of the big atom, it will ignores all Hydrogens and connectivity angles. We will have another input asking: what is the name of this compound and the text file will be saved as {name}.txt

For example, CCCC(CCCC)CCCC can possibly be saved as:
[[C,-,C,-,C,-,C,-,C,-,C,-,C,-,C],

[_,_,_,_,_,_,-,_,_,_,_,_,_,_,_],

[_,_,_,_,_,_,C,_,_,_,_,_,_,_,_],

[_,_,_,_,_,_,-,_,_,_,_,_,_,_,_],

[_,_,_,_,_,_,C,_,_,_,_,_,_,_,_],

[_,_,_,_,_,_,-,_,_,_,_,_,_,_,_],

[_,_,_,_,_,_,C,_,_,_,_,_,_,_,_],

[_,_,_,_,_,_,-,_,_,_,_,_,_,_,_],

[_,_,_,_,_,_,C,_,_,_,_,_,_,_,_]]

Or simply as

C-C-C-C-C-C-C-C

-

C

-

C

-

C

-

C

Or it can be saved in a different file format that can best represent the structure.

Note: the smiles structure uses = to represent double bonds and # to represent triple bonds, in our presentation we will use the same symbols

Task 2: Create a dictionary

Imagine there are Beads. Beads are conceptual groups of atoms are composed of 2 to 4 big atoms. Here is what I have so far:
def get_c_only_dict(): c_dict = {}

```
c_dict["+C=C+"] = ["TC5"] #?
c_dict["C=C+"] = ["TC5"]
c_dict["CC=CC"] = ["C4"]
c_dict["++CC(C)=C+"] = ["SC4"]
c_dict["++C(C)+"] = ["TC4"]
c_dict["+CC(C)C+"] = ["SC3"]
c_dict["+CCC+"] = ["SC3"]
c_dict["+CC+"] = ["TC3"]
c_dict["CC+"] = ["TC3"]
c_dict["CCC+"] = ["SC2"]
c_dict["CCCC"] = ["C1"]

return c_dict
```

This is simply a dictionary with some possible “Beads” with arbitrary names, Let’s create a dictionary with the following constraints:
Create a dictionary following the above function layout with the following rules:

1. The key:
    a. It will have at most 4 Cs and at least 2 Cs
    b. It mimics Canonical SMILES with the following changes

- There will be some keys with no “+”, and these keys will represent a complete molecule because there will be no continuation from these keys
- + represent a continuation with 1 bond. For example. The bead with the chemical connection C-C-C-C- will be represented as CCCC+.
- ++ represent a continuation with 2 bonds. For example. The bead with the chemical connection C-C-C-C= will be represented as CCCC++.
- There will be no +++, but rather we will add 5 different keys that can represent all types of triple bonds and named the values from X1 to X7, namely:
  [“C#C”], [“C#C+”], [“C#CC”], [“+C#C+”], [“+CC#C+”], [“CC#C+”], and [“CC#CC”]
  this make sure that all molecules that contain a triple bonds(#) can be represented by “Beads” with at least 2 or more Cs, since we know for sure that a triple bond can only be followed by a singular bond.
- If it has 1 group of “+” or “++”, that “+” or “++” will be at the end and that will denote this “bead” has at least 1 edge node on the other side of the “+”, and the + denotes that the bead will continue from there.
- If it has 2 groups of “+” or “++”, that “+” or “++” will be at the end and the front and that will denote this “bead” has no edge node. and the + denotes that the bead will continue from there. There are a few exceptions that is when a bead is connected to a sidechain length of 1 and these beads will be named SS1 with the key value [“+C(C)+”], SS2 with the key value [“+C(=C)+”], and SS3 with the key value [“++C(C)+”]. SS1 is when the node of the backbone connected to the sidechain has singular bonds across all nodes, SS2 is when there is a double bond leading to the sidechain, and SS3 is when there is a double bond that is leading to the other node in the backbone
- If it has 3 groups of “+” or “++”, this bead is clearly a connection node between the backbone and the sidechains.

2. The value:
   - The first character will represent the size of bead:
     - T represents tiny beads(2 big atoms)
     - S represents small beads(3 big atoms)
     - 4 big atoms will not have any characters
   - The second character will represent the type of bead:
     - F: Full beads: the whole molecule (for molecule with 1-4 big atoms only)
     - E: Edge beads: nodes with 1 group of +

    - C: Continuation beads: beads on the backbone with 2 groups of +
    - S: Side chain beads: beads with 3 groups of +
    - Special beads:
        - F1: [“C”] this will be the only bead with 1 big atom
        - X: these beads have triple bonds
        - SS: these beads have sidechains with length 1
        - Please create more special beads for edge cases as you need
- The next character will be lowercase letter that represents:
    - b : this bead has at least a double bond
- The last character will be a number that will just differentiate very similar beads
- Note that symmetrical beads can just be group into 1

Goal: This dictionary can describe ALL comb polymers with only Carbon and backbone and sidechains

Task 3: Given the original inputs, these beads can be used to form the polymer and output a file that is like the file from task 1 but instead of atoms it will be these beads instead.

For example: let’s say we have CCCCCC as our smiles input and one of our bead is “CCC+” with the value “SE”, the text file could be:
SE – SE

This is trivial to look at and understand as CCC+ can be merged with +CCC which is symmetrical and form CCCCCC

To do this, we need to think of an algorithm that do the following:

1. Match beads with triple bonds first, check for continuation to find the correct bead
2. Match beads with sidechain length 1 and 2 next
3. For the side chains that is from the edges of the backbone, just use E and C beads \
4. Match beads with sidechain of any size next using the sidechain nodes
5. Use algorithms like trie or dp to then match everything else with the following priority:
    a. Use big beads when possible
    b. Use small beads and tiny beads the amount of bead left consecutively is 2(tiny), 3(small), 5(tiny + small), or 6(small+small)
    c. At the end, use the O beads for the rest of the 1 atoms.
6. Output the results that: use the minimum number of beads,

More note: the amount of connection is crucial, as noted + and ++. Without + or ++, that bead cannot continue. Let's give it a try

---

Task: Create an algorithm to reduce information while able to represent

Given:

1. A dictionary with 122 different key-values pairs representing all "beads"
2. A string input that is a Smiles code of a comb polymer consisting of a backbone and each node of the backbone can have at most 1 sidechain and sidechains cannot have extra sidechains.

Sample Input:

a. CCC(CCC)CCC
b. CC(C)CC(CCC)CCC

Sample output:

a.