PROGRAMMING EXERCISE

Rozana Alam, rozanamail12@gmail.com

May 2021

TASK

The exercise is to implement an algorithm that can generate a list of 96 words. Every word should be created using six (6) characters in total, all of which are to be picked from a four-letter alphabet consisting of A, C, G and T. Each word in the list must differ from every other word in the list in at least three positions.

1 Algorithm steps in C++ Programming Language

- Step 1: Generate all possible words with 'ACGT' by generateAll(6)
- Step 2: Go To generateValidList(lst[i])
- Step 3:Go To maintainsConstraint(validList, lst[i])
- Step 4: Go To difference(validList[i], candidate) calculate
- Step 5: Next validList.push_back(lst[i])
- Step 6: Print first 96 words finalLst.push_back(validList[i])

Problem solution Code with C++ Programming Language

```
// declear header file as below which include all
   standard libraries
#include<bits/stdc++.h>
```

```
using namespace std; //using standard namespace


// below variables are gloabal variable
char validChar[] = "ACGT";
vector<string> lst;
vector<string> finalLst;
int targetCount = 96;

void generateAll(int len){

/* generate a specific word with a
fix length (len) variable and store it
in a vector(lst) */

    string str;

    for(int bit = 0; bit < (1 << 12); bit++){
    /*(1: first operand, 12: second operand),left
   shifts the bits of the first operand, the second
   operand decides the number of places to shift. 2^12
    which is 2 raised to power 12, times runs is 4096
   */

        str = "";
        for(int i = 0; i < len; i++){
            int cur = (bit >> (2 * i)) & 3;

            /*(cur) variable stores the result of
            right shift operation on (bit) variable
            which is the position of random charactor
            in validChar vector*/

            str += validChar[cur];

            /* get the character
            using the (cur) value as the position
            of the character that you add to
```

```
            (str) from (validChar) */
        }
        lst.push_back(str);
    }
}

int difference(const string &a, const string &b){
/* calculate positional differences
between two string */
    int diff = 0;
    for(int i = 0; i < a.size(); i++){
        if(a[i] != b[i]) diff++;
    }
    return diff; }

/*maintainsConstraint function compare
candidate word with validList to see that
this candidate word has atleast
3 positional difference */

int maintainsConstraint(const vector<string> &
   validList,
const string &candidate){
    for(int i = 0; i < validList.size(); i++){
        if(difference(validList[i], candidate) < 3){

        /* compare candidate word with validList to
        see that this candidate word has
        at least 3 positional difference */


            return 0;

            /* if it has less than
            3 positional difference,
            this returns 0 else return 1*/
        }
    }
```

```
    return 1;
}

int generateValidList(string start){
    vector<string> validList;

    validList.push_back(start);

   int count =1;
   for(int i = 0; (i < lst.size()) && (count <
   targetCount); i++){

   /* i should be less than the element
   in vector(lst) => (i < lst.size()) */


        if(maintainsConstraint(validList, lst[i])){

        /*maintainsConstraint function
        must return 0 => less than 3 positional
   difference
        or 1=> 3 or more positional difference
        */

            validList.push_back(lst[i]);
            count++;
        }
    }

    if(validList.size() == targetCount){
        for(int i = 0 ;i < targetCount; i++){
            finalLst.push_back(validList[i]);
         }
    }

    return count;

}
```

```
int main(){

    generateAll(6);

    //for(int i=0;i< lst.size();i++)
    {
      /*  int count = generateValidList(lst[i]);  */

      /*generateValidList
      function =>(goto) maintainsConstraint
      function =>(goto) difference function */

        int count = generateValidList("AACTCA");
        /* starting form 'AACTCA' sring gives
        96 words perfectly */

        if(count == targetCount){

        /*count must be equal to 96
        since targetCount is 96 which is
        the number of words requested */
            for(int j=0 ; j < targetCount ; j++){
     cout << "String " << j+1 << " : " << finalLst[j
   ]<< endl;
            }
            cout << endl;
//    break;

/** if we use
int count = generateValidList(lst[i]);
then after 96 words it must need to break
and then beak should be activated */
        }
     }
    return 0;
}
```

Output results of the problem:

```
String 1 : AACTCA
String 2 : AAAAAA
String 3 : CCCAAA
String 4 : GGGAAA
String 5 : TTTAAA
String 6 : GCACAA
String 7 : TACCAA
String 8 : ATGCAA
String 9 : CGTCAA
String 10 : TGAGAA
String 11 : GTCGAA
String 12 : CAGGAA
String 13 : ACTGAA
String 14 : CTATAA
String 15 : TCGTAA
String 16 : GATTAA
String 17 : TCAACA
String 18 : CTGACA
String 19 : AGTACA
String 20 : CAACCA
String 21 : GGCCCA
String 22 : ATAGCA
String 23 : GCGGCA
String 24 : TATGCA
String 25 : CCTTCA
String 26 : CGAAGA
String 27 : GACAGA
String 28 : ACGAGA
String 29 : TTACGA
String 30 : AATCGA
String 31 : TCCGGA
String 32 : GGTGGA
```

```
String 33 : GTGTGA
String 34 : GTAATA
String 35 : TGCATA
String 36 : CATATA
String 37 : AGACTA
String 38 : CTCCTA
String 39 : GAGCTA
String 40 : TCTCTA
String 41 : CCAGTA
String 42 : TTGGTA
String 43 : TAATTA
String 44 : GCCTTA
String 45 : CGGTTA
String 46 : ATTTTA
String 47 : AGCAAC
String 48 : TAGAAC
String 49 : GCTAAC
String 50 : CCGCAC
String 51 : GAAGAC
String 52 : CTTGAC
String 53 : ACATAC
String 54 : CACTAC
String 55 : TGTTAC
String 56 : GGAACC
String 57 : TTCACC
String 58 : ACCCCC
String 59 : TGGCCC
String 60 : GATCCC
String 61 : CGCGCC
String 62 : AAGGCC
String 63 : ATAAGC
String 64 : GTCCGC
String 65 : AGGTGC
String 66 : TACGTC
String 67 : GGGGTC
String 68 : GTTCAG
String 69 : AACGAG
String 70 : GGATAG
```

```
String 71 : TTCTAG
String 72 : CACACG
String 73 : GAGTCG
String 74 : TAAAGG
String 75 : CCTAGG
String 76 : ACACGG
String 77 : CGCCGG
String 78 : CTAGGG
String 79 : TGGGGG
String 80 : ACCATG
String 81 : GGTATG
String 82 : AGGGAT
String 83 : GCCACT
String 84 : GTACCT
String 85 : AGATCT
String 86 : CTCTCT
String 87 : CAGAGT
String 88 : TGTAGT
String 89 : GCGCGT
String 90 : CTTCGT
String 91 : AAAGGT
String 92 : CCATGT
String 93 : TACTGT
String 94 : TCGATT
String 95 : AACCTT
String 96 : GATGTT
```

1.1 How to run the code with C++

1. To run the code please open the generate96.cpp file and select all the code by ctl+A and copy by ctl+ C then go the side C++ Shell (http://cpp.sh/) and after cleaning the C++shell window paste the code that you already copied. finally you need run by clicking on Run button.

2. Another option is click on the link http://cpp.sh/3mulz and you can

see the code and you can run it.

2 Solution with the Python Programming language

- **Step 1:** Generate all words
- **Step 2:** numberWord = 96, newWordList = finalList Each element from (wordListNew) is checked against all element of (newWordList) to see if all the check produce at least 3 positional difference.
- **Step 3:** This returns list of true which means that compare the one words against the list of words that has difference 3 or more than 3
- **Step 4:** The positional difference is greater than 3 or 3 then it returns true else it returns false
- **Step 6:** Print the final list of distinct character

```
class WordGenerator:

    def gen(self, wordList=None, numberWord=None):
        '''
          generate all combinations of ['A', 'C', 'G',
    'T']*5
        '''
        wordListNew = map(lambda x: "".join(x),
   wordList)

        newWordList = list()
        list(map(lambda x: WordGenerator.check(x,
   newWordList, numberWord), list(wordListNew)))
        return newWordList

    @staticmethod
    def check(word=None, newWordList=None, numberWord=
   None):
```

```
        '''
          check all position difference that are only
   3 or greater than 3
        '''
        result = [WordGenerator.diff(x, word) for x in
    newWordList]
        if False not in result and len(newWordList) <
   numberWord:
            newWordList.append(word)

    @staticmethod
    def diff(newWord=None, word=None):
        if len([i for i, j in zip(list(newWord), word)
    if i != j]) < 3:
            return False
        return True
```

Listing 1: 96 words Generator with python

```
from itertools import combinations

if __name__ == '__main__':

    gen = WordGenerator()
    result = gen.gen(list(combinations(['A', 'C', 'G',
    'T'] * 5, 6)), 96)
    for i, newWord in enumerate(result):
        print(i, newWord)
```

Listing 2: Print list with python Code

2.1 How to run the code with Python?:

The easiest and faster way to run this code is open the Python_code.ipynb file in Google colab and use the run button to run it.

3 Github link

https://github.com/Rozana-github/WordGenerator.git

Thank you