Project 1

Ning Xu

Oct. 2022

1 Problem introduction

1.1 Task A

Task A requires us to define a function **[p,q] = AppCat(N)** that satisfying three requirements. The first requirement is that the function should figure out a pair of positive integers p,q such that the sum of p,q is less than or equal to a given positive integer N(which is 2022 in task A). The second requirement is that it can provide the smallest absolute value of the difference between $\frac{p}{q}$ and $\mathcal{C}$, i.e. Min$abs(\frac{p}{q}-\mathcal{C})$. The third requirement is that if more than one pair of positive integers are determined, the pair with the smallest sum will be chosen, i.e. If there exists some positive integer x,y different from p,q, such that have the same absolute difference, i.e. $|\frac{x}{y}-C| = |\frac{p}{q}-C|$, and the sum of x and y are larger than p+q, i.e. x+y>p+q, then p and q should be chosen. Note that $\mathcal{C}$ is called the Catalan constant which is defined as $\mathcal{C} = \sum_{n=0}^{\infty} \frac{(-1)^k}{(2k+1)^2} = \frac{1}{1^2} + \frac{(-1)}{3^2} + \cdots$.

1.2 Task B

First, note that the Cubic taxicab number is defined as a positive integer that can be expressed as the sum of two positive cubic numbers in two or more methods.
In this task, a smallest cubic taxicab number M larger than given positive integer N along with its two pairs of corresponding positive integers a,b and c,d, such that $a^3 + b^3 = c^3 + d^3 = M$ are needed to be determined by a function **[a,b,c,d,M] = CubicTaxicabNum (N)** .

1.3 Task C

The smallest integer can be expressed as the sum of two positive cubic numbers in n different methods and is defined as T_n. In this task, T_3 and its three pairs of corresponding positive numbers a,b , c,d and e,f, such that $a^3 + b^3 = c^3 + d^3 = e^3 + f^3 = T3$ are needed to be determined by the function **[a,b,c,d,e,f,T3]=findT3(N)**.

2 Problem Solution

2.1 Task A

2.1.1 General Solution

The idea is straightforward: iterate through all possible values of p and q, using variables to record the historical minimum and its corresponding p and q values. Use another variable to record the current minimum under the current p and q fetches. The historical minimum is compared with the current minimum, and the historical minimum is replaced with the current minimum and the variables recording p and q are updated accordingly when either the current minimum is strictly less than the historical minimum or the current minimum is the same as the current minimum and the p and q corresponding to the historical minimum are greater than the p and q corresponding to the current minimum. When all possible values of p and q have been traversed, the p and q corresponding to the historical minimum is the result we seek.

2.1.2 Code Development Concept

A flowchart would be introduced to illustrate the concepts of coding development.

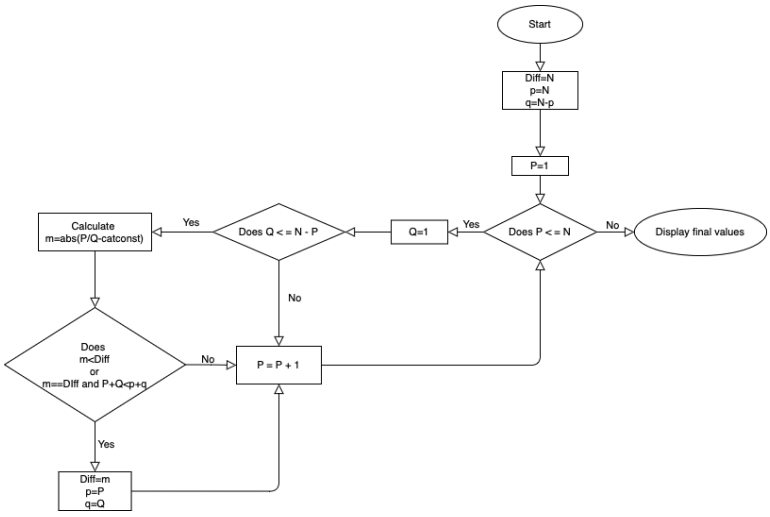

Figure 1: The flow chart of Task A

The logic and explanation of the flowchart will be illustrated in the following parts.
First, variables used in this task and their purpose will be listed.
· The constant **catconst** has been defined in the last section. In this part, it takes the value of $\approx 0.915965594177219$, which is in fifteen decimal points.
· A variable **Diff** has been used to store the smallest absolute difference between $\frac{p}{q}$ and the catconst that ever occurred.
· Variables **p** and **q** would record the value p and value q corresponding to the variable **Diff**, i.e. variables **p** and **q** would record the value p and value q which makes the variable **Diff** storing the smallest absolute value between $\frac{p}{q}$ and catconst.
· The variable **P** takes values from 1 to N by the construction of a **for loop** regarding **P**.

· The variable **Q** takes value from 1 to N-P by constructing a **for loop** inside the **for loop** regarding P, which means the largest value of Q is dependent on the current number of P. The reason for setting the upper bound of variable Q as N-P is to prevent the sum of an element from the variable P and elements from the variable Q is larger than N, i.e. $\forall x \in P, \forall y \in Q, x + y \leq N$.
· The variable m has been defined as $m = abs(\frac{P}{Q} - catconst)$. The value of this variable m would change as **for loop** regarding Q and P operating.
Then the operation of this flowchart will be explained.
STEP 1 Assigning values to variables. The reason for assigning N to variable Diff is to ensure it has the largest value at the very beginning, which implies it would be iterated as the function operating. The reason for assigning value to variables p and q is to ensure they can be used and assign values in operation.
STEP 2 A **for loop** concerning P has been entered and this following with another **for loop** regarding Q which is nested in the **for loop** of P, which means the **for loop** of P will enter the next round once the **for loop** of Q is over with the current value P.
STEP 3 A variable m has been assigned a value regarding P and Q in this round and would be compared with the variable Diff by an **if statement**.
STEP 4 The variable Diff would take the value of m as well as p and q would take the value of P and Q of this round if one of the following situations has been satisfied. The first is m is less than Diff, and the other one is that m is equal to Diff while the P and Q from this round have a smaller value than variable p and q, i.e.P+Q<p+q. After that, this round of **for loop** of Q is finished. If none of the above situations has been satisfied, this round of **for loop** of Q will be finished immediately. The next round of **for loop** will be started. Note that the next round of **for loop** regards which variable is dependent on whether the **for loop** of Q is finished at the current number of P.
STEP 5 Once the **for loop** of P is finished, the final answer to this task will be obtained.
Based on the flowchart, we are able to implement it in Matlab.

2.2 Task B

2.2.1 General Solution

Create a loop that takes a variable named **Possible Number** and makes it equal to N. Develop an array of all values less than the third root of the positive integer N. Find if there are two integers (possibly equal) from the array whose cubic sum is the same as the **Possible Number**. Stop when there are four numbers that meet this condition and output the result, otherwise make N=N+1 and continue the process until there is a result to output.

2.2.2 Code development concept

A flowchart would be introduced to illustrate the concepts of coding development.

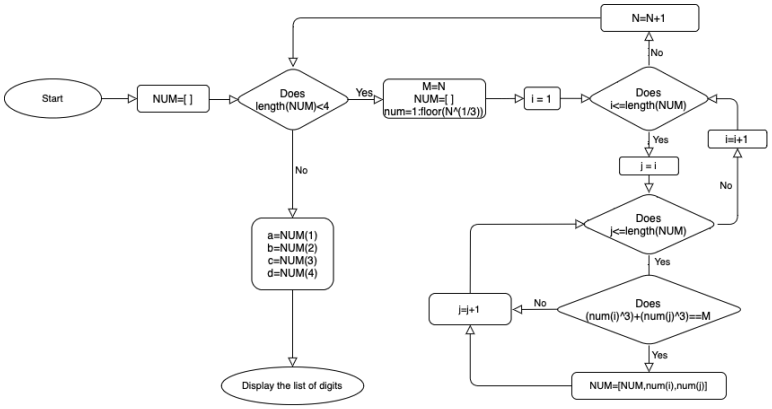

Figure 2: The flow chart of Task B

First, some variables will be introduced as before. · The variable M will store the **Possible Number** mentioned before.

· An array num will store all values less than the third root of the positive integer N.

· Two integers (possibly equal) from the array whose cubic sum is the same as the **Possible Number** would be stored into a variable called NUM. Hence, the number of elements in NUM would always be an even number.

Then the operation of this flowchart will be explained.

STEP 1 The variable NUM is created to ensure it can be used in the next moves.

STEP 2 A **While loop** is developed, and this loop would keep working until the number of elements in NUM is larger than or equal to four. Note that the Possible number M would have a different value each round as N increase, the array NUM is set to be empty to ensure it only store values regarding the current Possible Number M and the array num is also updated since N changes in each round.

STEP 3 A **for loop** of j is nested in a **for loop** of i, which ensures two values are taken from the array num each time and check whether their cubic sum is equal to the Possible Number. If the equality holds, array NUM would store the i-th and j-th elements in num, then process to the next step. If the equality fails, the function moves into the next step directly. Note that j would have the same value as i at the beginning of the **for loop** of j because of avoiding repeating calculation.

STEP 4 If the **for loop** of j is finished, the **for loop** of i will move to the next round and repeat the process mentioned before. If the **for loop** of i is finished, the integer N would equal N+1 and go back to the **while loop** to check whether the statement holds.

STEP 5 If the statement of **while loop** holds, the process mentioned before would restart with a new N. If the statement fails, i.e. the number of elements in array NUM is larger or equal to four, we would jump out of the **while loop** and assign values from the array NUM to a,b,c,d. After that, display the answer we get. Based on the flowchart, we are able to implement it in Matlab.

2.3 Task C

2.3.1 General Solution

Task C requires us to determine that the smallest integer can be expressed as the sum of two positive cubic numbers in 3 different ways. The basic logic and the code implementation of task B can be directly used in task C. With just a couple of slight modifications, the required answer can be obtained. In task B, the loop would carry to the end once there are four or more numbers that meet the condition, in task C, the loop will bring to the end once there are six or more numbers that meet the condition.

2.3.2 Code development concept

A flowchart would be introduced to illustrate the concepts of coding development.

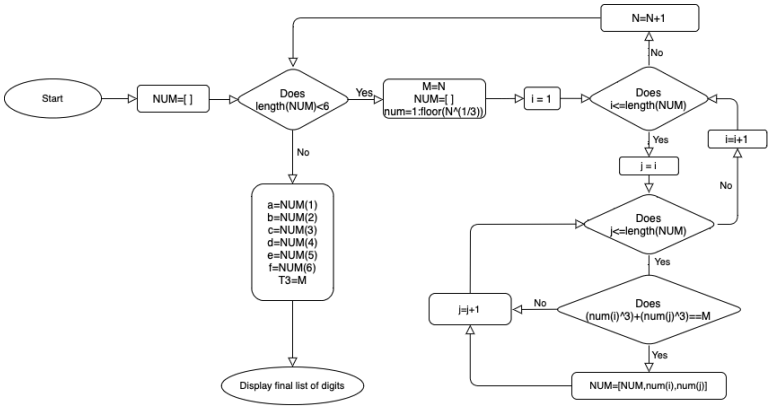

Figure 3: The flow chart of Task C

Since task B and task C are similar, just a little bit of modification is required in **STEP 2** and **STEP 5**, the rest steps would just be the same. Then follows the modified version of **STEP 2** and **STEP 5**. **STEP 2** A **While loop** is developed, and this loop would keep working until the number of elements in NUM is larger than or equal to six. Note that the Possible number M would have a different value each round as N increase, the array NUM is set to be empty to ensure it only store values regarding to the current Possible Number M and the array num is also updated since N changes in each round.

STEP 5 If the statement of **while loop** holds, the process mentioned before would restart with a new N. If the statement fails, i.e. the number of elements in array NUM is larger or equal to four, we would jump out of the **while loop** and assign values from the array NUM to a,b,c,d,e,f as well as let T3 equal to M. After that, display the answer we get. Based on the flowchart, we are able to implement it in Matlab.

3 Code Implementation

3.1 Task A

```
function [p,q] = AppCat(N)
%APPCat Approximates the catalan constant by the rational number p/q
% APPCat Approximates the catalan constant by the rational number
% p /q , among all pairs of positive integers (p , q ) such that p + q <= N
%The difference between p/q and c, we need to determine the smallest of it.
[Diff,p,catconst]=deal(N,N,0.915965594177219);
q=N-p;
for P=1:N
    for Q=1:N-P  %Make sure p+q<=N
        m=abs(P/Q-catconst);
        if m<Diff || (m==Diff && P+Q<p+q) %Diff stores the smallest difference that
            ever happened.
            [Diff,p,q]=deal(m,P,Q);%p and q store the value corresponding to Diff.
        end
    end
end
```

```
>> [p,q] = AppCat(2022)
p =
   109
q =
   119
```

3.2 Task B

```
function [a,b,c,d,M] = CubicTaxicabNum ( N )
%CUBICTAXICABNUM returns the smallest cubic taxicab number M
% CUBICTAXICABNUM returns the smallest cubic taxicab number
% M=a^3+b^3=c^3+d^3 greater than or equal to N
NUM=[];%Store the  given M
while length(NUM)<4 %The number does not have 2 pairs of numbers.
    M=N;
    NUM=[];%Delete the previous elements.
    num=1:floor(N^(1/3));
    for i=1:length(num)
        for j=i:length(num)
            if (num(i))^3+(num(j))^3==N
                NUM=[NUM,num(i),num(j)];
            end
        end
    end
```

```
    N=N+1;
end
[a,b,c,d,M]=deal(NUM(1),NUM(2),NUM(3),NUM(4),M);
end
```

```
>> [a,b,c,d,M] = CubicTaxicabNum ( 36031 )
a =
     2
b =
    34
c =
    15
d =
    33
M =
       39312
```

3.3 Task C

```
function [a,b,c,d,e,f,T3] = findT3 ( N )
%CUBICTAXICABNUM returns the smallest cubic taxicab number M
% CUBICTAXICABNUM returns the smallest cubic taxicab number
% M=a^3+b^3=c^3+d^3 greater than or equal to N
SUM=[];
while length(SUM)<6
    T3=N;
    SUM=[];
    num=1:floor(N^(1/3));
    for i=1:length(num)
        for j=1:length(num)
            if (num(i))^3+(num(j))^3==N && i<=j
                SUM=[SUM,num(i),num(j)];
            end
        end
    end
    N=N+1;
end
[a,b,c,d,e,f,T3]=deal(SUM(1),SUM(2),SUM(3),SUM(4),SUM(5),SUM(6),T3);
end
```

```
>> [a,b,c,d,e,f,T3] = findT3 ( 87000000 )
```

```
a =
   167
b =
   436
c =
   228
d =
   423
e =
   255
f =
   414
T3 =
    87539319
```