CHUYÊN ĐỀ HỘI THẢO CÁC TRƯỜNG CHUYÊN VÙNG DHBB

Mục lục:

MẢNG CỘNG DỒN VÀ MỘT SỐ ỨNG DỤNG

I. Mở đầu về mảng cộng dồn

Bài toán tìm tổng dãy con liên tiếp là một bài toán cơ bản có rất nhiều biến thể trong các bài toán tin mức độ trung bình đến tương đối khó, phù hợp với đối tượng học sinh tiền đội tuyển lớp 10 chuyên. Đây là chủ đề vận dụng kiến thức nâng cao sau khi học mảng một chiều, mảng hai chiều, quy hoạch động cơ bản.

Phát biểu bài toán cơ sở: cho mảng A gồm N phần tử. Gọi mảng F là mảng cộng dồn – là một mảng có kích thước như A và giá trị được tính theo công thức:
$F[i]=A[0]+A[1]+\ldots+A[i]=F[i-1]+A[i]$

Để thực hiện việc tính mảng F, ta sử dụng đoạn code đơn giản sau:

```
f[0]=0;
    for (int i=1; i<=n; i++)
        f[i]=f[i-1]+a[i];
```

II. Bài toán cơ bản

Trước khi tìm hiểu về các ứng dụng của mảng cộng dồn (prefix sum), ta cần thiết phải giới thiệu qua một số bài toán cơ bản, điển hình về kĩ thuật này. Dưới đây là ba dạng thức phổ biến.

1. Tìm tổng một đoạn con liên tiếp

Cho mảng A gồm N phần tử. Gọi mảng F là mảng cộng dồn – là một mảng có kích thước như A và giá trị được tính theo công thức:
$F[i]=A[0]+A[1]+\ldots+A[i]=F[i-1]+A[i]$

Để thực hiện việc tính mảng F, ta sử dụng đoạn code đơn giản sau:

```
f[0]=0;
    for (int i=1; i<=n; i++)
        f[i]=f[i-1]+a[i];
```

(GSS – SPOJ https://codeforces.com/group/FLVn1Sc504/contest/274493/problem/B)

2. Tăng giá trị từng đoạn

Bài toán: Cho mảng A kích thước N phần tử. Ban đầu A[i]=0; Thực hiện M truy vấn: Tăng giá trị đoạn từ A[a] đến A[b] một giá trị Val; Tìm giá trị lớn nhất trong mảng sau M truy vấn?

Input:

- Dòng đầu ghi số nguyên N ($1 \le N \le 10^6$): số lượng phần tử; M $(1 \le M \le 10^6)$ số lượng truy vấn, Val: giá trị tăng $(1 \le Val \le 10^9)$
- M dòng tiếp theo thể hiện M truy vấn: mỗi dòng gồm hai số i,j cách nhau bởi dấu cách $(1 \le i \le j \le N)$

Output:

- Ghi một số nguyên duy nhất là giá trị lớn nhất mảng sau khi thực hiện lần lượt M truy vấn

Ví dụ:

Input	Output
5 3 100 2 4 1 3 1 2	300

Giải thích:

- Sau lượt 1: A={0;100;100;100;0}
- Sau lượt 2: A={100;200;200;100;0}
- Sau lượt 3: A={200;300;200;100;0}
- Vậy giá trị lớn nhất là 300.

Hướng dẫn:

Sol1 (Trâu): mỗi cặp [a,b] ta tăng Val cho các phần tử từ A[a] đến A[b];

Sol2 (dùng mảng cộng dồn F):

- Mỗi cặp [a,b] ta cộng vào A[a]+=Val; và A[b+1]-=Val

- Sau M truy vấn, ta tính lại mảng cộng dồn F → Tìm phần tử lớn nhất trên F chính là kết quả cần tìm.

Cụ thể trên ví dụ như sau:

- Sau lượt 1; mảng A = {0,100,0,0,-100} → mảng cộng dồn F={0,100,100,100,0}
- Sau lượt 2: A={100,100,0,-100,-100} và F={100,200,200,100,0}
- Sau lượt 3: A = {100,100,0,-100,-100} và F={200,300,200,100,0}

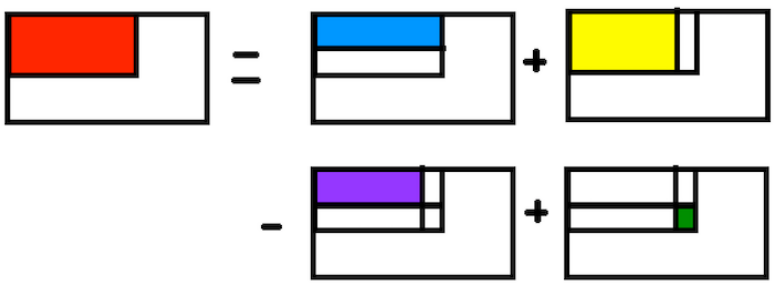
Như vậy phần tử có giá trị lớn nhất sẽ là 300.

(QMAX https://codeforces.com/group/FLVn1Sc504/contest/274521/problem/G)

3. Mảng cộng dồn trên mảng 2 chiều

Trên mảng hai chiều $A(i,j)$ ta đặt $F(i,j)$ là tổng các ô trong hình chữ nhật có đỉnh trên bên trái là $(1,1)$ và đỉnh dưới bên phải là (i,j). Khi đó ta có:

$$F(i,j)=F(i-1,j)+F(i,j-1)-F(i-1,j-1)+A(i,j)$$



(Nguồn: VNOI wiki)

(UVA 108 – Maximum Sum https://onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=44)

III. Một số ứng dụng của mảng cộng dồn

Bài 1. Equilibrium.* - Chỉ số cân bằng của dãy

Cho dãy A gồm N phần tử. Một chỉ số i thuộc [1..N] được gọi là chỉ số cân bằng nếu tại i chia dãy thành hai phần sao cho Tổng các phần tử có chỉ số bé hơn i bằng tổng các phần tử có chỉ số lớn hơn i; Tức là:

$$A[1]+A[2]+\ldots+A[i-1]=A[i+1]+A[i+2]+\ldots+A[N]$$

Input:

- Dòng đầu ghi số N (1<=N<=10^6) ,
- Dòng tiếp theo ghi N số nguyên cách nhau bởi dấu cách

Output:

- Gồm một số nguyên ghi chỉ số cân bằng bé nhất tìm được. Nếu không tồn tại chỉ số cân bằng, ghi -1;

Ví dụ:

Input	Output
7 -7 1 5 2 -4 3 0	4

Giải thích:

4 là chỉ số cân bằng vì A[1]+A[2]+A[3]=-7+1+5=-1=A[5]+A[6]+A[7]=-4+3+0=-1;
Ngoài ra ta có 7 là chỉ số cân bằng vì sum(A[1]..A[6])=0

Hướng dẫn:

Sol1 (trâu): làm như yêu cầu đề bài: kiểm tra với mỗi vị trí i, tính tổng LEFT=A[1]+…+A[i] và tổng RIGHT=A[i+1]+…+A[N], nếu LEFT=RIGHT thì kết quả là i; Độ phức tạp $O(n^2)$

Sol2:
```
sum=sum(A[1]..A[N]); LeftSum=0;
For (int i=1; i<=N; i++){
     RightSum=sum-A[i]; LeftSum=LeftSum+A[i];
     If (RightSum==LeftSum) return i;
}
Return -1;
```

Với cách này độ phức tạp là $O(n)$

Bài 2. ZEROSUM.* - Tìm dãy con có tổng bằng 0

Cho dãy N phần tử số nguyên; Xác định xem dãy đã cho có tồn tại dãy con mà có tổng các phần tử thuộc dãy con bằng 0? Nếu có ghi YES, ngược lại ghi NO.

Input:

- Dòng đầu ghi số N (1<=N<=10^6),

- Dòng tiếp theo ghi N số nguyên, các số cách nhau bởi dấu cách |A[i]|<=10^9

Output:

- Ghi YES nếu tồn tại dãy con có tổng bằng 0; ngược lại ghi NO

Ví dụ:

Input	Output	Giải thích
5 4 2 -3 1 6	YES	Dãy A[2..4] có sum=0
5 -3 2 3 1 6	NO	

Hướng dẫn:

- Sol1: Xét tất cả các đoạn con [i,j], tính tổng và kiểm tra xem có thỏa mãn điều kiện hay không? Độ phức tạp $O(n^3)$
- Cải tiến: Giả sử đoạn [i,j] có tổng bằng 0; khi đó f[j]=f[i-1] → tính mảng cộng dồn f, kiểm tra xem có cặp (i,j) nào thỏa mãn điều kiện không? Độ phức tạp $O(n^2)$

Code mẫu:

```
bool subArrayExists(int arr[], int n)
{
    unordered_set<int> sumSet;
  //Duyệt qua mọi phần tử của mảng, tính và lưu trữ mảng cộng dồn
    int sum = 0;
    for (int i = 0 ; i < n ; i++)
    {
        sum += arr[i];
        //Nếu sum=0 hoặc giá trị này đã tồn tại trong set nghĩa là tồn tại dãy
có tổng bằng 0
        if (sum == 0 || sumSet.find(sum) != sumSet.end())
            return true;

        sumSet.insert(sum);
    }
    return false;
}
```

Bài 3. MaxiSubArr.* - Kích thước mảng con lớn nhất

Cho dãy A gồm N phần tử số nguyên dương và một số nguyên dương K; Tìm mảng con có kích thước lớn nhất sao cho tất cả các mảng con có kích thước bé hơn nó đều có tổng các phần tử nhỏ hơn K.

Input:

- Dòng thứ nhất chứa số nguyên N (N<=10^6) và số nguyên K (K<=10^12)
- Dòng thứ hai ghi N số nguyên thuộc dãy, các số cách nhau bởi dấu cách

Output:

- Ghi kích thước mảng con lớn nhất cần tìm

Ví dụ:

Input	Output	Giải thích
4 8 1 2 3 4	2	Tổng dãy con có: • Độ dài 1: 1,2,3,4 • Độ dài 2: 3,5,7 • Độ dài 3: 6,9 • Độ dài 4: 10
4 8 1 2 10 4	-1	Không có dãy con thỏa mãn yêu cầu

Hướng dẫn: Độ phức tạp $O(nlogn)$

Đương nhiên, mảng con cần tìm sẽ có kích thước trong [1,n]

Vì mọi phần tử đều >0 nên ta thấy, giá trị các phần tử thuộc mảng cộng dồn sẽ tăng dần.

Tức là,

nếu $a[i]+a[i+1]+...+a[j-1]+a[j] \le K$

thì $a[i]+a[i+1]+...+a[j-1] \le K$ với a[j]>0

Ta tìm kiếm nhị phân trong đoạn 1,n để tìm dãy con có kích thước lớn nhất sao cho mọi dãy con đó có kích thước đó có tổng $\le K$

Code mẫu:

```
#include<bits/stdc++.h>
using namespace std;

int bsearch(int prefixsum[], int n, int k)
{
    int ans = -1;    // Initialize result
```

```
    // Tìm kiếm nhị phân: tìm kích thướng mảng con lớn nhất
    int left = 1, right = n;
    while (left <= right)
    {
        int mid = (left + right)/2;

        int i;
        for (i = mid; i <= n; i++)
        {
            if (prefixsum[i] - prefixsum[i - mid] > k)
                break;
        }

        if (i == n+1)
        {
            left = mid + 1;
            ans = mid;
        }
        else
            right = mid -1;
    }

    return ans;
}
int maxSize(int arr[], int n, int k)
{
    // Khởi tạo mảng cộng dồn bằng 0
    int prefixsum[n+1];
    memset(prefixsum, 0, sizeof(prefixsum));

    // Tìm mảng cộng dồn của mảng arr ban đầu
    for (int i = 0; i < n; i++)
        prefixsum[i+1] = prefixsum[i] + arr[i];

    return bsearch(prefixsum, n, k);
}

const int maxN=100000;
int n, a[maxN],k;

int main()
{
    freopen("MaxiSubArr.inp","r",stdin);
    freopen("MaxiSubArr.out","w",stdout);
    cin >> n >> k;
    for (int i=0; i<n; i++) cin >> a[i];
    cout << maxSize(a, n, k) << endl;
    return 0;
}
```

Bài 4. NUMBERS.* - Dãy chẵn lẻ cân bằng

Cho dãy A gồm N số nguyên được đánh số từ 0..N-1. Hãy tìm một chỉ số i thỏa mãn điều kiện: dãy trước A[i] và dãy sau A[i] thỏa mãn điều kiện: số lượng số chẵn/lẻ trong dãy trước bằng số lượng số chẵn/lẻ trong dãy sau. Nếu tồn tại nhiều chỉ số thỏa mãn điều kiện, in chỉ số có giá trị bé nhất. Nếu không có chỉ số nào thỏa mãn, in -1.

Input:

- Dòng đầu ghi số nguyên N (1<=N<=10^6)
- Dòng tiếp theo ghi N số nguyên, các số cách nhau bởi dấu cách

Output:

- Ghi chỉ số cần tìm hoặc -1 nếu không tồn tại chỉ số phù hợp.

Ví dụ:

NUMBERS.INP	NUMBERS.OUT
6 4 3 2 1 1 2 4	2
7 1 2 4 5 8 3 12	3

Hướng dẫn:

Cách 1 (trâu): Độ phức tạp $O(n^2)$

Với mỗi vị trí i, đếm số lượng số chẵn, số lẻ ở hai phía TRÁI, PHẢI và so sánh

Cách 2: Độ phức tạp $O(n)$

Tạo hai vecto Left và Right là hai mảng cộng dồn lưu:

Left[i] lưu tần suất số lẻ (Left[i].first) và số chẵn (Left[i].second) của dãy bên trái i

Right[i] lưu tần suất số lẻ (Right[i].first) và chẵn (Right[i].second) của dãy bên phải i

Nếu Left[i].fisrt=Right[i].first **hoặc** Left[i].second=Right[i].second thì i chính là vị trí cần tìm.

Code mẫu:

```
#include <bits/stdc++.h>
using namespace std;

int Find_Index(int n, int arr[]) {
```

```
  int odd = 0, even = 0;
  vector<pair<int, int>> v_left, v_right;
  v_left.push_back(make_pair(odd, even));
  for (int i = 0; i < n - 1; i++) {
    if (arr[i] % 2 == 0)
      even++;
    else
      odd++;

    v_left.push_back(make_pair(odd, even));
  }
  odd = 0, even = 0;
  v_right.push_back(make_pair(odd, even));
  for (int i = n - 1; i > 0; i--) {
    if (arr[i] % 2 == 0)
      even++;
    else
      odd++;

    v_right.push_back(make_pair(odd, even));
  }
  reverse(v_right.begin(), v_right.end());
  for (int i = 0; i < v_left.size(); i++) {
    if (v_left[i].first == v_right[i].first ||
        v_left[i].second == v_right[i].second)
      return i;
  }
  return -1;
}

const int maxN=1000005;
int n, arr[maxN];
int main() {
    freopen("Numbers.inp","r",stdin);
    freopen("Numbers.out","w",stdout);
    cin >> n;
    for (int i=0; i<n; i++) cin >> arr[i];

    int index = Find_Index(n, arr);
    ((index == -1) ? cout << "-1" : cout << index);
    return 0;
}
```

Bài 5. SPLITARR.* - Tách dãy

Cho dãy A gồm N phần tử được đánh số từ 0..N-1. Tìm cách tách A thành ba phần sao cho tổng các phần tử trong ba phần là bằng nhau?

Input:

- Dòng đầu ghi số N (N<=10^6)
- Dòng tiếp theo ghi N số nguyên, các số cách nhau bởi dấu cách (Ai<=10^9)

Output:

- Kết quả gồm hai chỉ số i,j thỏa mãn 0<=i<j<=N sao cho i,j chia dãy thành 3 mảng con có tổng bằng nhau. Nếu không tồn tại cách chia, in ra -1

Ví dụ:

SPLITARR.INP	SPLITARR.INP
5 1 3 4 0 4	1 2
3 2 3 4	-1

Hướng dẫn: độ phức tạp O(n)

Gọi S là tổng các phần tử của dãy. Nếu S không chia hết cho 3 thì dĩ nhiên dãy ban đầu không thể tách thành 3 dãy bằng nhau.

Nếu tồn tại 3 dãy con có tổng bằng nhau, tức là mỗi dãy có tổng là S/3. Tức là tồn tại hai chỉ số i, j sao cho a[0]+...+a[i-1]=a[i]+...+a[j-1]=a[j]+...+a[n-1]=S/3.

Mà a[0]+...+a[i-1]=F[i] (F là mảng cộng dồn)

a[i]+...+a[j-1]=F[j]-F[i];

→ F[i]=F[j]-F[i]=S/3

→ 2*F[i]=F[j]

Bài toán trở thành: tìm hai chỉ số i, j sao cho F[i]=S/3 và F[j]=2*(S/3)

Code mẫu:

```
#include <bits/stdc++.h>
using namespace std;
int findSplit(int arr[], int n)
{
    int i;
    int preSum[n];
    int index[n];

    int count = 0;
    int k = 0;
    preSum[0] = arr[0];
```

```
    for (i = 1; i < n; i++) {
        preSum[i] = preSum[i - 1] + arr[i];
    }
    int S = preSum[n - 1] / 3;

    for (i = 0; i < n; i++) {
        if (preSum[i] % S == 0) {
            count++;
            index[k++] = i;
        }
    }
    if (count >= 3) {
        cout << index[0] << " "
             << index[1];// << ")";
        return 1;
    }
    return 0;
}
const int maxN=1000005;
int n, arr[maxN];
int main()
{
    freopen("splitarr.inp","r",stdin);
    freopen("splitarr.out","w",stdout);
    cin >> n;
    for (int i=0; i<n; i++)
        cin >> arr[i];
    if (findSplit(arr, n) == 0)
        cout << "-1";
    return 0;
}
```

Bài 6. SPECSUBSTR.* - Xâu con đặc biệt

Cho xâu s chỉ gồm các kí tự trong bảng chữ cái tiếng Anh thường, hãy đếm các xâu con đặc biệt của s. Một xâu đặc biện nếu xâu đó thỏa mãn một trong hai điều kiện sau:

- Xâu được bắt đầu bằng một nguyên âm và kết thúc bằng một phụ âm
- Xâu được bắt đầu bằng một phụ âm và kết thúc bằng một nguyên âm

Input:

- Một dòng gồm xâu s (có độ dài không quá 10^6)

Output:

- Ghi một số nguyên là kết quả của bài toán.

Ví dụ:

Input	Output
aba	2
adceba	9

Hướng dẫn:

Cách 1 (trâu): Duyệt mọi xâu con và kiểm tra xem xâu có thỏa mãn một trong hai điều kiện không và đếm.

Cách 2: Độ phức tạp O(n)

Dùng mảng cộng dồn đếm số lượng nguyên âm, phụ âm thuộc xâu phía sau vị trí kí tự i. Sau khi đếm được như vậy, duyệt xâu từ đầu tới cuối, với mỗi phụ âm, ta cộng thêm số nguyên âm phía sau nó và kết quả. Tương tự như vậy đối với mỗi phụ âm.

Code mẫu:

```
#include <bits/stdc++.h>
using namespace std;
bool isVowel(char ch)
{
    return (ch == 'a' || ch == 'e' ||
            ch == 'i' || ch == 'o' ||
            ch == 'u');
}
bool isCons(char ch)
{
    return (ch != 'a' && ch != 'e' &&
            ch != 'i' && ch != 'o' &&
            ch != 'u');
}
int countSpecial(string &str)
{
    int len = str.length();

    // co[i] lưu số phụ âm từ vị trí i đến cuối xâu
    // vo[i] lưu số nguyên âm từ vị trí i đến cuối xâu
    int co[len + 1];
    int vo[len + 1];
    memset(co, 0, sizeof(co));
    memset(vo, 0, sizeof(vo));

    // Ở mỗi vị trí i, đếm số lượng phụ âm và nguyên âm từ sau i tới cuối xâu
    if (isCons(str[len - 1]) == 1)
        co[len-1] = 1;
    else
        vo[len-1] = 1;
    for (int i = len-2; i >= 0; i--)
    {
        if (isCons(str[i]) == 1)
        {
            co[i] = co[i + 1] + 1;
```

```
            vo[i] = vo[i + 1];
        }
        else
        {
            co[i] = co[i + 1];
            vo[i] = vo[i + 1] + 1;
        }
    }

    // Duyệt qua các phần tử thuộc xâu
    long long ans = 0;
    for (int i = 0; i < len; i++)
    {
        // Nếu là nguyên âm thì số xâu con bắt đầu bằng str[i] sẽ bằng số phụ
âm phía sau nó
        if (isVowel(str[i]))
           ans = ans + co[i + 1];

        else
           ans = ans + vo[i + 1];
    }

    return ans;
}

int main()
{
    string str = "adceba";
    cout << countSpecial(str);
    return 0;
}
```

Bài 7. MINGROUP1.* - Gộp dãy toàn số 1

Cho dãy số A chỉ gồm các số có giá trị 0 hoặc 1. Hãy đếm số lượt đổi chỗ ÍT NHẤT các phần tử để gộp được tất cả các số 1 trong dãy vào một miền liên tiếp?

Input:

- Dòng 1 gồm số nguyên N chỉ số phần tử thuộc dãy ($1 \le N \le 10^6$¿
- Dòng 2 gồm N số nguyên chỉ mảng A (các phần tử cách nhau bởi dấu cách)

Output:

- Số lượt đổi chỗ ít nhất

Ví dụ:

Input	Output
5	1

1 0 1 0 1	
6 1 0 1 0 1 1	1

Hướng dẫn:

Cách trâu: Độ phức tạp O(n^2): đếm số lượng số 1 trong dãy. Giả sử là x. Ta cần tìm dãy con có độ dài x với số lượng số 1 nhiều nhất. Như vậy, số lượng phép tráo đổi cần thiết sẽ là số lượng số 0 trong dãy vừa tìm có độ dài x ở trên.

Cách 2. Độ phức tạp O(n). cải tiến từ cách trâu ở trên, sử dụng kĩ thuật dịch cửa sổ (sliding window). Ta gọi preCount là mảng lưu số lượng phần từ 1 trong mỗi dãy con - tính bằng kĩ thuật cộng dồn. Tiếp tục sử dụng kĩ thuật sliding window tìm đoạn con kích thước x có số lượng 1 nhiều nhất.

Code mẫu:

```
#include <bits/stdc++.h>
using namespace std;

int minSwaps(int arr[], int n) {
  int noOfOnes = 0;
  for (int i = 0; i < n; i++) {
    if (arr[i] == 1)
      noOfOnes++;
  }
  int x = noOfOnes;
  int maxOnes = INT_MIN;
  int preCompute[n] = {0}; //Mảng cộng dồn, lưu số lượng số 1 của dãy con từ
A1 tới An

  if (arr[0] == 1)
    preCompute[0] = 1;
  for (int i = 1; i < n; i++) {
    if (arr[i] == 1) {
      preCompute[i] = preCompute[i - 1] + 1;
    } else
      preCompute[i] = preCompute[i - 1];
  }

  for (int i = x - 1; i < n; i++) {
    if (i == (x - 1))
      noOfOnes = preCompute[i];
    else
      noOfOnes = preCompute[i] - preCompute[i - x];

    if (maxOnes < noOfOnes)
      maxOnes = noOfOnes;
```

```
  }

  int noOfZeroes = x - maxOnes;
  return noOfZeroes;
}
const int maxN = 1e6+5;
int n;
int a[maxN];
int main() {
    freopen("MINGROUP1.inp","r",stdin);
    freopen("MINGROUP1.out","w",stdout);
    cin >> n;
    for (int i=1; i<=n; i++) cin >> a[i];
    cout << minSwaps(a, n);
    return 0;
}
```

Bài 8. MULARR.* - Tích đặc biệt

Cho dãy A gồm N phần tử số nguyên. Tìm tổng các tích của của mỗi phần tử A[i] với các phần tử A[j] với mọi j>i.

Input:

- Dòng đầu ghi số N (N<=10^6)
- Dòng tiếp theo ghi N số nguyên, các số cách nhau bởi dấu cách A[i]<=10^6.

Output:

- Ghi một số là kết quả của bài toán

Ví dụ:

Input	Output	Giải thích VD
4 9 3 4 2	107	Tích = (9*3+9*4+9*2)+(3*4+3*2)+(4*2) = 107

III. Kết luận

Mảng cộng dồn (prefix sum) là một kĩ thuật khá phổ biến trong các bài toán Tin học, cũng phổ biến như kĩ thuật dịch cửa sổ (sliding window), kĩ thuật hai con trỏ (two pointers)… Nếu thành thạo kĩ thuật này, có thể vận dụng trong nhiều bài toán, nhằm đưa đến một hướng đơn giản, rõ ràng hơn rất nhiều.

Về code mẫu và test các thầy cô và học sinh có thể tham khảo tại đường dẫn này: https://drive.google.com/drive/folders/1arSg-oTLsMLAeN6Km0hWd3j42_8g_ojD?usp=sharing

Trên tinh thần học hỏi tác giả chuyên đề rất mong nhận được sự đóng góp của đồng nghiệp và học sinh để chuyên đề được hoàn thiện hơn. Xin chân thành cảm ơn!