The 2023 ICPC Asia Hue City Regional Contest

University of Science and Technology of Hanoi - USTH

UCC ICPC Team Notebook

1 Phi hàm Euler

```cpp
#include <bits/stdc++.h>
using namespace std;
#define si size()
#define int long long

const int maxn = 1e6 + 1;
signed main(){
    //freopen("a.inp", "r", stdin);
    //freopen("a.out", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    vector<int> p(maxn + 1, 0);
    for(int i = 2; i <= maxn; i++){
        if(p[i] == 0){
            p[i] = i;
            for(int j = i; i * j <= maxn; j++){
                if(p[i * j] == 0){
                    p[i * j] = i;
                }
            }
        }
    }
    int t; cin >> t;
    while(t--){
        int n; cin >> n;
        vector<pair<int, int>> x;
        while(n > 1){
            if(x.empty()){
                x.push_back({p[n], 1});
            }else{
                if(x.back().first == p[n]){
                    x.back().second += 1;
                }else{
                    x.push_back({p[n], 1});
                }
            }
```

```
            n /= p[n];
        }
        int res = 1;
        for(auto a: x){
            res *= ((a.first - 1) * pow(a.first, a.second - 1));
        }
        cout << res << "\n";
    }

}
```

2 Cây khung nhỏ nhất.

```
#include <iostream>
#include <vector>
#include <queue>
#include <climits>

using namespace std;

typedef pair<int, int> pii;

class Graph {
private:
    int V; // Số đỉnh
    vector<vector<pii>> adjList; // Danh sách kề

public:
    Graph(int vertices) : V(vertices), adjList(vertices) {}

    void addEdge(int u, int v, int weight) {
        adjList[u].push_back(make_pair(v, weight));
        adjList[v].push_back(make_pair(u, weight));
    }

    int primMST() {
        priority_queue<pii, vector<pii>, greater<pii>> pq; // Min heap để chọn cạnh có trọng s
        vector<int> key(V, INT_MAX); // Lưu trọng số tốt nhất hiện tại của mỗi đỉnh
        vector<bool> inMST(V, false); // Đỉnh đã được thêm vào cây khung

        int src = 0; // Bắt đầu từ đỉnh 0
        pq.push(make_pair(0, src));
        key[src] = 0;

        while (!pq.empty()) {
            int u = pq.top().second;
            pq.pop();
```

```
            inMST[u] = true;

            for (const auto& neighbor : adjList[u]) {
                int v = neighbor.first;
                int weight = neighbor.second;

                if (!inMST[v] && key[v] > weight) {
                    key[v] = weight;
                    pq.push(make_pair(key[v], v));
                }
            }
        }

        // Tính tổng trọng số của cây khung nhỏ nhất
        int minWeight = 0;
        for (int i = 0; i < V; ++i) {
            minWeight += key[i];
        }

        return minWeight;
    }
};

int main() {
    int V = 4; // Số đỉnh của đồ thị
    Graph g(V);

    // Thêm cạnh và trọng số tương ứng
    g.addEdge(0, 1, 2);
    g.addEdge(0, 2, 4);
    g.addEdge(1, 2, 1);
    g.addEdge(1, 3, 3);

    int minWeight = g.primMST();
    cout << "Tong trong so cay khung nho nhat: " << minWeight << endl;

    return 0;
}
```

3 In ra các cạnh của cây khung nhỏ nhất

```
#include <iostream>
#include <vector>
#include <queue>
#include <climits>
```

```cpp
using namespace std;

typedef pair<int, int> pii;

class Graph {
private:
    int V; // Số đỉnh
    vector<vector<pii>> adjList; // Danh sách kề

public:
    Graph(int vertices) : V(vertices), adjList(vertices) {}

    void addEdge(int u, int v, int weight) {
        adjList[u].push_back(make_pair(v, weight));
        adjList[v].push_back(make_pair(u, weight));
    }

    void primMST() {
        priority_queue<pii, vector<pii>, greater<pii>> pq; // Min heap để chọn cạnh có trọng s
        vector<int> key(V, INT_MAX); // Lưu trọng số tốt nhất hiện tại của mỗi đỉnh
        vector<int> parent(V, -1); // Lưu đỉnh cha của mỗi đỉnh

        int src = 0; // Bắt đầu từ đỉnh 0
        pq.push(make_pair(0, src));
        key[src] = 0;

        while (!pq.empty()) {
            int u = pq.top().second;
            pq.pop();

            for (const auto& neighbor : adjList[u]) {
                int v = neighbor.first;
                int weight = neighbor.second;

                if (key[v] > weight) {
                    key[v] = weight;
                    pq.push(make_pair(key[v], v));
                    parent[v] = u;
                }
            }
        }

        // In ra các cạnh của cây khung nhỏ nhất
        for (int i = 1; i < V; ++i) {
            cout << "Cạnh: " << parent[i] << " - " << i << " Trọng số: " << key[i] << endl;
        }
    }
};
```

```
int main() {
    int V = 4; // Số đỉnh của đồ thị
    Graph g(V);

    // Thêm cạnh và trọng số tương ứng
    g.addEdge(0, 1, 2);
    g.addEdge(0, 2, 4);
    g.addEdge(1, 2, 1);
    g.addEdge(1, 3, 3);

    cout << "Cac canh cua cay khung nho nhat:" << endl;
    g.primMST();

    return 0;
}
```

4 Đường đi ngắn nhất Dijkstra

```
#include <iostream>
#include <vector>
#include <queue>
#include <climits>

using namespace std;

typedef pair<int, int> pii;

class Graph {
private:
    int V; // Số đỉnh
    vector<vector<pii>> adjList; // Danh sách kề

public:
    Graph(int vertices) : V(vertices), adjList(vertices) {}

    void addEdge(int u, int v, int weight) {
        adjList[u].push_back(make_pair(v, weight));
        // Đối với đồ thị vô hướng, bạn cần thêm cả cạnh (v, u)
        adjList[v].push_back(make_pair(u, weight));
    }

    void dijkstra(int src) {
        priority_queue<pii, vector<pii>, greater<pii>> pq; // Min heap để chọn đỉnh có đường đ
        vector<int> dist(V, INT_MAX); // Lưu đường đi ngắn nhất tạm thời từ src đến các đỉnh k
        dist[src] = 0;
        pq.push(make_pair(0, src));
```

```
        while (!pq.empty()) {
            int u = pq.top().second;
            pq.pop();

            for (const auto& neighbor : adjList[u]) {
                int v = neighbor.first;
                int weight = neighbor.second;

                if (dist[v] > dist[u] + weight) {
                    dist[v] = dist[u] + weight;
                    pq.push(make_pair(dist[v], v));
                }
            }
        }

        // In ra đường đi ngắn nhất từ src đến các đỉnh khác
        cout << "Duong di ngan nhat tu dinh " << src << " den cac dinh khac:" << endl;
        for (int i = 0; i < V; ++i) {
            cout << "Den dinh " << i << ": " << dist[i] << endl;
        }
    }
};

int main() {
    int V = 6; // Số đỉnh của đồ thị
    Graph g(V);

    // Thêm cạnh và trọng số tương ứng
    g.addEdge(0, 1, 2);
    g.addEdge(0, 2, 4);
    g.addEdge(1, 2, 1);
    g.addEdge(1, 3, 7);
    g.addEdge(2, 4, 3);
    g.addEdge(3, 5, 1);
    g.addEdge(4, 5, 5);

    int src = 0; // Đỉnh nguồn
    g.dijkstra(src);

    return 0;
}
```

5 Cây phân đoạn (Segment Tree)

```
#include <iostream>
#include <vector>
```

```cpp
using namespace std;

class SegmentTree {
private:
    vector<int> tree;
    vector<int> arr;

    // Xây dựng cây phân đoạn từ dãy số arr
    void build(int node, int start, int end) {
        if (start == end) {
            tree[node] = arr[start];
        } else {
            int mid = (start + end) / 2;
            int leftChild = 2 * node + 1;
            int rightChild = 2 * node + 2;

            build(leftChild, start, mid);
            build(rightChild, mid + 1, end);

            tree[node] = tree[leftChild] + tree[rightChild];
        }
    }

    // Truy vấn tổng trong đoạn [qStart, qEnd]
    int query(int node, int start, int end, int qStart, int qEnd) {
        if (qStart > end || qEnd < start) {
            return 0; // Đoạn [qStart, qEnd] nằm ngoài đoạn [start, end]
        }

        if (qStart <= start && qEnd >= end) {
            return tree[node]; // Đoạn [start, end] hoàn toàn nằm trong [qStart, qEnd]
        }

        int mid = (start + end) / 2;
        int leftChild = 2 * node + 1;
        int rightChild = 2 * node + 2;

        int leftSum = query(leftChild, start, mid, qStart, qEnd);
        int rightSum = query(rightChild, mid + 1, end, qStart, qEnd);

        return leftSum + rightSum;
    }

public:
    // Khởi tạo cây phân đoạn từ dãy số arr
    SegmentTree(const vector<int>& array) {
        arr = array;
```

```
        int n = arr.size();

        // Số lượng phần tử của cây phân đoạn có thể lớn hơn 2 * n - 1
        // để đảm bảo cây là một cây đầy đủ
        int treeSize = 2 * n - 1;
        tree.resize(treeSize, 0);

        build(0, 0, n - 1);
    }

    // Truy vấn tổng trong đoạn [qStart, qEnd]
    int query(int qStart, int qEnd) {
        int n = arr.size();
        return query(0, 0, n - 1, qStart, qEnd);
    }
};

int main() {
    int numTests;
    cin >> numTests;

    while (numTests--) {
        int n;
        cin >> n;

        vector<int> arr(n);
        for (int i = 0; i < n; ++i) {
            cin >> arr[i];
        }

        SegmentTree segmentTree(arr);

        int numQueries;
        cin >> numQueries;

        cout << "Ket qua cac truy van:" << endl;
        while (numQueries--) {
            int qStart, qEnd;
            cin >> qStart >> qEnd;

            int result = segmentTree.query(qStart, qEnd);
            cout << "Tong tu " << qStart << " den " << qEnd << ": " << result << endl;
        }
    }

    return 0;
}
```

6 Kỹ thuật chia căn để tính tổng

```cpp
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

class SquareRootDecomposition {
private:
    vector<int> blockSum;
    vector<int> arr;
    int blockSize;

public:
    SquareRootDecomposition(const vector<int>& array) {
        arr = array;
        int n = arr.size();
        blockSize = static_cast<int>(sqrt(n)) + 1;

        // Khởi tạo và tính toán tổng của từng khối
        blockSum.resize(blockSize, 0);
        for (int i = 0; i < n; ++i) {
            blockSum[i / blockSize] += arr[i];
        }
    }

    // Truy vấn tổng trong đoạn [qStart, qEnd]
    int query(int qStart, int qEnd) {
        int sum = 0;

        // Truy vấn trên khối đầu tiên
        while (qStart < qEnd && qStart % blockSize != 0) {
            sum += arr[qStart];
            qStart++;
        }

        // Truy vấn trên các khối tiếp theo
        while (qStart + blockSize <= qEnd) {
            sum += blockSum[qStart / blockSize];
            qStart += blockSize;
        }

        // Truy vấn trên khối cuối cùng (nếu có)
        while (qStart <= qEnd) {
            sum += arr[qStart];
            qStart++;
        }
```

```
        return sum;
    }

    // Cập nhật giá trị tại vị trí index thành newValue
    void update(int index, int newValue) {
        // Tìm khối chứa phần tử cần cập nhật
        int blockIndex = index / blockSize;

        // Cập nhật tổng trong khối
        blockSum[blockIndex] += (newValue - arr[index]);

        // Cập nhật giá trị trong mảng
        arr[index] = newValue;
    }
};

int main() {
    vector<int> arr = {1, 3, 5, 7, 9, 11};
    SquareRootDecomposition sqrtDecomp(arr);

    // Truy vấn tổng trong đoạn [1, 4]
    int result = sqrtDecomp.query(1, 4);
    cout << "Tong tu 1 den 4: " << result << endl;

    // Cập nhật giá trị tại vị trí 2 thành 8
    sqrtDecomp.update(2, 8);

    // Truy vấn tổng sau khi cập nhật trong đoạn [0, 5]
    result = sqrtDecomp.query(0, 5);
    cout << "Tong tu 0 den 5 sau khi cap nhat: " << result << endl;

    return 0;
}
```

7 Cho một số xóa đi k số của số (số còn lại lớn nhất)

```
58812
2
```

```
882
```

```
#include <bits/stdc++.h>
#define int long long
int k;
using namespace std;

string n;
```

```
stack<char> st;

void enter()
{
    cin >> n;
    cin >> k;
    for(int i = 0; i  < n.size(); i++)
    {
        while(!st.empty() && st.top() < n[i] && k > 0)
        {
            st.pop();
            k --;
        }
        st.push(n[i]);
    }
    string d = "";
    while(st.size() != 0)
    {
        d = d + st.top();
        st.pop();
    }
    for(int i = d.size() - 1 ; i >= 0; i --)
        cout << d[i];
}

main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    //freopen("a.inp","r", stdin);
    //freopen("a.out", "w", stdout);
    enter();
}
```

8 Đếm cách trả tiền

Một cửa hàng có n đồng xu mệnh giá lần lượt là $c_1, c_2, ..c_n$số lượng mỗi loại coi như không giới hạn. Cửa hàng cần phải trả lại cho một vị khách số tiền đúng bằng m. Tuy nhiên, do trong ngày còn rất nhiều khách hàng, nên chủ cửa hàng cần tính toán số cách trả tiền có thể để lựa chọn một cách phù hợp nhất.

Yêu cầu: Hãy giúp chủ cửa hàng đếm số lượng cách trả tiền cho vị khách nói trên số tiền đúng bằng m? Biết rằng, hai cách trả tiền là hoán vị của nhau chỉ tính là một cách.

```
#include <bits/stdc++.h>
using namespace std;
#define endl "\n"
```

```
const int ll = 1e6, oo = 1e9, mod = 1e9 + 7;
int dp[101][ll + 1],t[ll], n, m;
void enter()
{
    cin >> n >> m;
    for(int  i = 1; i <=n; i++)
    {
        cin >> t[i];
    }
}

void build()
{
    dp[0][0] = 1;
    for (int i = 1; i <= n; i++){
        for (int j = 0; j <= m; j++)
        {
            dp[i][j] = dp[i - 1][j];
            if (j >= t[i])
                dp[i][j] = (dp[i][j] + dp[i][j - t[i]]) % mod;
        }
    }

    cout << dp[n][m];
}
main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    enter();
    build();

}
```

9 Xâu Con Chung

Cho hai xâu kí tự A và B. Một xâu con C được gọi là xâu con chung của hai xâu A và B nếu như có thể xóa đi một số kí tự của A và B, giữ nguyên thứ tự các kí tự còn lại thì thu được C.

Yêu cầu: Hãy tìm xâu con chung dài nhất của hai xâu A và B?

```
#include<bits/stdc++.h>
#define float double
using namespace std;
string a, b;
int dp[1001][1001];
int ma, mb;
void enter()
```

```
{
    cin >> a;
    cin >> b;
    ma = a.length();
    mb = b.length();
}
void build()
{
    int res = 0;
    for(int i = 1; i <= ma; i++)
    {
        for(int j = 1; j <= mb; j++)
        {
            if(a[i - 1] == b[j - 1 ]) dp[i][j] = dp[i - 1][ j - 1] + 1;
            else dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            res = max(dp[i][j], res);
        }
    }
    cout << res;
}
main()
{
    enter();
    build();
}
```

10 Cây khế

Truyện cổ tích Cây khế là một câu chuyện rất quen thuộc với tuổi thơ của mỗi người. Trong chuyện, chúng ta đã biết người em được chim thần yêu cầu mang theo túi ba gang để lấy vàng trả cho những quả khế. Tuy nhiên, chiếc túi của người em chỉ có trọng lượng tối đa là m. Trên hòn đảo chứa vàng có n thỏi vàng lớn, mỗi thỏi vàng có giá trị là a và khối lượng là b. Vì số lượng vàng quá nhiều nên người em không biết phải lấy những thỏi vàng nào để có thể bán được nhiều tiền nhất.

Yêu cầu: Hãy giúp người em chọn ra những thỏi vàng sao cho tổng khối lượng của chúng không vượt quá tải trọng m của chiếc túi, và tổng giá trị của chúng là lớn nhất?

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define endl "\n"
const int ll = 1e7, oo = 1e9;
int dp[1001][1001], n, m;
vector<pair<int , int>> v;
void enter()
{
    cin >> n >> m;
```

```
    v.push_back({oo,oo});
    for(int  i = 1; i <=n; i++)
    {
        int a, b;
        cin >> a >> b;
        v.push_back({b, a});
    }

}
void build()
{
    for(int i = 1; i <= n; i++)
    {
        for(int j = 0; j <= m; j++)
        {
            dp[i][j] = dp[i - 1][j];
            if(j >= v[i].first)
                dp[i][j] = max(dp[i][j], dp[i - 1][j - v[i].first] + v[i].second);
        }
    }
    cout << dp[n][m];
}
main()
{
    enter();
    build();
}
```

11 Tổ hợp C

```
int C(int k, int n) {
    if (k == 0 || k == n) return 1;
    if (k == 1) return n;
    return C(k - 1, n - 1) + C(k, n - 1);
}
```

12 Tổ hợp

Cho tập A gồm 5 chữ số hệ thập phân $A = 1, 2, 3, 4, 5$

- Số các số tự nhiên 4 chữ số lập thành từ 5 chữ số trên là $5^4 = 625$
- Số các số tự nhiên gồm 3 chữ số khác nhau lập thành từ 5 chữ số trên là $A_5^3 = \dfrac{5!}{2!} = 60$
- Số các tập con 3 phần tử của 5 chữ số trên là $C_5^3 = \dfrac{5!}{2!3!} = 10.$
- Số các hoán vị của 5 số đó là $5! = 120.$

- Số các hoán vị vòng quanh là $Q(n) = 4! = 24$.
- Số các hoán vị khác nhau có thể có khi hoán vị các chữ cái trong từ XAXAM là $\frac{5!}{2!2!1!} = 30$.
- Số cách chia 7 chiếc kẹo cho 4 trẻ em là tổ hợp lặp chập 4 của 7

Dưới đây là 33 ví dụ về tổ hợp toán học:

1. Số cách chọn 2 phần tử từ 3 phần tử là $(C(3, 2) = 3)$.
2. Số cách xếp 6 ghế trên một hàng xếp hình là $(P(6) = 720)$.
3. Số cách chọn 3 môn học từ 8 môn học để học trong học kỳ là $(C(8, 3) = 56)$.
4. Số cách chia 12 sinh viên thành 3 nhóm là $(C(12, 4) = 495)$.
5. Số cách xếp 4 cuốn sách sách giáo trình và 3 cuốn sách giáo trình vào một kệ sách là $(P(7) = 5040)$.
6. Số cách chọn 6 bài hát từ 15 bài hát để tạo một danh sách phát là $(C(15, 6) = 5,005)$.
7. Số cách sắp xếp 8 học sinh vào 2 hàng là $(P(8) = 40,320)$.
8. Số cách chọn 4 ô vuông từ một bảng 8x8 để tạo một hình vuông là $(C(64, 4) = 635,376)$.
9. Số cách sắp xếp 10 đội bóng vào vòng đấu loại trực tiếp là $(P(10) = 3,628,800)$.
10. Số cách chọn 7 quyển sách từ 10 quyển sách để đọc là $(C(10, 7) = 120)$.
11. Số cách sắp xếp 9 cặp giày vào một giá đựng giày là $(P(18) = 6,402,373,705,728,000)$.

12.1 Chia 3 nhóm tổng gần nhất.

```cpp
#include <bits/stdc++.h>
#define ii pair<int, int>
#define st first
#define nd second
#define endl "\n"
#define all(v) v.begin(), v.end()
#define Unique(v) v.erase(unique(all(v)), v.end())

using namespace std;
const int oo = 1e9;

const int base = 1e5;
const int maxw = 1e5 + 5;
const int maxn = 505;

int n, a[maxn];
int dp[2][maxw + base];

void PROGRAM(int _){
    cin >> n;

    for (int i = 1; i <= n; i++) cin >> a[i];

    int sum = accumulate(a + 1, a + n + 1, 0ll);
```

```
    for (int weight = 0; weight <= base * 2; weight++){
        dp[0][weight] = oo;
        dp[1][weight] = oo;
    }

    dp[0][base] = 0;

    for (int i = 1; i <= n; i++){
        for (int w = 0; w <= base * 2; w++){
            if (w + a[i] <= base * 2) dp[1][w] = min(dp[1][w], dp[0][w + a[i]]);
            if (w - a[i] >= 0) dp[1][w] = min(dp[1][w], dp[0][w - a[i]]);
            dp[1][w] = min(dp[1][w], dp[0][w] + a[i]);
        }
        for (int w = 0; w <= base * 2; w++){
            dp[0][w] = dp[1][w];
            dp[1][w] = oo;
        }
    }

    cout << (sum - dp[0][base]) / 2;

}

signed main(){
    freopen("CT.INP", "r", stdin);
    freopen("CT.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(0);

    int test = 1;

    for (int _ = 1; _ <= test; _++){
        PROGRAM(_);
    }
    return 0;
}
```

13 Tổng toàn bộ

Cho dãy số nguyên a gồm n phần tử. Hãy tính tổng của $|a_i - a_j|$ với mọi cặp (i, j) thỏa mãn $1 \leq i < j \leq n$.

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define s second
const int ll = 1e6 + 10;
```

```cpp
int n, a[ll], sum[ll], res = 0;
main()
{
    //freopen("a.inp","r", stdin);
    //freopen("a.out", "w", stdout);
    cin >> n;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    sort(a + 1, a + 1 + n);
    for(int i = 1; i <= n; i++)
    {
        sum[i] = sum[i - 1] + a[i];
    }
    for(int i = n; i >= 1; i --)
    {
        res += ((a[i] * (i - 1)) - sum[i - 1]);
    }
    cout << res;
}
```

14 Dãy con liên tiếp đối xứng dài nhất.

```
5
abacd
```

```
3
```

```python
def build(s):
    t = '^'
    for ch in s:
        assert ch != '^' and ch != '#' and ch != '$'
        t += '#' + ch
    t += '#$'
    n = len(t)
    p = [0] * (n + 7)
    c = 0
    r = 0
    for i in range(1, n - 1):
        if r > i:
            p[i] = min(r - i, p[2 * c - i])
        else:
            p[i] = 0
        while t[i + 1 + p[i]] == t[i - 1 - p[i]]:
            p[i] += 1
        if i + p[i] > r:
            c = i
            r = i + p[i]
```

```
    ret = 0
    for i in range(1, n - 1):
        ret = max(ret, p[i])
    return ret
n = input()
s = input()
print(build(s))
```

15 Max GCD

Cho hai mảng a: $a_1, a_2, a_3, a_4...a_n$ và b: $b_1, b_2, b_3, b_4...b_m$. Với mỗi $j = 1, 2, ..., m$ tìm ước chung lớn nhất của $(a1 + bj, a2 + bj, ...an + bj)$

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int ll = 1e6 + 10;
int a[ll], n, m;
signed main(){
    cin  >> n >> m;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    int g = a[2] - a[1];
    for(int i = 10; i<= n; i++){
        g = __gcd(g, a[i] - a[1]);
    }
    for(int i = 1; i <= m; i++){
        int u;  cin >> u;
        cout << abs(__gcd(a[1] + u, g)) <<  ' ';
    }
}
```

16 Chia 2 mảng con có hiệu nhỏ nhất

```
using namespace std;

int main() {
    int n;
    cout << "Nhap so phan tu cua tap A: ";
    cin >> n;

    vector<int> A(n);
    cout << "Nhap cac phan tu cua tap A:\n";
    for (int i = 0; i < n; ++i) {
        cin >> A[i];
    }
```

```
    // Tính tổng tất cả các phần tử của tập A
    int sum = 0;
    for (int i : A) {
        sum += i;
    }

    // Sử dụng mảng dynamic programming để lưu thông tin
    vector<vector<bool>> dp(n + 1, vector<bool>(sum + 1, false));
    // Khởi tạo trạng thái cơ bản
    for (int i = 0; i <= n; ++i) {
        dp[i][0] = true;
    }
    // Cập nhật trạng thái
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= sum; ++j) {
            dp[i][j] = dp[i - 1][j];
            if (A[i - 1] <= j) {
                dp[i][j] = dp[i][j] || dp[i - 1][j - A[i - 1]];
            }
        }
    }

    // Tìm hiệu số tốt nhất
    int minDiff = sum;
    for (int j = sum / 2; j >= 0; --j) {
        if (dp[n][j]) {
            minDiff = sum - 2 * j;
            break;
        }
    }

    cout << "Hieu so giua hai tap con gan nhau nhat la: " << minDiff << endl;
}
```