

SỬ DỤNG NGÔN NGỮ C++ DẠY HSG MÔN TIN HỌC NAM ĐỊNH

A – MỤC ĐÍCH, YÊU CẦU

Phong trào thi học sinh giỏi Tin học diễn ra trên khắp các tỉnh thành ở nước ta nói riêng và các nước trên thế giới nói chung hướng tới các mục đích:

- Đẩy mạnh phong trào dạy và học Tin học nhằm đáp ứng các yêu cầu của cuộc sống đang được tin học hóa sâu rộng trong mọi lĩnh vực.
- Phát hiện các học sinh có năng khiếu để đào tạo và phát triển nguồn nhân lực đỉnh cao, có tri thức và có tay nghề theo kịp sự phát triển của xã hội.

Dạy và thi học sinh giỏi tin học ở tỉnh Nam Định cũng diễn ra nhiều năm ở hai cấp học THCS và THPT. Bắt đầu từ năm học 2017-2018, Sở Giáo dục và Đào tạo bắt đầu triển khai giảng dạy ngôn ngữ lập trình C++ với hệ thống lập trình CodeBlocks cho các trường THPT trong toàn tỉnh.

Để đáp ứng yêu cầu đề ra và mong muốn chia sẻ kinh nghiệm về dạy ngôn ngữ lập trình C++ cho đối tượng HSG tôi mạnh dạn viết chuyên đề: “Sử dụng ngôn ngữ lập trình C++ trong dạy HSG”.

B. GIẢI PHÁP

Trang bị cho học sinh có kiến thức vững chắc về ngôn ngữ lập trình C++ cùng với hệ thống lập trình cần sử dụng để hỗ trợ cho ngôn ngữ đó.

Biết càng nhiều và sâu các thư viện chuẩn hỗ trợ lập trình để sử dụng chúng một cách tối ưu.

Tạo được thói quen suy nghĩ và làm việc phù hợp với ngôn ngữ lập trình đang sử dụng. Ở nước ta, trong bậc PTTH hệ thống lập trình được sử dụng phổ biến là ngôn ngữ PASCAL với hệ thống lập trình Free Pascal và ngôn ngữ C++ với hệ thống lập trình CodeBlocks. Việc trang bị công cụ cho học sinh năng khiếu, phục vụ cho các kỳ thi Tin học là vấn đề ta cần xem xét và xử lý.

Trong chuyên đề này tôi tập trung vào giải các bài toán bằng ngôn ngữ lập trình C++ với hệ thống lập trình CodeBlocks và giới thiệu một số thư viện C++.

C. NỘI DUNG

C1. Giới thiệu các dạng bài trong bồi dưỡng HSG.

Để hỗ trợ cho việc triển khai giảng dạy ngôn ngữ C++ (dựa trên cơ sở hệ thống lập trình CodeBlocks). Các bài tập dưới đây sẽ được giới thiệu kèm theo với lời giải và chương trình mẫu.

Sau đây tôi phân chia bài tập cơ bản gồm bốn dạng:

- + Các bài tập về số học.

- + Các bài tập về xử lí trên dãy số.
- + Các bài tập sử dụng xâu.
- + Các bài tập sử dụng mảng tính trước

1. Các bài tập về số học.

Bài tập 1: Kiểm tra tính nguyên tố của một số nguyên dương N ?

Input: cho số nguyên dương N ($N \leq 10^{14}$).

Output: đưa ra “Yes” nếu N là nguyên tố hoặc đưa ra “NO” nếu N không là số nguyên tố.

Hướng dẫn giải thuật:

- Do đây là bài quá kinh điển nên không cần trình bày dài dòng, sau đây là chương trình được cài đặt bằng ngôn ngữ C++.

```
#include <iostream>
using namespace std;
bool prime(long long n)
{
    if (n<2) return false;
    for(long long i=2;i*i<=n;i++)
        if (n%i==0) return false;
    return true;
}
int main()
{
    long long n;
    cin>>n;
    if (prime(n)) cout<<n<<" la so nguyen to.";
    else cout<<n<<" khong la so nguyen to.";
    return 0;
}
```

Bài tập 2: ƯỚC SỐ

Cho số nguyên dương n ($1 \leq n \leq 10^{14}$). Với số n đã cho hãy xác định số lượng ước số của nó.

Ví dụ với $n = 4$, số lượng ước số của 4 là 3 (1, 2, 4).

Dữ liệu: Vào từ file văn bản DIVISOR.INP: chứa số nguyên n ,

Kết quả: Đưa ra file văn bản DIVISOR.OUT chứa số lượng ước số của n .

Ví dụ:

DIVISOR.INP	DIVISOR.OUT
4	3

Hướng dẫn giải thuật:

Đây là một bài cơ bản, nhưng do kích thước của n lớn nên nếu học sinh không hiểu căn kẽ thì khi làm chỉ giải quyết được những test nhỏ.

Dựa vào tính chất ước số ta có nhận xét sau: nếu a là ước của n thì ta có $n \div a$ cũng là ước của n .

Ta chỉ cần thử tất cả các số từ 1 đến \sqrt{n} là đếm được số ước của n .

Chương trình:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    freopen("DIVISOR.INP","r",stdin);
    freopen("DIVISOR.OUT","w",stdout);
    long long n;
    cin>>n;
    int d=0;
    for(long long i=1;i*i<=n;i++)
        if (n%i==0)
        {
            d++;
            if (i!=n/i) d++;
        }
    cout<<d;
    return 0;
}
```

Bài tập 3: KHÔNG CHỨA CHÍNH PHƯƠNG

Một số nguyên gọi là không chứa chính phương nếu nó không chia hết cho bất kỳ số nguyên nào dạng x^2 với $x > 1$.

Yêu cầu: Cho số nguyên n ($1 \leq n \leq 10^{13}$). Hãy tìm ước lớn nhất không chứa chính phương của n . Lưu ý là 1 và n cũng là ước của n .

Dữ liệu: Vào từ file văn bản SQFREE.INP gồm nhiều tests, mỗi test cho trên một dòng chứa số nguyên n .

Kết quả: Đưa ra file văn bản SQFREE.OUT, kết quả mỗi test đưa ra trên một dòng.

Ví dụ:

SQFREE.INP	SQFREE.OUT
9	3
20	10

Hướng dẫn giải thuật:

- Nếu làm bài toán này theo đúng định nghĩa của bài đưa ra (hay làm theo cách tự nhiên) thì chỉ qua được những test nhỏ.
- Để giải quyết triệt để bài toán tìm ước không chứa chính phương của N ta phân tích N thành tích các thừa số nguyên tố, ước lớn nhất thỏa mãn yêu cầu của bài là tích các thừa số nguyên tố.

Ví dụ $N=2^i 3^j 5^k$ thì ước lớn nhất không chứa chính phương là $2.3.5$

Chương trình:

```
#include <bits/stdc++.h>
using namespace std;
ifstream fi("SQFREE.INP");
ofstream fo("SQFREE.OUT");
long long n,ans;
int main()
{
    while (!fi.eof()){
        fi>>n;
        if (fi.fail()) break;
        ans=1;
        for (long long i=2;i*i<=n;i++){
            if (n % i==0) ans*=i;
            while (n%i==0) n/=i;
        }
        if (n>1) ans*=n;
        fo<<ans<<"\n";
    }
    fi.close();
    fo.close();
    return 0;
}
```

2. Các bài tập về dãy số

Tận dụng khả năng của máy tính hiện nay, bộ nhớ trong kích thước lớn. Hệ thống lập trình Free Pascal mở rộng bộ nhớ cho chương trình lớn gấp nhiều lần so với Turbo Pascal (Điều này đã được nói kĩ trong chuyên đề tập huấn đầu năm). Sau đây là một số bài tập cho thấy việc bộ nhớ chương trình mở rộng giúp cho việc giải một số bài tập một cách dễ dàng.

Bài tập 1: ĐỊNH ĐỀ BERTRAN

Định đề Bertran: “Với mọi số nguyên $n \geq 2$ bao giờ cũng tìm thấy số nguyên tố p thỏa mãn $n < p < 2n$ ”. Định đề này do nhà toán học Pháp Jojeph Bertran đưa ra năm 1845 sau khi đã kiểm tra với mọi $n \leq 3\,000\,000$. Điều này đã được Tchebursep chứng minh năm 1850. Năm 1932 Erdoeus đã tìm được cách chứng minh mới đơn giản hơn.

Nhiệm vụ của bạn rộng hơn một chút: với n cho trước, hãy xác định số lượng số nguyên tố p thỏa mãn điều kiện $n < p < 2n$.

Dữ liệu: Vào từ file văn bản BERTRAN.INP gồm nhiều tests, mỗi test cho trên một dòng chứa số nguyên dương n ($1 \leq n \leq 10^7$).

Kết quả: Đưa ra file văn bản BERTRAN.OUT, kết quả mỗi test đưa ra trên một dòng dưới dạng một số nguyên.

Ví dụ:

BERTRAN.INP	BERTRAN.OUT
2	1
239	39
3000	353

Hướng dẫn giải thuật:

+ Với kích thước của bài toán trên, nếu với mỗi số n ta đi thử tất cả các số m từ $n+1$ tới $2n$, kiểm tra m có là số nguyên tố hay không thì chương trình chạy chậm và không đạt yêu cầu bài.

+ Để làm bài này ta phải sử dụng thuật toán sàng số nguyên tố. Ta khai báo mảng a chứa 10 triệu phần tử. Trong đó $a[i]=\text{true}$ nếu i là nguyên tố ngược lại thì i không là nguyên tố.

Chương trình:

```
#include <bits/stdc++.h>

using namespace std;
const int MAXN=3000000+5;
int n;
bool prime[MAXN*2];
void khoitao()
{
    memset(prime,true,sizeof(prime));
    for(int i=2;i*i<2*MAXN;i++)
        if (prime[i])
            for(int j=2;i*j<2*MAXN;j++)
                prime[i*j]=false;
    return;
}
int main()
{
    freopen("BERTRAN.INP","r",stdin);
    freopen("BERTRAN.OUT","w",stdout);
    khoitao();
    while (cin>>n)
    {
```

```

        int ans=0;
        for(int i=n+1;i<2*n;i++)
            ans+=(int)prime[i];
        cout<<ans<<'\n';
    }
    return 0;
}

```

Bài tập 2: Bộ tộc

Có một hòn đảo rất đẹp, thu hút nhiều khách du lịch đến thăm. Trên đảo có n người thuộc nhiều bộ tộc sinh sống. Dân cư trên đảo rất thân thiện. Mỗi người thuộc một bộ tộc nào đó. Trong đoàn du lịch có một nhà nhân chủng học. Tranh thủ dịp may được ghé thăm đảo ông không bỏ phí thời gian tiến hành khảo sát. Ông gặp từng người một trên đảo với một câu hỏi duy nhất: “Trên đảo, bộ tộc của bạn có bao nhiêu người?”. Từ kết quả khảo sát, ông đã xác định được số bộ tộc khác nhau tồn tại trên đảo.

Ví dụ: với $n=10$ và các câu trả lời là 5, 1, 2, 5, 5, 2, 5, 5, 2, 2 ta có thể suy ra là trên đảo có 4 bộ tộc khác nhau.

Yêu cầu: cho n và các câu trả lời. Hãy xác định số bộ tộc trên đảo. Dữ liệu đảm bảo bài toán có nghiệm.

Dữ liệu: vào từ file văn bản CLAN.INP

- Dòng 1 chứa số nguyên n ($1 < n < 3000000$)
- Mỗi dòng trong n dòng sau chứa một số nguyên, câu trả lời nhận được.

Kết quả: Đưa ra file văn bản CLAN.OUT một số nguyên, số bộ tộc trên đảo.

Ví dụ:

Clan.inp	Clan.out
10	4
5	
1	
2	
5	
5	
2	
5	
5	
2	
2	

Hướng dẫn giải thuật

- Gọi $d[i]$ là số người có câu trả lời là i , số bộ tộc có câu trả lời i là $d[i] \text{ div } i$. Ta chỉ việc đếm số lần xuất hiện của i lưu vào mảng $d[i]$ là giải quyết xong bài toán

Chương trình:

```
#include <bits/stdc++.h>
```

```

using namespace std;
int ans[3000000+5], n;
int main()
{
    freopen("CLAN.INP", "r", stdin);
    freopen("CLAN.OUT", "w", stdout);
    scanf("%d", &n);
    for(int i=1; i<=n; i++)
    {
        int x;
        scanf("%d", &x);
        ans[x]++;
    }
    int kq=0;
    for(int i=1; i<=n; i++) kq+=ans[i]/i;
    printf("%d", kq);
    return 0;
}

```

Bài tập 3: HÀNG CÂY

Trong khu vườn, người ta trồng một hàng cây chạy dài gồm có N cây, mỗi cây có độ cao là a_1, a_2, \dots, a_N .

Người ta cần lấy M mét gỗ bằng cách đặt cửa máy sao cho lưỡi cưa ở độ cao H (mét) để cưa tất cả các cây có độ cao lớn hơn H (dĩ nhiên những cây có độ cao không lớn hơn H thì không bị cưa).

Ví dụ: Nếu hàng cây có các cây với độ cao tương ứng là 20; 15; 10 và 18 mét, cần lấy 7 mét gỗ. Lưỡi cưa đặt tại độ cao hợp lý là 15 mét thì độ cao của các cây còn lại sau khi bị cưa tương ứng là 15; 15; 10 và 15 mét. Tổng số mét gỗ lấy được là 8 mét (dư 1 mét).

Yêu cầu: Hãy tìm vị trí đặt lưỡi cưa hợp lý (số nguyên H lớn nhất) sao cho lấy được M mét gỗ và số mét gỗ dư ra là ít nhất.

Dữ liệu: Vào từ tệp văn bản WOOD.INP

Dòng thứ nhất chứa 2 số nguyên dương N và M ($1 \leq N \leq 10^6$; $1 \leq M \leq 2 \times 10^9$) cách nhau một dấu cách.

Dòng thứ hai chứa N số nguyên dương a_i là độ cao của mỗi cây trong hàng ($1 \leq a_i \leq 10^9$, $i=1 \dots N$), mỗi số cách nhau ít nhất một dấu cách.

Kết quả: Đưa ra tệp WOOD.OUT là một số nguyên cho biết giá trị cần tìm.

Ví dụ:

WOOD.INP	WOOD.OUT
4 7 20 15 10 18	15

Hướng dẫn giải thuật:

- Cách 1: Thuật toán vét cạn thử tất cả các độ cao từ cây cao nhất ($\max H$) tới 0. Với mỗi độ cao H , tính tổng số mét gỗ thu được, nếu tổng lớn hơn bằng M đầu tiên ta thu được độ cao cần tìm (đây là cách không đạt điểm tối đa).

- Cách 2: Ta có nhận xét, nếu cắt ở độ cao H lấy được M mét gỗ thì khi cắt độ cao H-1 cũng lấy được M mét gỗ. Ngược lại, nếu cắt ở độ cao H không lấy được M mét gỗ thì khi cắt độ cao H+1 cũng không lấy được M mét gỗ. Từ nhận xét trên ta có thể sử dụng giải thuật tìm kiếm nhị phân để tìm độ cao H cao nhất cần cắt để lấy được M mét gỗ thỏa mãn yêu cầu bài.

Chương trình:

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
int n;
ll m, h[1000005];
bool ok(int x)
{
    ll s=0;
    for(int i=1; i<=n; i++)
        if (h[i]>x) s+=h[i]-x;
    if (s>=m) return true;
    return false;
}
int main()
{
    freopen("wood.inp", "r", stdin);
    freopen("wood.out", "w", stdout);
    scanf("%d %lld", &n, &m);
    for(int i=1; i<=n; i++) scanf("%lld", &h[i]);
    int l=0, r=1000000000, mid;
    while (l<r)
    {
        mid=(l+r+1)/2;
        if (ok(mid)) l=mid;
        else r=mid-1;
    }
    printf("%d", l);
    return 0;
}
```

3. Các bài tập về xâu

Bài tập 1: TÌM SỐ (Đề thi HSG lớp 9 năm 2015)

Để giúp các em học sinh có kỹ năng lập trình tốt chuẩn bị cho kì thi học sinh giỏi tin học, thầy giáo cho các em bài tập sau:

Cho một xâu chỉ gồm các ký tự chữ cái thường trong bảng chữ cái tiếng anh và các chữ số từ 0 đến 9. Đoạn các ký tự số liên tiếp tạo thành một số nguyên. Ở mỗi đoạn ký tự số liên tiếp phải trích ra số lớn nhất có thể, mỗi số lấy ra không có các số 0 không có nghĩa.

Ví dụ, với xâu là **05aab21bc3956cde488a** các số được trích ra là **5, 21, 3956, 488**.

Yêu cầu: Cho xâu *S* có độ dài không quá 100000 kí tự chỉ gồm các kí tự chữ cái thường và chữ số. Hãy viết chương trình tìm số bé nhất và lớn nhất trong các số được trích ra?

Dữ liệu vào cho trong tệp văn bản **TIMSO.INP** gồm một xâu *S* chỉ chứa các ký tự chữ cái thường và chữ số.

Kết quả đưa ra tệp văn bản **TIMSO.OUT**

- Dòng 1 đưa ra số bé nhất tìm được.
- Dòng 2 đưa ra số lớn nhất tìm được.

Ví dụ:

TIMSO.INP	TIMSO.OUT
05aab21bc3956cde488a	5 3956

Hướng dẫn giải thuật:

- Lưu dữ liệu vào xâu *S*.
- Trích lần lượt các số *X* từ trong *S* (lưu ý *X* là số rất lớn).
- Với mỗi số *X* lấy được so sánh để tìm được số lớn nhất và bé nhất.

Chương trình:

```
#include <bits/stdc++.h>

using namespace std;
string s,maxs="",mins="",x;
bool bigger(string a, string b)
{
    if (a.size()==b.size()) return (a>b);
    return (a.size())>b.size());
}
int main()
{
    freopen("timso.inp","r",stdin);
    freopen("timso.out","w",stdout);
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin>>s;
    for(unsigned int i=0;i<s.size();i++)
        if (s[i]>='0' && s[i]<='9')
            x+=s[i];
        else
        {
            if (x.size())>0)
            {
                while (x.size())>1 && x[0]=='0')
                    x.erase(0,1);
```

```

        if (bigger(x,maxs)|| maxs=="") maxs=x;
        if (!bigger(x,mins)|| mins=="") mins=x;
        x="";
    }
}
cout<<mins<<'\n'<<maxs;
return 0;
}

```

Bài tập 2: TRÒ CHƠI (Đề thi HSG lớp 11 năm 2015)

An và Bình chơi trò chơi xếp chữ. Từ một xâu nguồn độ dài n chỉ chứa các ký tự la tinh thường hai người lần lượt đi. Đến lượt ai người đó chọn một ký tự hiện có trong xâu ghi vào cuối xâu riêng của mình và xóa ký tự được chọn trong xâu nguồn. Trò chơi kết thúc khi xâu nguồn rỗng. Ban đầu xâu riêng của 2 người đều rỗng. Từ của ai có thứ tự từ điển nhỏ hơn là người ấy thắng. Nếu hai xâu riêng giống nhau thì coi là cả hai cùng thua. An là người đi trước.

An chơi tốt hơn và thường thắng, vì vậy muốn bạn mình không chán An quyết định luôn luôn chỉ chọn ký tự cuối của xâu nguồn. Bình nhanh chóng nhận ra chiến lược chơi của An và cố gắng tìm cách giành chiến thắng.

Yêu cầu: Cho n ($2 \leq n \leq 10^5$) và xâu nguồn. Hãy xác định xem Bình có thắng được hay không và đưa ra câu trả lời **YES** hoặc **NO** tương ứng. Trong mọi trường hợp – đưa ra từ riêng mà Bình nhận được khi chơi theo chiến lược tối ưu.

Dữ liệu: Vào từ file văn bản **GAME.INP**:

- Dòng đầu tiên chứa số nguyên n ,
- Dòng thứ 2 chứa xâu độ dài n chỉ chứa các ký tự la tinh thường.

Kết quả: Đưa ra file văn bản **GAME.OUT**:

- Dòng đầu tiên chứa thông báo **YES** hoặc **NO**,
- Dòng thứ 2 chứa xâu riêng của Bình.

Ví dụ:

GAME.INP	GAME.OUT
8 cokolada	YES acko

Hướng dẫn giải thuật:

Chiến thuật chơi tối ưu của Bình là luôn chọn ký tự có thứ tự từ điển nhỏ nhất trong xâu còn lại. Nếu có nhiều ký tự giống nhau có thứ tự từ điển nhỏ nhất thì phải chọn ký tự có thứ tự từ điển nhỏ nhất có vị trí lớn nhất trong xâu.

Chương trình:

```

#include <bits/stdc++.h>
using namespace std;
string s, an, binh;
int i, j, n;
stack <int> a[26];
bool kt[100005];
int main()
{
    freopen("game.inp", "r", stdin);
    freopen("game.out", "w", stdout);
    cin>>n;
    cin>>s;
    for (i=0; i<n; i++)
    {
        a[s[i]-'a'].push(i);
    } i=n-1;
    while (i>=0)
    if (!kt[i])
    {
        kt[i]=true;
        an=an+s[i];
        a[s[i]-'a'].pop();
        for (j=0; j<=25; j++)
            if (a[j].size() !=0)
            {
                kt[a[j].top()]=true;
                binh=binh+s[a[j].top()];
                a[j].pop(); break;
            }
        i--;
    }
    else i--;
    if (an>binh) cout<<"YES"; else cout<<"NO";
    cout<<'\\n'<<binh;
    return 0;
}

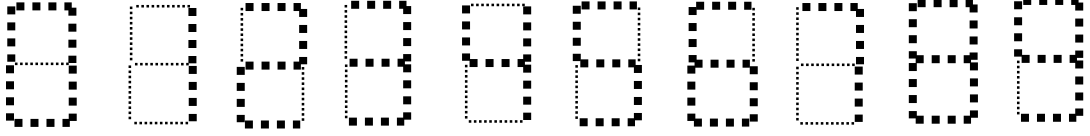
```

4. Các bài tập sử dụng mảng tính trước kết quả.

Những bài toán sử dụng mảng tính trước thường được dùng để giảm thời gian chạy của chương trình.

Bài tập 1: Bảng số điện tử (Đề thi HSG tỉnh lớp 12 năm 2012-2013)

Trong một số hoạt động xã hội, người ta thường dùng một bảng điện tử hiển thị các số tự nhiên liên tiếp, mỗi số hiện thị trong một giây sau đó sẽ đổi sang số tiếp theo. Mỗi chữ số được hiển thị bằng một số đoạn bóng đèn nhỏ, số đoạn bóng đèn cần bật sáng của mỗi chữ số được cho trong bảng dưới đây:

Chữ số	0	1	2	3	4	5	6	7	8	9
Số đoạn bóng đèn	6	2	5	5	4	5	6	3	7	6
										

Ví dụ: khi hiện số 19 cần dùng $(2+6)=8$ đoạn bóng đèn, số 7 cần dùng 3 đoạn bóng đèn.

Yêu cầu: Cho biết hai số S và T là số đầu tiên và kết thúc cần thể hiện trên bảng số điện tử. Hãy xác định lượng điện tiêu thụ W cần thiết để hiển thị các số tự nhiên liên tiếp từ S đến T . Biết mỗi đoạn bóng đèn sáng tiêu thụ hết 1 đơn vị điện năng trong một giây.

Dữ liệu vào: Từ tệp văn bản DEN.INP chứa nhiều dòng, mỗi dòng chứa hai số S, T . (S, T nguyên; $0 \leq S < T \leq 10000$); số dòng không quá 10000.

Dữ liệu ra: Đưa ra tệp văn bản DEN.OUT, chứa nhiều dòng, mỗi dòng là một số W là năng lượng cần xác định theo yêu cầu.

Ví dụ:

DEN.INP	DEN.OUT
8 12	32
9 11	18

Hướng dẫn giải thuật:

Đối với bài toán này, khi học sinh không có kỹ năng sử dụng kỹ thuật mảng tính trước thì giải thuật sử dụng cho bài toán là làm tuần tự. Kết quả là thời gian chạy chương trình rất chậm vì phải tính đi tính lại nhiều lần công suất của các bóng đèn khi hiển thị các số từ S đến T .

Nếu ta sử dụng mảng tính trước để tính kết quả thì chương trình chạy nhanh và hiệu quả. Cụ thể, ta gọi $F[i]$ là tổng lượng tiêu thụ điện của các bóng đèn khi hiện các số từ 1 đến i . Khi đó, $F[i] = F[i - 1] + CS[i]$ (trong đó, $CS[i]$ là công suất tiêu thụ các bóng đèn hiện chữ số i ($0 \leq i \leq 9$)). Như vậy, để tính công suất tiêu thụ các bóng đèn khi hiển thị các số từ S đến T được tính bằng $F[T] - F[S - 1]$

Chương trình:

```
#include <bits/stdc++.h>

using namespace std;
const int power[10]={6,2,5,5,4,5,6,3,7,6};
int f[10005],s,t;
int DEN(int a)
{
    int s=0;
```

```

while (a)
{
    s+=power[a%10];
    a/=10;
}
return s;
}
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    f[0]=6;
    for(int i=1;i<=10000;i++)
        f[i]=f[i-1]+DEN(i);
    freopen("DEN.INP","r",stdin);
    freopen("DEN.OUT","w",stdout);
    while(cin>>s>>t)
    {
        cout<<f[t]-f[s]+DEN(s)<<"\n";
    }
    return 0;
}

```

Bài tập 2: PHÂN TÍCH SỐ (Đề thi HSG lớp 9 năm 2014-2015)

Nhà toán học Goldbach đưa ra giả thuyết như sau: “Mỗi số nguyên lớn hơn 3 có thể biểu diễn thành tổng của hai số nguyên tố”. Từ giả thuyết này, ta có phát biểu mới: “Mỗi số nguyên lớn hơn 11 đều có thể biểu diễn thành tổng của hai hợp số”.

Ví dụ, số $15=6+9$, 6 và 9 là hợp số.

Yêu cầu: Bạn hãy viết chương trình đếm số cách phân tích số n thành tổng hai số nguyên a và b ($1 < a \leq b$ và a, b là hợp số)?

Chú ý: Một số tự nhiên m được gọi là hợp số nếu có nhiều hơn hai ước.

Dữ liệu vào cho trong tệp văn bản **PT.INP** gồm một số nguyên n ($12 \leq n \leq 10^7$).

Kết quả đưa ra tệp văn bản **PT.OUT** một số duy nhất là số cách phân tích tìm được.

Ví dụ:

PT.INP	PT.OUT
15	1

Hướng dẫn giải:

Giải thuật tự nhiên của bài này là duyệt lần lượt số a từ 2 ... $n \div 2$. Với mỗi số a ta kiểm tra xem a và $n - a$ có cùng là hợp số hay không, nếu thỏa mãn ta tìm được một cặp số thỏa mãn.

Vấn đề đặt ra ở đây là làm thế nào để kiểm tra một số a có phải là hợp số hay không? Có một cách đơn giản ở đây là ta sử dụng hàm kiểm tra nguyên tố. Nếu số

a ($a > 1$) không phải là số nguyên tố thì sẽ là hợp số. Vậy thuật toán giải quyết bài toán trên có thể viết đơn giản như sau:

```
int d=0;
for(int i=2;i<= n/2;i++)
    if (!NT(i) && !NT(n-i))
        d++;
//NT(a) là hàm kiểm tra nguyên tố
```

Ta lại thấy hàm NT(a) trong thuật toán trên được gọi thực hiện rất nhiều lần và làm tăng thời gian chạy bài toán khi n lớn.

Có một cách hiệu quả hơn để giải quyết vấn đề trên là sử dụng mảng tính trước $P[i]$ có giá trị *TRUE* nếu i là số nguyên tố và là *FALSE* nếu i không là số nguyên tố. Để tính mảng P ta sử dụng thuật toán sàng số nguyên tố. Thuật toán có thể viết lại đơn giản như sau:

```
int d=0;
for(int i=2;i<= n/2;i++)
    if (!p[i] && !p[n-i])
        d++;
```

Như vậy thuật toán chạy nhanh hơn khi giá trị n lớn vì không phải gọi hàm kiểm tra nguyên tố nhiều lần.

Chương trình:

```
#include <bits/stdc++.h>
using namespace std;
bool prime[10000007];
int main()
{
    memset(prime,true,sizeof(prime));
    for(int i=2;i*i<=10000000;i++)
        if (prime[i])
            for(int j=2;j*i<=10000000;j++)
                prime[i*j]=false;
    freopen("PT.INP","r",stdin);
    freopen("PT.OUT","w",stdout);
    int n;
    cin>>n;
    int d=0;
    for(int i=2;i<= n/2;i++)
        if (!prime[i] && !prime[n-i])
            d++;
    cout<<d;
    return 0;
}
```

Bài tập 3: HÌNH VUÔNG (Đề thi HSG lớp 9 năm 2014-2015)

Cho một hình chữ nhật gồm m dòng và n cột. Các dòng của bảng được đánh số từ 1 đến m , từ trên xuống dưới, các cột được đánh số từ 1 đến n , từ trái sang phải. Ô nằm trên giao của dòng i và cột j là ô (i, j) , chứa số nguyên có giá trị 0 hoặc 1 .

Yêu cầu: Hãy viết chương trình tìm diện tích lớn nhất của một hình vuông trong hình chữ nhật trên mà các ô trong hình vuông đó toàn bằng 1 ?

Chú ý: Diện tích hình vuông là số ô trong hình vuông đó.

Dữ liệu vào cho trong tệp văn bản **HV.INP**

- Dòng đầu tiên chứa hai số nguyên m, n ($0 < m, n \leq 1000$).
- m dòng tiếp theo, dòng thứ i chứa n số 0 hoặc 1 . Các số cách nhau ít nhất một dấu cách.

Kết quả đưa ra tệp văn bản **HV.OUT** gồm một số duy nhất là diện tích hình vuông lớn nhất tìm được.

HV.INP	HV.OUT
3 4 0 0 1 1 0 1 1 1 1 0 1 0	4

Hướng dẫn thuật toán:

Giải thuật tự nhiên là vét cạn: xét tất cả các hình vuông, tìm hình vuông lớn nhất thỏa mãn:

```

Kq=0;
For (int i=1; i<=m; i++)
    For (int j=1; j<= n; j++)
        For (int k=1; k<=min(m-i+1, n-j+1))
            If (Tong(i, j, k)=k*k)
                Kq= max(kq, k*k);

```

Hàm Tong(i,j,k): tổng các số trong hình vuông có góc trên trái là (i,j) và góc dưới phải (i+k-1, j+k-1) dùng hai vòng lặp để tính.

Để tăng hiệu quả chương trình, ta sử dụng mảng tính trước

- Gọi $F[i, j]$ là tổng các số từ ô (1,1) đến ô (i,j):

$$F[i, j] = F[i-1, j] + F[i, j-1] - F[i-1, j-1] + a[i, j];$$
- $Tong(i, j, k) = F[i+k-1, j+k-1] - F[i+k-1, j-1] - F[i-1, j+k-1] + F[i-1, j-1]$

Chương trình:

```

#include <bits/stdc++.h>
#define fort(i,a,b) for(int i=(a); i<=(b); i++)
using namespace std;
const int MAXN=1005;

```

```

int n, m, a[MAXN][MAXN], f[MAXN][MAXN];
int sum(int dd,int cd,int k)
{
    int dc=dd+k-1, cc=cd+k-1;
    return f[dc][cc]-f[dd-1][cc]-f[dc][cd-1]+f[dd-1][cd-1];
}
bool ok(int x, int y,int mid)
{
    return (sum(x,y,mid)==mid*mid);
}
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    freopen("HV.INP","r",stdin);
    freopen("HV.OUT","w",stdout);
    cin>>n>>m;
    for(int i=1,n) for(int j=1,m) cin>>a[i][j];
    for(int i=1,n) for(int j=1,m) f[i][j]=f[i-1][j]+f[i][j-1]-f[i-1][j-1]+a[i][j];
    int ans=0;
    for(int i=1,n) for(int j=1,m)
    {
        int maxd=min(n-i+1,m-j+1);
        int d=1, c=maxd;
        while (d<c)
        {
            int mid=(d+c+1)/2;
            if (ok(i,j,mid)) d=mid;
            else c=mid-1;
        }
        if (ok(i,j,d) && ans<d*d) ans=d*d;
    }
    cout<<ans;
    return 0;
}

```

C2. Giới thiệu một số thư viện C++

Như chúng ta đã biết, một trong những điểm mạnh của ngôn ngữ lập trình C++ là hệ thống thư viện rất phong phú và dễ sử dụng. Sau đây tôi sẽ giới thiệu một số thư viện thông dụng và dễ dùng, đáp ứng được yêu cầu trong giải các bài tập phục vụ cho việc bồi dưỡng học sinh giỏi tin học.

1. Thư viện Math

- Để sử dụng các hàm có sẵn trong thư viện này ta phải khai báo ở phần đầu chương trình như sau :

#include<cmath>

- Một số hàm hay sử dụng trong thư viện math

- Hàm abs(x): Trả về giá trị tuyệt đối của số x (số thực hoặc số nguyên).
- Hàm pow(x,y): Trả về giá trị x^y (x, y là số thực hoặc số nguyên)
- Hàm round(x): Làm tròn số x, trả về giá trị nguyên gần số x.
- Hàm sqrt(x): Trả về căn bậc hai của x.
- Ngoài ra còn một số hàm lượng giác và các hàm khác nhưng chúng tôi không đề cập thêm, nếu cần tham khảo thêm các hàm trong thư viện math xin truy cập vào địa chỉ cplusplus.com

2. Thư viện algorithm

- Để sử dụng các hàm có sẵn trong thư viện này ta phải khai báo ở phần đầu chương trình như sau :

#include<algorithm>

- Một số hàm hay sử dụng trong thư viện algorithm

- Hàm min(x,y): trả về số bé nhất trong hai số.
- Hàm max(x,y): trả về số lớn nhất trong hai số.
- Hàm swap(x,y): trao đổi hai giá trị x, y cho nhau
- Hàm count(first, last, val): trả về số giá trị bằng val trong [first,last)

Ví dụ:

```
int a[5]={ 1,4,3,2,3};
```

```
cout<<count(a,a+5,3);//trả về 2 là số giá trị bằng 3 trong mảng a
```

- Hàm *min_element(first,last): trả về giá trị nhỏ nhất trong [first, last)

Ví dụ:

```
int a[5]={ 1,4,3,-5,3};
```

```
cout<<*min_element(a,a+5);//trả về -5 là giá trị nhỏ nhất trong mảng a
```

- Hàm *max_element(first,last): trả về giá trị lớn nhất trong [first, last)

Ví dụ:

```
int a[5]={ 1,4,3,-5,3};
```

```
cout<<*max_element(a,a+5);//trả về 4 là giá trị lớn nhất trong mảng a
```

- Hàm sort(first, last): sắp xếp các phần tử theo thứ tự tăng dần trong [first, last)

Ví dụ:

```
int a[5]={ 1,4,3,2,3};
```

```
sort(a,a+5);
```

```
for(int i=0;i<5;i++) cout<<a[i]<<' ';//Day sx: 1 2 3 3 4
```

- Hàm sort(first, last, cmp): Sắp xếp các phần tử trong [first, last) theo điều kiện cmp. Trong đó, cmp là một hàm logic trả về true nếu hai phần tử cần so sánh đúng thứ tự, ngược lại trả về false.

Ví dụ:

```

bool cmp(int x, int y)
{
    return (x>y);
}

int main()
{
    int a[5]={ 1,4,3,2,3};
    sort(a,a+5,cmp);
    for(int i=0;i<5;i++) cout<<a[i]<<' ';//day sx: 4 3 3 2 1
    return 0;
}

```

- Hàm `lower_bound(first, last, val)`: Trả về con trỏ đến phần tử đầu tiên trong `[first, last)` có giá trị $\geq val$, nếu không có phần tử thỏa mãn hàm trả về con trỏ `last`.

Chú ý:

- Hàm `lower_bound` thực hiện đúng khi dãy các đối tượng từ con trỏ `first` đến `last` đã được sắp xếp theo thứ tự không giảm.
- Con trỏ `first` là biến đặc biệt dùng chứa địa chỉ phần tử đầu tiên, `last` là con trỏ chứa địa chỉ sau phần tử cuối.

Ví dụ:

```

int main()
{
    int a[8]={ 1,4,3,2,3,6,7,2};
    sort(a,a+8);//1, 2, 2, 3, 3, 4, 6, 7
    int vt=lower_bound(a,a+8,3)-a;
    cout<<vt;//tra ve vi tri thu 3 cua mang, mang bat dau tu vi tri 0
    return 0;
}

```

- Hàm `lower_bound(first, last, val,cmp)`: Trả về con trỏ đến phần tử đầu tiên trong `[first, last)` có giá trị $= val$ hoặc nếu không bằng `val` trả về con trỏ sau `val` theo điều kiện sắp xếp của hàm `cmp`. nếu không thỏa hai điều kiện trên trả về con trỏ `last`.

Ví dụ:

```

bool cmp(int x, int y)
{
    return (x>y);
}

```

```

    }
    int main()
    {
        int a[8]={ 1,1,3,2,3,6,7,2};
        sort(a,a+8,cmp);
        int vt=lower_bound(a,a+8,3,cmp)-a;
        cout<<vt;//tra ve vi tri thu 2 cua mang, mang bat dau tu vi tri 0
        return 0;
    }

```

Chú ý: Trong chương trình trên, hàm `cmp` dùng để sắp xếp dãy `a` theo thứ tự không tăng. Hàm `lower_bound` trả về vị trí của phần tử đầu tiên nhỏ hơn hoặc bằng 3.

- Hàm `upper_bound(first, last, val)`: Trả về con trỏ đến phần tử đầu tiên trong `[first, last)` có giá trị $> val$, nếu không có phần tử thỏa mãn hàm trả về con trỏ `last`.

Chú ý:

- Hàm `upper_bound` thực hiện đúng khi dãy các đối tượng từ con trỏ `first` đến `last` đã được sắp xếp theo thứ tự không giảm.
- Con trỏ `first` là biến đặc biệt dùng chứa địa chỉ phần tử đầu tiên, `last` là con trỏ chứa địa chỉ sau phần tử cuối.

Ví dụ:

```

    int main()
    {
        int a[8]={ 1,4,3,2,3,6,7,2};
        sort(a,a+8);//1, 2, 2, 3, 3, 4, 6, 7
        int vt=upper_bound(a,a+8,3)-a;
        cout<<vt;//tra ve vi tri thu 5 cua mang, mang bat dau tu vi tri 0
        return 0;
    }

```

- Hàm `upper_bound(first, last, val, cmp)`: Trả về con trỏ đến phần tử đầu tiên trong `[first, last)` có giá trị sau `val` được sắp xếp theo điều kiện hàm `cmp`, nếu không có phần tử thỏa mãn hàm trả về con trỏ `last`.

Ví dụ:

```

    bool cmp(int x, int y)
    {
        return (x>y);
    }

```

```

int main()
{
    int a[8]={ 1,1,3,2,3,6,7,2};
    sort(a,a+8,cmp);
    int vt=upper_bound(a,a+8,3,cmp)-a;
    cout<<vt;//tra ve vi tri thu 4 cua mang, mang bat dau tu vi tri 0
    return 0;
}

```

Chú ý: Trong chương trình trên, hàm cmp dùng để sắp xếp dãy a theo thứ tự không tăng. Hàm upper_bound trả về vị trí của phần tử đầu tiên nhỏ hơn 3.

D. Kết luận

Trên đây là một số kinh nghiệm chúng tôi đã được rút ra trong quá trình dạy học sinh giỏi Tin học trong những năm qua. Các bài tập đã được cài đặt hiệu quả trên ngôn ngữ C++.

Kinh nghiệm cho thấy, khi học sinh chuyển từ ngôn ngữ lập trình Pascal sang ngôn ngữ C++ không gặp khó khăn gì, khi các em đã quen với ngôn ngữ C++ thấy được rất nhiều tiện lợi và tỏ ra thích thú.