

Xin chào các bạn , mình là một học sinh chuyên tin khóa 72 THPT CNH . Qua 2 năm theo đội tuyển tin , mình đã gặp nhiều vấn đề không chỉ ở code mà còn các mặt khác nữa . Sau đây là những khó khăn mình gặp phải và cách giải quyết chúng , hy vọng nó có thể giúp ích cho các bạn theo học đội tuyển tin .

1 >> vào chuyên tin thì làm gì đầu tiên :) ?

đầu tiên các bạn cần tải codeblock (ide giúp các bạn code) :









<https://www.codeblocks.org/downloads/binaries/>

(mình hay chọn tải cái “ codeblocks-20.03mingw-setup.exe”) và themis (để nếu các bạn có test của thầy cô thì các bạn có thể về nhà tự chấm , cách dùng themis các bạn có thể xem trong phần hướng dẫn khi bật themis lên) :

<https://dsapblog.wordpress.com/2013/12/24/themis/>

(Cái này thì các bạn chọn “bản đầy đủ “)

Ngoài ra , khi tải test các bạn có thể sẽ gặp trường hợp này :

	0.ans
	0.in
	1.ans
	1.in
	10.ans
	10.in
	11.ans
	11.in
	12.ans
	12.in
	13.ans
	13.in

Rõ ràng thế này thì khi tải về không thể cho vào themis chấm ngay được và các bạn cần đưa nó về dạng chuẩn : file ghi tên bài cần chấm → testXX (XX là số thứ tự của test , 00 → 99 chẳng hạn) → file tên bài.inp và file tên bài.out

Lúc này mà các bạn ngồi làm thủ công thì rất tốn sức và thời gian (do có thể có rất nhiều test) , lúc này các bạn cần tải cái này : <https://sites.google.com/site/kc97bla/test-formatter>

Nói chung là cái này giúp các bạn chuyển test về dạng chuẩn , cách dùng các bạn có thể lên mạng là thấy .

Với các bài đồ thị thì các bạn có thể lên trang : https://csacademy.com/app/graph_editor/

Trang này sẽ giúp vẽ đồ thị cho các bạn trong khi các bạn chỉ cần nhập dữ liệu đồ thị vào là xong (nhập các cạnh hoặc các cạnh và trọng số của nó)

2 >> những thắc mắc về comment và define

define đơn giản là bạn định dạng lại những “từ” mà bạn có thể lặp lại nhiều trong code thành 1 bản ngắn gọn hơn để dễ code hơn và nhanh hơn

ví dụ : `#define pb push_back`

thì bây giờ thay vì phải gõ `v.push_back()` thì bạn chỉ cần gõ `v.pb()`

comment để ta note những ý tưởng của thuật toán hoặc đơn giản là bất kì kí tự nào mà nếu không comment chương trình sẽ chạy sinh lỗi :v

mình xin đi thẳng vào vấn đề luôn :D , có những kiểu comment sau

```
/*  
    loại này có thể comment nhiều dòng  
*/  
  
//    loại comment này chỉ được trên một dòng  
  
///    loại comment này cũng trên một dòng nhưng mà nó sẽ nổi bật lên , không phải màu xám cho các bạn dễ nhìn
```

3 >> kinh nghiệm debug

debug là một phần mà hầu như gặp bài nào các bạn cũng sẽ phải làm , bởi chương trình chúng ta code ra có thể có những lỗi tiềm ẩn mà chúng ta chưa nhìn ra ngay được . Theo mình thấy thì đầu tiên khi code xong thì các bạn nên xem lại một lượt code của mình , những lỗi căn bản nhất có thể thấy bằng mắt là sai giới hạn mảng , khai báo mảng trong hàm main , chưa xóa debug (nhớ xóa hoặc comment trước khi nộp bài) , đặt tên biến trùng tên hàm hoặc tên mảng , sai tên đề bài , sai output format , ... và lưu ý là nếu tên các bạn đặt cho hàm hay mảng mà chữ nó đổi màu (không phải đen bình thường) thì các bạn nên đổi để tránh chạy sinh lỗi không đáng có . Sau khi nhìn lại một lượt như thế thì có thể đảm bảo rằng các bạn đã code được đúng suy nghĩ của mình rồi . Sau đó nếu chạy thử các test mà ra sai thì có 2 khả năng : các bạn sai về ý tưởng làm bài hoặc code-có-về-đúng-nhưng-mà-lại-sai. Theo mình thì các bạn nên suy nghĩ lại xem ý tưởng của mình đã chuẩn chưa và sinh vài test tay ra để check (nên làm cái này ngay từ đầu). Sau khi đảm bảo mình nghĩ chuẩn rồi thì bây giờ chỉ cần sửa code là xong . Ở phần này thì các anh khóa trước của mình và cả lứa của mình đều dùng cách cout ra (mặc dù trên mạng người ta có cái công cụ gì đó trên codeblock giúp việc này nhưng mà mình thấy cout ra nó tiện hơn :v) . Các bạn nên cout ra các chỗ mà các bạn nghi ngờ (ví dụ một giá trị sau khi thực hiện các lệnh của bạn có biến đổi đúng với suy nghĩ của bạn hay không , hoặc các bạn kiểm tra xem chỗ gọi chỉ số mảng chẳng hạn (có bị âm hoặc quá giới hạn không) , hoặc kiểm tra hàm nào đó chạy chuẩn không , ...) . Và để debug hiệu quả thì các bạn cần sinh các test ,

mình thấy các bạn nên sinh test tay bởi khi sinh tay thì các bạn có thể sinh các test hiếm hoặc các trường hợp đặc biệt mà các bạn có thể nghĩ ra , mặt khác khi debug xong thì các bạn cũng nên sinh các test max giới hạn để xem code mình có bị tràn hay không . Còn nếu mà các bạn muốn sinh test random thì sau đây là cách để sinh random :)

```
srand(time(0)) // nhớ phải có dòng này nha  
val = a + rand() % b
```

lúc này val sẽ là giá trị $a + (\text{một số bất kì từ } 0 \text{ đến } b - 1)$

Khi đi thi thì các bạn nên code thêm một code trâu nữa (hoặc dùng code sub nhỏ) để check với sub lớn , hoặc trong trường hợp code sub lớn sai thì các bạn có thể dùng code trâu (code sub nhỏ) để nộp được để vét điểm (bởi nếu rơi vãi điểm sẽ hối hận lắm đó :()

4 >> các lưu ý khi code

- + đọc nhanh thì code các bạn nên luôn có :
`ios_base::sync_with_stdio(false);`
`cin.tie(0); cout.tie(0);`
- + `freopen` (cái này khi đi thi hoặc chấm bằng themis thì phải có)

do nếu chỉ `freopen` không thì không chạy được nên có nhiều trường hợp comment `freopen` để test code rồi khi nộp quên bỏ cái comment ra , rồi bị 0 điểm . Để khắc phục thì các bạn nên làm như sau :

```
if(ifstream("tên bài.inp")){  
    freopen("tên bài.inp", "r", stdin) ;  
    freopen("tên bài.out", "w", stdout) ;  
}
```

thế này thì các bạn có thể chạy thoải mái mà không cần comment , nộp được luôn (ncl đã năng) .

- + khai báo mảng ngoài hàm `main`
(để trong có thể bị chạy sinh lỗi , đây là mảng , còn các `map` , `set` , `vector` , ... thì các bạn để trong hàm thoải mái)

- + cách đặt tên hàm , tên mảng , tên biến (đã nói ở trên)
- + code sub thì các bạn nên code theo namespace : <https://pastebin.com/RqHZxU3E>

5 >> một chút động lực và lý do nên theo đội tuyển

khi theo học đội tuyển và thi hsg , các bạn sẽ có nhiều cơ hội hơn vào đại học , có thể thi tối đa 2 lần (lớp 11 , 12) , không những thế có giải thành phố thôi là được xét tuyển thẳng rồi . Hơn nữa , theo đuổi đội tuyển cũng có thể coi là một cái gì đó lớn lao đáng để các bạn theo đuổi trong thời gian học cấp 3 của mình . Và trong thời gian học đội tuyển , các bạn có thể có những trải nghiệm mới , được học những người thầy giỏi hay được học cùng những anh

chị khóa trên . Và các bạn cũng có thể có cơ hội được đi trại hè tin học , một nơi mà có thể được học cùng các bạn từ tỉnh khác , trường khác trong suốt 1 tuần . Và thỉnh thoảng , mình hay comment thời gian mình code bài nào đó vào code bài đó để có thể nhìn lại được quá trình của mình .

6 >> các kì thi các bạn có thể sẽ được tham gia

- + hsg trường
- + thi hsg thành phố (v1 lấy giải , v2 chọn ra top15 đại diện cho Hà Nội đi thi hsg qg)
- + thi hsg qg
- + kì thi hsg quốc tế , châu á , ...
- + kì thi duyên hải và đồng bằng bắc bộ
- + kì thi tin học trẻ thành phố
- + kì thi tin học trẻ quốc gia
- + vnu oi
- + free contest (cái này trên mạng và các bạn có thể tham gia thoải mái)
- + các kì thi trên các trang luyện tập : codeforces , ...
- + icpc (có vòng dành cho cấp 3)

7 >> nên bắt đầu học ở đâu

theo mình đầu tiên các bạn nên học ở các trang vnoi wiki , cp-algorithms . Ngoài ra , có một số sách khá là hay như tài liệu giáo khoa chuyên tin , giải thuật lập trình (thầy Lê Minh Hoàng) . Nếu đọc sách thì các bạn nên đọc ý tưởng thôi , chứ code thì nên tìm trên mạng . Mình thì thấy nên học theo chuyên đề trước để trang bị các công cụ thì khi làm đề nó sẽ dễ dàng hơn . Lộ trình học thì các bạn cứ học theo các trang mình nói ở trên , dưới đây là một số code mẫu của mình (nếu bạn nào cần thì có thể tham khảo) :

- stl : <https://pastebin.com/kJpaLZ59>
các bạn tìm hiểu kĩ hơn trên vnoi wiki (có 1 file nói về stl)
- bitset :
https://pastebin.com/cMWNu0n0?fbclid=IwAR3xTqh85HR1z1bx7s0yrveOwhBNv3K4ZnQ0zALtb3s_CBwEJFRVwHnR78U

đây là những gì mình biết về bitset , các bạn có thể tìm hiểu thêm trên mạng

- toán : <https://pastebin.com/GNUWCEAD>
<https://www.hackerearth.com/practice/notes/powerful-tricks-with-calculation-modulo/>
các bạn có thể tìm hiểu thêm định lý lucas , định lý thặng dư trung hoa , ...
- string : <https://pastebin.com/c1s2gy5Q>
trie : <https://pastebin.com/ZH1vnpwQ>

+ mình hay tính giới hạn mảng của trie là độ dài tối đa * n

hash : các bạn dùng 1 mod bình thường có thể không full thì có thể dùng pair với 2

mod khác nhau

- bignum : <https://pastebin.com/TnBk0wHe>
- lca : <https://pastebin.com/AuF7kJZn>
- hld : <https://pastebin.com/Wpgub0DK>
- euler tour : <https://pastebin.com/g7Vs0s07>
- centroid: <https://pastebin.com/JbQpanEa>

- dsu : <https://pastebin.com/XM2bjjrX>
ngoài ra còn một cách code dsu nữa , cách này “linh hoạt” hơn cách trên bởi ta có thể “trở lại” một trạng thái trong quá khứ , chẳng hạn ta thực hiện một số phép “gộp” và ta muốn cho nó quay trở lại một số bước (hay “gỡ đi” một số phép gộp trước đó)
<https://pastebin.com/QKpStTwP>

8 >> các kiến thức , dạng bài các bạn nên biết

- trong dạng bài mà các bạn cần xử lý nhiều truy vấn , các bạn giải theo phong cách duyệt hay đệ quy gì đó , nếu đến tầng cuối cùng (ta sẽ ngắt hàm duyệt tại đó) . Lúc này nếu dùng “return” thì các bạn chỉ ngắt được tầng cuối của hàm duyệt thôi , hay nói cách khác , các nhánh khác ở các tầng bên trên vẫn có thể duyệt tiếp . Nếu dùng “exit(0)” thì nó sẽ ngắt cả chương trình , không được vì ta cần xử lý nhiều truy vấn mà .
Lúc này , giải pháp là ta dùng câu lệnh “try , catch , throw “
→ link code kèm giải thích : <https://pastebin.com/FW4pF7SY>
- chặt nhị phân trên cây IT : đặc trưng dạng này là với vị trí R thì tìm vị trí $L \leq R$ mà L nhỏ nhất hay lớn nhất thỏa mãn tính chất nào đó hoặc với L tìm R $\geq L$ mà R nhỏ nhất hay lớn nhất thỏa mãn tính chất nào đó
ví dụ : với mỗi R tìm L $\leq R$ mà L lớn nhất thỏa mãn $a(L) < a(R)$

bài điển hình dạng này là <https://vn.spoj.com/problems/BALLGAME/>

các bạn có thể tham khảo code mình là làm được dạng này :

<https://ideone.com/ZoQLTg>

- dijkstra nhiều chiều : nghe có vẻ hơi lạ nhưng đó là dạng bài mà tìm đường đi thỏa mãn “tính chất nào đó” mà tối ưu nhất , lúc này ta chỉ việc gọi thêm chiều khác để quản lý trạng thái là xong .
Một số bài dạng này : roads spoj , centre28 spoj (bài này cần làm bài qbschool trước) , netaccel spoj , revamp spoj ,
<https://codeforces.com/group/FLVn1Sc504/contest/308633/problem/A> (tham gia group vnoi trên codeforces) , ...
- chặt nhị phân song song : nói đơn giản là thay vì chặt nhị phân cho từng truy vấn thì các bạn sẽ chặt nhị phân cho tất cả các truy vấn cùng lúc luôn , phương pháp này giúp giảm độ phức tạp .
code mẫu : <https://ideone.com/FmN1pA>
đọc thêm tại : <https://codeforces.com/blog/entry/45578>
- chia căn : những bài dạng này thường khá khó và chả có quy luật gì , chủ yếu là các bạn cần “sáng tạo” thôi .
- dsu on tree : <http://www.codeforces.com/blog/entry/44351>
- mo on tree: <http://codeforces.com/blog/entry/43230>
- một cách giải đặc biệt của dạng bài tìm vị trí lớn nhất có tổng $\leq x$:
<https://ideone.com/hBKrlY>

9 >> link bài tập với tài liệu hay :

https://drive.google.com/drive/folders/0B_Dd-qODogA6flRmQjh2UV9PbUpwdURlaFI1T3E3RnJFX0xIT2tpbFpvMVJObDJzelJuVlk?fbclid=IwAR3AmSgsd5vSTdtedm11yegrZKEx_o5FAYZIG7EsCPxicbNhJ8kduyMmuWA&resourcekey=0-6l36AKVqZgxs0zo1LjtfTg

https://drive.google.com/drive/folders/1LBcmCf7TEwKJeaIgDRk-BBkHQbkHyR3n?fbclid=IwAR3InMu5xkAjR0MQ4Dz4ZP97_vM3EIM-uyaSWnC-1nrvHQTXC34jPaDuftE

<https://github.com/koosaga/olympiad>

<https://sites.google.com/site/kc50dhv/>

<https://vnoi.info/problems/list/?tag=156&page=1#tag-category-14>

<https://cses.fi/problemset/list/>

<https://vnspoj.github.io/>

<https://onlylove97.wordpress.com/author/onlylove97/page/2/>

<https://hackmd.io/@SPyofgame>

<https://kienthuc24h.com/tong-hop-tai-lieu-chuyen-tin-can-thiet/>

<https://vnoi.info/wiki/Home>

<https://cp-algorithms.com/>

<https://sites.google.com/site/kc97ble/>

10 >> kinh nghiệm thi cử

phần này mình sẽ không trình bày nhiều về cách phân bố thời gian hay chuẩn bị tâm lý gì đó vì cái này tùy từng người và chắc các bạn cũng đã nghe lời khuyên hay đi thi nhiều rồi . Mình chỉ khuyên là các bạn hãy cố gắng code được hết các sub mình nghĩ được , và check được một số test hiểm hay max giới hạn thì mới có thể nộp được . Và khi làm , thay vì nhập test tay và chạy như bình thường , đi thi các bạn nên làm khác một chút , bởi phiên bản các bạn nộp cần phải là phiên bản cuối cùng của các bạn và phiên bản đó “sẵn sàng” để chấm . Vậy làm sao để biết được là phiên bản mình nộp sẽ không bị chạy sinh lỗi khi chấm thì mình sẽ chỉ cho các bạn cách sau . Đó là các bạn tạo file “tên bài.inp” cùng nơi với file code của mình , sau đó các bạn nhập test của mình vào file .inp rồi lưu lại . Sau đó chạy file code , rồi mở file .out là xong . Rất đơn giản phải không :D ? Sau đó khi các bạn check thì chỉ việc nhập test vào file .inp rồi lưu lại rồi chạy file code , rồi mở file .out để xem ok chưa . Như vậy sẽ đảm bảo code mình khi chấm chạy bình thường và nếu file .out có chạy bị sai hay chạy quá lâu thì các bạn sẽ kịp thời sửa code của mình . Làm việc chắc chắn như vậy sẽ giúp tâm lý các bạn ổn hơn rất nhiều khi đi thi (không nơm nớp sợ sai bất thành linh ở đâu :v) . Cuối cùng , thì mình chúc các bạn luôn cháy hết mình với đội tuyển , thi cử gặp nhiều may mắn và làm được hết sức của mình !