I. ĐẶT VẤN ĐỀ

1.1 Lý do chọn đề tài:

Trong quá trình giảng dạy môn Tin học, bản thân tôi luôn trăn trở làm thế nào để học sinh có thể đạt kết quả tốt nhất khi học lập trình. Đặc biệt là mũi nhọn học sinh giỏi.

Trong các nội dung bồi dưỡng học sinh giỏi, một nội dung rất quan trọng khi bồi dưỡng là nguồn bài tập. Bao gồm bài tập cho quá trình học và bài tập khi làm đề luyện tập và kiểm tra đội tuyển. Khi có được bài tập thì phần quan trọng là đánh giá bài làm học sinh đạt yêu cầu hay không. Mà để đánh giá thì với mỗi bài tập ta cần một số lượng Test nhất định.

Trong quá trình bồi dưỡng, khi đưa các bài tập cho các em rèn luyện tôi thường tạo ra một số lượng test phù hợp để đánh giá. Các em sẽ được kiểm tra chương trình của mình đúng hay sai, hiệu quả hay không. Qua đó tôi nhận thấy học sinh rất hứng thú vì có thể kiểm tra được thuật toán của mình đúng hay sai.

Nhưng làm thế nào để tạo ra được một bộ test chất lượng, kiểm tra được hết các trường hợp của thuật toán thì không hề đơn giản. Một số chương trình tập huấn ở Sở và một vài tài liệu khác chỉ dừng ở mức biết cách tạo ra được một bộ test case **ngẫu nhiên** chứ chưa đi sâu vào cách tạo ra một test case chất lượng.

Từ lý do trên, qua quá trình giảng dạy tôi xin trình bày sáng kiến “Kinh nghiệm tạo test cho bài tập lập trình”. Nhằm góp thêm một chút kinh nghiệm cho các bạn đồng nghiệp có thể tham khảo và áp dụng vào công việc của mình.

1.2 Mục đích nghiên cứu

- Nghiên cứu cách tạo ra một bộ test chất lượng cho các bài tập lập trình.

1.3 Đối tượng và khách thể nghiên cứu:

- Đối tượng: Test của bài toán lập trình trong Tin học.

- Khách thể: Học sinh lớp 11 và 12 trường THPT Vĩnh Định.

1.4 Phương pháp nghiên cứu:

- Phân tích tổng kết kinh nghiệm.

1.5 Nhiệm vụ nghiên cứu:

- Nghiên cứu cách tạo được bộ test.

- Nghiên cứu kỹ thuật tạo bộ test chất lượng.

1.6 Phạm vi và kế hoạch nghiên cứu:

- Phạm vi nghiên cứu:

* Chương trình bồi dưỡng HSG – Phần bài tập và kiểm tra.

- Kế hoạch nghiên cứu:

* Đợt 1: từ tháng 9 năm 2013 đến tháng 4 năm 2014.
* Đợt 2: từ tháng 9 năm 2015 đến tháng 4 năm 2016.
* Đợt 3: từ tháng 6 năm 2017 đến tháng 8 năm 2018.
* Đợt 4: từ tháng 8 năm 2018 đến tháng 10 năm 2018.

II. NỘI DUNG SÁNG KIẾN

2.1 Cơ sở lý luận

2.1.1 Bài tập lập trình

Bài tập lập trình là các bài tập được sử dụng trong hầu hết các kỳ thi HSG Tin học các cấp hiện nay. Nội dung yêu cầu học sinh giải quyết một vấn đề nào đó bằng cách viết một chương trình trên máy tính để giải.

Các bài tập lập trình được sử dụng trong các kỳ thi rất đa dạng, thuộc nhiều chủ đề khác nhau. Từ yêu cầu đánh giá kỹ năng viết chương trình đơn giản cho đến đánh giá tư duy trong việc giải quyết các vấn đề phức tạp.

2.1.2 Bộ test chấm là gì

Sau khi học sinh viết chương trình để giải quyết bài tập lập trình trên máy, giám khảo sẽ tiến hành đánh giá và cho điểm chương trình đã viết của học sinh. Chương trình của học sinh sẽ được thực hiện với bộ test chấm.

Một bộ test chấm bao gồm nhiều test. Mỗi test chấm sẽ bao gồm một file dữ liệu vào và một file dữ liệu đáp án. Một số bài toán đa nghiệm thì sẽ có thêm chương trình check cho bộ test.

Một bộ test chấm tốt là một bộ test đánh giá được hết các trường hợp của vấn đề cần giải quyết, đánh giá chương trình của học sinh có giải quyết được vấn đề đặt ra của bài tập hay không. Từ đó giúp cho điểm để phân loại thí sinh.

2.2 Thực trạng hiện tại:

2.2.1 Thuận lợi

Trong các năm học vừa qua, ban giám hiệu nhà trường thường xuyên quan tâm, tạo mọi điều kiện cho Tổ tin học. Nhà trường đầu tư 2 phòng máy với khoảng 20 máy mỗi phòng. Đây là một yếu tố rất thuận lợi, giúp cho thầy và trò trường THPT Vĩnh Định học tốt bộ môn lập trình. Vì lập trình thì không thể thiếu công cụ đó là máy tính.

Nguồn bài tập hiện nay rất phong phú trên internet. Có rất nhiều trang giải bài tập trực tuyến như spoj, NTU, Codeforces… Đây là một điều kiện rất thuận lợi cho việc tìm kiếm nguồn bài tập.

Các bài tập lập trình hiện nay được chấm trên phần mềm Themis được Bộ GD-ĐT sử dụng trong các kỳ thi. Nên rất thuận tiện cho giáo viên chấm. Chỉ cần người ra đề chuẩn bị tốt bộ test chấm.

2.2.2 Khó khăn

Các bài tập lập trình cần một lượng Test nhất định để kiểm tra tính đúng đắn của các thuật toán. Đặc biệt khi chuẩn bị nguồn bài tập cho quá trình học cũng như làm đề ôn luyện và thi cử cần phải có bộ test chất lượng để kiểm tra.Và hiện tại ít có tài liệu hướng dẫn những người mới bắt đầu cách xây dựng một bộ test case chất lượng.

Một số chương trình tập huấn của Sở cũng như một vài tài liệu chỉ mới dừng ở mức giáo viên biết cách sử dụng tạo ra một bộ test ngẫu nhiên. Mà bộ test được tạo ngẫu nhiên thì rất yếu và không thể đáp ứng yêu cầu chấm.

2.3 Giải pháp.

Giải pháp của tôi là xây dựng file mẫu để tạo ra bộ test và đi sâu phân tích chương trình tạo ra các bộ input mạnh thông qua các ví dụ. Từ đó giúp cho người đọc định hình cách xây dựng được các nhóm test chất lượng.

Trong sáng kiến kinh nghiệm của tôi năm học 2017-2018 tôi đã trình bày cách để có một chương trình đúng, hợp lý để giải một bài tập lập trình. Dựa trên chương trình đã viết này ta có thể tiến hành tạo ra bộ test.

Trong quá trình tạo test cho các bài tập lập trình tôi thường tạo một **template** sinh test. Từ file mẫu này sẽ tiến hành tạo test cho các bài tập nhanh chóng.

**Template sinh test của tôi viết bằng Pascal:**

uses

sysutils;

const

tenbai = 'Govaoday';

var

itest : longint;

folder, filevao, filera, r : ansistring;

**Procedure taothumuc**( itest : longint);

**begin**

str(itest,r); while(length(r) < 3) do r := '0' + r;

folder := 'Test' + r;

removedir(folder); createdir(folder);

filevao := folder + '\' + tenbai + '.inp' ;

filera := folder + '\' + tenbai + '.out' ;

**end**;

**Procedure xuly;**

var

**begin**

assign(input, filevao); reset(input);

assign(output, filera); rewrite(output);

close(input); close(output);

**end**;

**Procedure taoinput;**

var

**begin**

assign(output, filevao); rewrite(output);

randomize;

close(output);

**end**;

**BEGIN**

for itest:=1 to 10 do begin

taothumuc(itest);

taoinput;

xuly;

end;

**END.**

Vòng for trong chương trình chính sẽ duyệt và tạo ra số lượng test case theo yêu cầu. Với mỗi **itest** chương trình sẽ tạo ra một thư mục có phần hậu tố có số thứ tự chính là **itest**. Ví dụ itest = 1 sẽ tạo ra một thư mục **Test001**. (Tên thư mục này bạn có thể thay đổi tùy theo nhu cầu của mình khi tạo test). Sau khi tạo thư mục này sẽ gọi hai chương trình:

* Taoinput: Tạo ra file input
* Xuly: Giải bài và tạo ra file output

Ta chỉ cần lưu lại **templates** này và có thể sử dụng cho tất cả các bài tập muốn tạo test case. Với mỗi bài tập khi muốn tạo test case ta chỉ cần:

* Tạo thư mục chứa test (Thường là tên của bài tập)
* Copy file **template** tạo test vào thư mục vừa tạo
* Sửa đổi tên bài tập, copy chương trình đã được viết chính xác vào chương trình con **Xuly**.
* Điều chỉnh vòng for trong chương trình chính và sửa đổi chương trình con **Taoinput** để tạo ra test case.

Ưu điểm của việc sử dụng **template** so với cách sử dụng phần mềm trung gian tạo test đó chính là đơn giản, mọi thao tác sửa đổi chương trình, biên dịch, chạy chương trình để tạo ra test sẽ được thực hiện hoàn toàn trên Pascal nên rất thuận tiện, dễ quản lý.

Tính chuẩn xác của chương trình con **Xuly** tôi đã nhắc đến trong SKKN năm trước, trong phần tiếp theo tôi chỉ xin nói đến kinh nghiệm tạo test của tôi bằng cách sửa đổi chương trình con **taoinput** để tạo ra một bộ test đủ tốt. Để dễ hình dung tôi sẽ minh hoạ từng bước tạo ra một bộ test chất lượng cho bài tập sau:

Bạn Mì năm nay học lớp chồi, bạn rất thích vui đùa với các con số, với các chữ cái. Bạn mì dùng máy tính của Ba để gõ các kí tự này vào máy tính. Ba bạn Mì đã giúp bạn lưu lại các kí tự đã gõ vào tệp **Xauhay.inp**

Thật tình cờ hôm nay Ba bạn Mì dạy các anh chị 11 về xâu đối xứng, nên ba bạn Mì có bài tập cho các bạn như sau: đếm xem trong tệp bạn Mì gõ có bao nhiêu số đối xứng.

**Dữ liệu**

* Gồm nhiều dòng, mỗi dòng là một đoạn các kí tự chữ cái, chữ số bạn Mì đã gõ.

**Kết quả**

* Ghi ra số lượng số đối xứng tìm được vào tệp xauhay.out

**Ví dụ**

|  |  |
| --- | --- |
| **Xauhay.inp** | **Xauhay.out** |
| abcBG343Mi121San3568vinanilk9  chaoban12 | 3 |

Giải thích: có 3 số đó là 343, 121 và 9

**Ràng buộc:**

- Số dòng tối đa là 100.

- Mỗi dòng chứa tối đa 300 kí tự.

Để tạo test cho một bài toán tôi thường chia ra các nhóm test khác nhau, mỗi nhóm test bao gồm một số test để đánh giá chương trình ở một mức độ nhất định nào đó. Sau đây là một số nhóm test tôi thường tạo ra.

2.3.1 Các test nhỏ, test đặc biệt

Các test này chúng ta sẽ dành cho các trường hợp đặc biệt của bài toán, các test nhỏ có thể sinh bằng tay. Nhóm test này rất quan trọng, cần suy nghĩ cẩn thận để xét hết các trường hợp đặc biệt của bài toán.

Trong bài này ta có thể tự suy nghĩ về các trường hợp như: không có số đối xứng, số đối xứng có 1 chữ số, xâu đó toàn là số và không đối xứng, xâu đó toàn là số và đối xứng, xâu gồm các số đối xứng cách nhau một kí tự, xâu gồm các số đối xứng cách nhau bởi các xâu kí tự đối xứng, xâu ngẫu nhiên …

**Ví dụ 1:** Ta sửa chương trình **taoinput** và chương trình chính tương ứng cho trường hợp: Xâu gồm các số đối xứng cách nhau bởi các xâu kí tự đối xứng.

* Chương trình chính:

for itest:=1 to 1 do begin

* Chương trình tạo input:

**Procedure taoinput1;**

var s : ansistring;

begin

assign(output, filevao); rewrite(output);

s := ‘121aba444ccc56765g90909hkjjkh’;

write(s);

close(output);

end;

* Chạy chương trình sẽ tạo ra test đầu tiên là **Test001**

**Ví dụ 2:** Trường hợp chỉ có một xâu, chỉ gồm chữ cái đối xứng

* Chương trình chính:

**for itest:=2 to 2 do begin**

* Chương trình tạo input:

**Procedure taoinput1;**

var s : ansistring;

begin

assign(output, filevao); rewrite(output);

**s := ‘abccbahhhakka’;**

write(s);

close(output);

end;

* Chạy chương trình sẽ tạo ra **Test002**

**Ví dụ 3:** Trường hợp có hai xâu trên hai dòng, mỗi xâu chứa toàn là số và đối xứng

* Chương trình chính:

for itest:=3 to 3 do begin

* Chương trình tạo input:

**Procedure taoinput1;**

var s, r : ansistring;

begin

assign(output, filevao); rewrite(output);

**s := ‘123321’;**

**r := ‘123456789987654321’;**

**writeln(s);**

**writeln(r);**

close(output);

end;

* Chạy chương trình sẽ tạo ra **Test003**

Qua ba ví dụ trên ta đã hình dung được cách sửa đổi chương trình chính và chương trình con Taoinput để tạo ra test case. Trong các ví dụ sau tôi chỉ xin trình bày cách sửa đổi chương trình **Taoinput**.

2.3.2 Test cho các trường hợp thuật toán còn sai

Các test này được tạo ra nhằm mục đích kiểm tra và đánh giá các sai lầm mà thuật toán của học sinh còn mắc phải. Một số sai lầm như sai kiểu dữ liệu, tham lam sai, sai giới hạn, đọc dữ liệu thiếu, vòng lặp while sai do kiểm tra điều kiện chưa hợp lý… Một số sai lầm đặc biệt trong khi học sinh cài đặt thuật toán nếu ta tạo ra test case ngẫu nhiên không đánh giá hết được. Đôi khi các sai lầm của học sinh lại rơi vào các trường hợp đặc biệt ở nhóm test 1 nên chúng ta cần xem xét cẩn thận tránh trùng lặp test.

**Ví dụ 4:** Trường hợp học sinh sai khi tách xâu, chuyển xâu về số rồi mới kiểm tra tính đối xứng. Như thế chỉ kiểm tra được các xâu số có giá trị bé hơn xấp xỉ 18\*1018 Lúc này ta có thể tạo test chứa các xâu số đối xứng từ 21 kí tự trở lên để bắt lỗi này.

**Procedure taoinput2;**

var s, r : ansistring;

begin

assign(output, filevao); rewrite(output);

**s := ‘123456789000987654321’;** //Xâu 21 kí tự DX

**writeln(s);**

close(output);

end;

**Ví dụ 5:** Trường hợp học sinh sai lầm trong vòng lặp dẫn đến tách thiếu số cuối cùng trong xâu. Lúc này test của ta sẽ có xâu số đối xứng ở cuối.

**Procedure taoinput2;**

var s, r : ansistring;

begin

assign(output, filevao); rewrite(output);

**s := ‘abc121cd1d4vinhdinh555’;**

**writeln(s);**

close(output);

end;

**Ví dụ 6:** Trường hợp học sinh sai lầm trong đọc đề dẫn đến chỉ đọc một xâu duy nhất. Khi đó test case của mình sẽ có từ 3 dòng trở lên, dòng 2 trở đi có số đối xứng thì đáp án của học sinh chắc chắn sai.

**Procedure taoinput2;**

var s, r : ansistring;

begin

assign(output, filevao); rewrite(output);

**s := ‘vinhdinh2507chaodonban444vetham123321’;**

**r := ‘truongcu212bensong1111’;**

**writeln(s);**

**writeln(r);**

close(output);

end;

2.3.3 Test ngẫu nhiên

Một số bài tập ta có thể không thể hình dung hết được các sai lầm mà học sinh có thể mắc phải. Như vậy thì các test trong hai giải pháp trên có thể bỏ sót các sai lầm của học sinh. Trường hợp này tôi thường sinh các test ngẫu nhiên. Để test ngẫu nhiên đủ độ mạnh cần thiết thì cần một số lượng nhất định chứ 1 vài test ngẫu nhiên thì chưa đủ. Ví dụ bài tập **SEQ198** trên NTUcoder, cách code bài này khá phức tạp, tôi sinh ra 260 test ngẫu nhiên. Xem trang chấm bài sẽ thấy học sinh sai rất nhiều trường hợp khác nhau với bộ test ngẫu nhiên này. Ảnh sau chỉ là một vài trường hợp như thế:



**Ví dụ 7:** Với nhóm test này ta sẽ tạo ra input không quá 50 dòng, Mỗi dòng có độ dài không quá 50 gồm các kí tự được tạo ngẫu nhiên. Với mỗi kí tự trong xâu 70% ưu tiên tạo kí tự số còn lại là kí tự chữ cái.

- Số dòng input: Ta lấy ngẫu nhiên từ 10 đến 50

- Với mỗi dòng: Độ dài lấy ngẫu nhiên là Len; ta sẽ duyệt từ 1 đến Len để lần lượt tạo ra các kí tự cho dòng này. Lệnh chr(random(10)+48) trả về một kí tự số.

Procedure taoinput3;

var i, j, len, m, x, y : longint;

begin

assign(output, filevao); rewrite(output);

randomize;

m := 10 + random(41); //So dong

for i:=1 to m do begin

len := 40 + random(11); //Do dai dong i

for j:=1 to len do begin

x := 1 + random(10);

if(x<=7) then

write(chr(random(10)+48))

else

write(chr(random(26)+97));

end;

writeln;

end;

close(output);

end;

**Ví dụ 8:** Tương tự ví dụ 7 nhưng số dòng lên đến 100 và độ dài dòng không quá 100. Ta chỉ cần sửa giới hạn hàm random của m và len.

Procedure taoinput3;

var i, j, len, m, x, y : longint;

begin

assign(output, filevao); rewrite(output);

randomize;

**m := 90 + random(11);**

for i:=1 to m do begin

**len := 90 + random(11);**

for j:=1 to len do begin

x := 1 + random(10);

if(x<=7) then write(chr(random(10)+48))

else write(chr(random(26)+97));

end;

writeln;

end;

close(output);

end;

2.3.4 Test ngẫu nhiên có tính toán

Test ngẫu nhiên vẫn có nhược điểm là không thực sự mạnh, đặc biệt các test về xâu kí tự được tạo ngẫu nhiên rất xấu và yếu. Các test ở giải pháp 4 này ta vẫn tạo ra ngẫu nhiên bằng random, tuy nhiên khi sinh ngẫu nhiên ta tính toán để sinh ra theo đúng ý đồ của mình.

Quay lại bài xauhay, với cách làm ví dụ 7 và ví dụ 8 các xâu được tạo ngẫu nhiên nên số đối xứng được sinh ra có độ dài rất bé. Ta cần ngẫu nhiên nhưng các xâu đối xứng dài hơn, hoặc các xâu đối xứng ngẫu nhiên ngăn cách nhau bởi các kí tự ngẫu nhiên … thì random thế nào? Khi đó ta cần có chương trình **taoinput** tốt hơn.

**Ví dụ 9:** Số dòng ngẫu nhiên không quá 100, mỗi dòng là một xâu số đối xứng.

- Số dòng: Ta sẽ lấy ngẫu nhiên từ 50 đến 100

- Mỗi dòng là một xâu số đối xứng nên thuật toán của ta sẽ tạo ra ngẫu nhiên một xâu số **s**. Khi đó xâu đối xứng sẽ là: **s** và đảo ngược của **s**.

**function TaoxauDX(len: longint) : ansistring;**

var s, r : ansistring; i:longint;

begin

s:=''; r:='';

for i:=1 to len div 2 do s := s + chr(random(10)+48);

for i:=len div 2 downto 1 do r:=r+s[i];

if (2\*length(r) = len) then exit(s+r);

exit(s + chr(random(10)+48) + r);

end;

**Procedure taoinput4;**

var i, j, len, m, x, y : longint;

begin

assign(output, filevao); rewrite(output);

randomize;

m := 50 + random(51);

for i:=1 to m do begin

len := 1 + random(4);

writeln(TaoxauDX(len));

end;

close(output);

end;

Chương trình **TaoxauDX**(len) sẽ tạo ra một xâu số đối xứng có độ dài len.

**Ví dụ 10:** Số dòng ngẫu nhiên không quá 100, mỗi dòng gồm các xâu đối xứng số độ dài không quá 18 cách nhau bởi các kí tự chữ cái. Mục đích kiểm tra trường hợp các em đọc đoạn kí tự số ra thành số nguyên rồi mới kiểm tra đối xứng.

Procedure taoinput4;

var i, j, len, m, n, x, y : longint;

begin

assign(output, filevao); rewrite(output);

randomize;

m := 50 + random(51);

for i:=1 to m do begin

len := 90 + random(6);

while(len > 0) do begin

x := 1 + random(4);

n := min(len-x, 1 + random(18));

len := len – n - x;

write(TaoxauDX(n));

for j:=1 to x do write(chr(random(26)+97));

end;

writeln;

end;

close(output);

end;

- Số dòng: Ngẫu nhiên từ 50 đến 100.

- Mỗi dòng: độ dài **len** ngẫu nhiên từ 90 đến 95. Lần lượt sinh ngẫu nhiên các xâu số đối xứng có độ dài không quá 18. in xâu này kèm theo x kí tự chữ cái để phân cách.

Nhóm test ngẫu nhiên 3 và 4 này nhất thiết cần phải xem lại chất lượng từng test sau khi sinh ra để đảm bảo test như mong muốn. Nếu test sinh ra không chất lượng thì cẩn cải tiến phần sinh ngẫu nhiên để tạo lại nhóm test này.

2.3.5 Test lớn để kiểm tra thuật toán tối ưu.

Một phần không thể thiếu trong bộ test hoàn chỉnh đó là các test lớn để đánh giá được các thuật toán tối ưu. Chỉ có những thuật toán tối ưu mới đạt điểm ở nhóm test này. Thường thì nhóm test này sẽ là test với bộ dữ liệu lớn, bao trùm cả 4 nhóm test đầu nhưng với dữ liệu lớn hơn.

Ví dụ: Một bài tập tham lam nhằm mục đích đánh giá thuật toán sắp xếp của học sinh với ràng buộc:

* 50% test có n <= 104.
* 50% test có n <= 105.

Vậy thì 50% test đầu n <=104 nếu học sinh sử dụng thuật toán sắp xếp có độ phức tạp O(n2) có thể ăn hết điểm nhưng 50% test cuối thì không. Do vậy 50% test cuối phải là các test có N lớn xấp xỉ 105 để thuật toán sắp xếp có độ phức tạp O(n2) không thể đáp ứng yêu cầu về thời gian chạy.

Với bài **xauhay** nhóm test tối ưu này ta cần đánh giá được một thuật toán toán tốt có thể giải quyết được các trường hợp đặc biệt, chạy được với các dữ liệu lớn, có sử dụng kiểu ansistring …

**Ví dụ 11:** Test gồm 100 dòng, mỗi dòng có 300 kí tự: Gồm các số đối xứng có độ dài không quá 30 phân cách nhau 1 kí tự chữ cái.

Procedure taoinput5;

var i, j, len, m, n, x, y : longint;

begin

assign(output, filevao); rewrite(output);

randomize;

m := 100;

for i:=1 to m do begin

len := 300;

while(len > 0) do begin

n := min(len - 1, random(31));

len := len - n - 1;

write(TaoxauDX(n));

write(chr(random(26)+97));

end;

writeln;

end;

close(output);

end;

**Ví dụ 12:** Test gồm 100 dòng, mỗi dòng có 300 kí tự: Gồm các số đối xứng có độ dài không quá 60 phân cách nhau bởi xâu kí tự. Riêng dòng cuối đánh giá thêm trường hợp đặc biệt.

Procedure taoinput5;

var i, j, len, m, n, x, y : longint;

begin

assign(output, filevao); rewrite(output);

randomize;

for i:=1 to 99 do begin

len := 300;

while(len > 0) do begin

n := min(len - 1, random(31));

len := len - n - 1;

write(TaoxauDX(n));

write(chr(random(26)+97));

end;

writeln;

end;

//Dong cuoi danh gia truong hop dac biet

Writeln(‘1a121aba2222aa12345678900987654321abc454’);

close(output);

end;

Qua 12 ví dụ được trình bày ở trên tôi đã minh hoạ một phần giải pháp của tôi trong quá trình tạo ra test case cho các bài tập lập trình. Khi tạo xong bộ test thì nhất định nên kiểm tra chất lượng test, chấm thử bài làm với bộ test này để đảm bảo tính chính xác.

Quá trình tạo ra một bộ test case chất lượng không hề dễ dàng nên tôi vẫn đang tiếp tục học tập nâng cao kiến thức chuyên môn nghiệp vụ của mình từng ngày. Mong rằng một số kinh nghiệm tạo test trên có thể giúp cho các bạn đồng nghiệp tham khảo trong quá trình tạo test case của mình.

2.4 Ưu điểm của giải pháp

- Một là việc xây dựng **template** để tạo test so với cách sử dụng phần mềm trung gian tạo test đó chính là đơn giản, mọi thao tác sửa đổi chương trình, biên dịch, chạy chương trình để tạo ra test sẽ được thực hiện hoàn toàn trên Pascal nên rất thuận tiện, dễ quản lý.

- Hai là giải pháp này tạo ra được một bộ test chất lượng giúp kiểm tra được mức độ hiệu quả của thuật toán có đáp ứng yêu cầu của bài hay không.

- Ba là giúp kiểm tra tính đúng của của các thuật toán mà học sinh đã xây dựng. Từ đó tạo hứng thú cho học sinh trong quá trình học.

- Bốn là có thể xem SKKN như là một tài liệu tham khảo cho anh chị em đồng nghiệp, giúp mọi người có thể xây dựng tốt các bài tập lập trình trong kiểm tra học sinh cũng như tạo đề cho các kì thi.

2.5 Kết quả đạt được:

Năm đầu tiên tôi được giao phụ trách đội tuyển HSG 12, các bài tập thường đánh giá theo kiểu cảm nhận là đúng khi đã test một vài trường hợp. Kết quả năm đó đội của tôi không có thành tích cấp tỉnh.

Qua các năm sau áp dụng sáng kiến bước đầu mang lại kết quả khả quan. Các đội tuyển học sinh giỏi tiếp theo do tôi tham gia giảng dạy đều học tập tốt hơn. Kết quả các đội này đều có đạt giải trong kì thi HSG cấp tỉnh.

* + Đội 11 năm 2014: 1 giải khuyến khích
  + Đội 12 năm 2016: 2 giải khuyến khích.
  + Đội 12 năm 2017: 1 giải ba, 1 giải KK.
  + Đội 12 năm 2018: 1 giải nhì, 1 giải KK

Kinh nghiệm tạo test này được tôi đưa vào triển khai cho đồng nghiệp trong tổ Tin học vào chuyên đề tháng 02 năm 2019 và được đồng nghiệp trong tổ đánh giá rất cao.

III. KẾT LUẬN

3.1 Về mặt nội dung sáng kiến:

Qua áp dụng thực tế giảng dạy từ năm 2014 đến năm 2018 nội dung của SKKN đảm bảo đánh giá được chất lượng thuật toán mà học sinh đã xây dựng.

Học sinh được đánh giá với một số lượng test đủ lớn giúp cho các em nắm được các trường hợp đặc biệt của bài toán từ đó có thể xây dựng được thuật toán tốt cho các bài toán khác.

Nội dung sáng kiến bao quát được các trường hợp đặc biệt của bài toán. Từ đó đưa vào áp dụng rất hiệu quả và tính khả thi cao.

3.2 Ý nghĩa và hiệu quả khi áp dụng sáng kiến:

Làm đề và tạo test là hai bước không thể thiếu khi dạy các bài tập lập trình, đặc biệt là công tác BD HSG. Có được một bộ test case tốt giúp đánh giá được thuật toán của học sinh từ đó giúp học sinh rút kinh nghiệm và viết chương trình tốt hơn. Vậy nên SKKN của tôi có ý nghĩa rất lớn trong giảng dạy và đánh giá chất lượng học tập thuật toán.

SKKN của tôi có thể viết lại thành một tài liệu cho các anh chị em đồng nghiệp nghiên cứu và thực hiện. Vì hiện tại chưa có tài liệu nào nói chi tiết từng bước quá trình tạo ra một bộ test chất lượng cho các bài toán.

3.3 Kiến nghị đề xuất:

Qua áp dụng và đề xuất trong chuyên đề cấp tổ của tổ Tin học các em học sinh rất hứng thú vì được đánh giá một cách chi tiết. Các đồng nghiệp trong tổ đánh giá rất cao kinh nghiệm tạo test này. Tôi xin kiến nghị được đưa SKKN của tôi lên Website của nhà trường và thư viện trường để học sinh và đồng nghiệp tham khảo.

TÀI LIỆU THAM KHẢO

1. *Sách giáo khoa Tin học 11*
2. *Sách giáo viên Tin học 11.*
3. [*https://vi.wikipedia.org*](https://vi.wikipedia.org)

*4. Các bài tập trên NTUCoder.*

*5. Các nguồn tham khảo khác trên internet.*

|  |  |
| --- | --- |
| XÁC NHẬN CỦA  THỦ TRƯỞNG ĐƠN VỊ | *Triệu Phong, Ngày 13 tháng 5 năm 2019*  Tôi xin cam đoan đây là SKKN của mình viết, không sao chép nội dung của người khác.  **LÊ THANH PHÚ** |

MỤC LỤC

[I. ĐẶT VẤN ĐỀ 1](#_Toc8653774)

[1.1 Lý do chọn đề tài: 1](#_Toc8653775)

[1.2 Mục đích nghiên cứu 1](#_Toc8653776)

[1.3 Đối tượng và khách thể nghiên cứu: 1](#_Toc8653777)

[1.4 Phương pháp nghiên cứu: 1](#_Toc8653778)

[1.5 Nhiệm vụ nghiên cứu: 1](#_Toc8653779)

[1.6 Phạm vi và kế hoạch nghiên cứu: 1](#_Toc8653780)

[II. NỘI DUNG SÁNG KIẾN 2](#_Toc8653781)

[2.1 Cơ sở lý luận 2](#_Toc8653782)

[2.1.1 Bài tập lập trình: 2](#_Toc8653783)

[2.2.2 Bộ test chấm là gì: 2](#_Toc8653784)

[2.2 Thực trạng hiện tại: 2](#_Toc8653785)

[2.2.1 Thuận lợi 2](#_Toc8653786)

[2.2.2 Khó khăn 2](#_Toc8653787)

[2.3 Giải pháp. 2](#_Toc8653788)

[2.3.1 Các test nhỏ, test đặc biệt 4](#_Toc8653789)

[2.3.2 Test cho các trường hợp thuật toán còn sai 5](#_Toc8653790)

[2.3.3 Test ngẫu nhiên 6](#_Toc8653791)

[2.3.4 Test ngẫu nhiên có tính toán 8](#_Toc8653792)

[2.3.5 Test lớn để kiểm tra thuật toán tối ưu. 9](#_Toc8653793)

[2.4 Ưu điểm của giải pháp 10](#_Toc8653794)

[2.5 Kết quả đạt được: 11](#_Toc8653795)

[III. KẾT LUẬN 12](#_Toc8653796)

[3.1 Về mặt nội dung sáng kiến: 12](#_Toc8653797)

[3.2 Ý nghĩa và hiệu quả khi áp dụng sáng kiến: 12](#_Toc8653798)

[3.3 Kiến nghị đề xuất: 12](#_Toc8653799)

[TÀI LIỆU THAM KHẢO 12](#_Toc8653800)

[MỤC LỤC 13](#_Toc8653801)