**Mục lục**

**Phần A. Đặt vấn đề…………………………………………………………………....1**

1. Cơ sở khoa học và thực tiễn trong việc chọn đề tài…………………………………..1
2. Mục đích nhiệm vụ của việc thực hiện đề tài…………………………………...2
3. Đối tượng, thời gian và phương pháp nghiên cứu……………………………… 2
4. Đối tượng nghiên cứu……………………………………………………2
5. Thời gian thực hiện……………………………………………………...2
6. Phương pháp nghiên cứu………………………………………………...2

**Phần B. Nội dung……………………………………………………………………...4**

1. Bài toán liệt kê và phương pháp sinh tin học…………………………………...4
2. Bài toán liệt kê……………………………………………………………...4
3. Phương pháp sinh tin học…………………………………………………...4
4. Ứng dụng phương pháp sinh để giải bài toán liệt kê……………………………6
5. Dạng 1: Tìm tất cả các nghiệm……………………………………………..6
6. Dạng 2: Tìm một nghiệm………………………………………………….27
7. Dạng 3: Nghiệm tối ưu thỏa mãn điều kiện……………………………….29

**Phần C. Kết luận và kiến nghị………………………………………………………32**

**Tài liệu tham khảo…………………………………………………………………...33**

***Tên đề tài:***

**ỨNG DỤNG PHƯƠNG PHÁP SINH**

**ĐỂ GIẢI BÀI TOÁN LIỆT KÊ**

**PHẦN A. ĐẶT VẤN ĐỀ:**

**I. Cơ sở khoa học và thực tiễn trong việc chọn đề tài:**

Đối với mỗi giáo viên trung học phổ thông nói chung và giáo viên bộ môn tin học ở trường trung học phổ thông nói riêng, ngoài nhiệm vụ chính là giảng dạy ở lớp thì ôn thi học sinh giỏi là nhiệm vụ rất quan trọng và cần thiết. Việc tìm tòi, sưu tầm, biên tập và tích lũy các dạng bài toán và phương pháp giải là công việc thường nhật nhằm nâng cao trình độ, chuyên môn, nghiệp vụ, tích lũy kinh nghiệm của bản thân.

Ngôn ngữ lập trình là một trong những nội dung chính được đưa vào giảng dạy ở trường trung học phổ thông, việc nắm vững tư duy thuật toán và sử dụng ngôn ngữ lập trình thành thạo thì học sinh mới có thể giải được bài toán trên máy tính.

Trong khi dạy học môn tin học ở trường trung học phổ thông, đặc biệt là khi ôn thi đội tuyển học sinh giỏi tôi nhận thấy rằng những kiến thức trong tin học phổ thông còn hạn chế, không đủ đáp ứng cho việc giải một số bài toán trong kì thi học sinh giỏi, thậm chí có nhiều bài toán không thể giải được. Mặt khác mỗi bài toán có thể có nhiều phương pháp khác nhau để giải, mỗi phương pháp lại có ưu điểm, nhược điểm riêng. Với đề thi học sinh giỏi môn tin học lớp 11 thì dạng bài toán liệt kê hay được đề cập đến. Thông thường khi gặp dạng bài toán này thì học sinh có thể sử dụng phương pháp đệ quy để giải. Tuy nhiên với dạng bài toán liệt kê tất cả các cấu hình với dữ liệu vào khá lớn nếu dùng kỹ thuật đệ quy sẽ bị hạn chế về ô nhớ và tốn rất nhiều thời gian để thực hiện. Trong khi đó yêu cầu thời gian thực hiện càng ngắn càng tốt, với trường hợp này nếu sử dụng phương pháp sinh tin học để giải quyết thì bài toán sẽ trở nên đơn giản hơn nhiều.

Nhằm cung cấp cho học sinh và giáo viên nhiều hơn các phương pháp khác nhau để có thể giải bài toán trên máy tính, từ đó học sinh có thể lựa chọn, so sánh và đưa ra cách giải quyết bài toán nhanh nhất, tối ưu nhất tôi chọn đề tài “ứng dụng phương pháp sinh để giải bài toán liệt kê”.

II. **Mục đích, nhiệm vụ của việc thực hiện đề tài:**

Khi giải các bài toán dạng liệt kê bằng phương pháp sinh tin hoc tôi nhận thấy rằng đây là một phương pháp đơn giản, dễ hiểu, dễ áp dụng để giải các bài toán dạng liệt kê.

Việc tìm hiểu và ứng dụng phương pháp sinh để giải bài toán liệt kê là cần thiết nhằm giúp học sinh có nhiều phương pháp khác nhau để giải bài toán, từ đó áp dụng vào giải các bài toán cụ thể.

Hiện nay ngôn ngữ lập trình được giảng dạy trong chương trình tin học phổ thông là ngôn ngữ lập trình Pascal nên các ví dụ đều được viết bằng ngôn ngữ Free Pascal để minh họa và do đó tôi đề ra mục đích, nhiệm vụ cụ thể của việc thực hiện đề tài:

* Giới thiệu bài toán liệt kê và phương pháp sinh tin học.
* Tìm hiểu rõ hơn, sâu hơn về phương pháp sinh tin học.
* Ứng dụng phương pháp sinh để giải các bài toán dạng liệt kê.
* Nâng cao khả năng lập trình bằng ngôn ngữ lập trình Pascal.

**Vì thế cấu trúc nội dung gồm:**

Mục I. Bài toán liệt kê và phương pháp sinh tin học

Mục II. Ứng dụng phương pháp sinh để giải bài toán liệt kê

**III. Đối tượng, thời gian và phương pháp nghiên cứu:**

1. **Đối tượng nghiên cứu:**

Bài viết sáng kiến kinh nghiệm “ỨNG DỤNG PHƯƠNG PHÁP SINH ĐỂ GIẢI BÀI TOÁN LIỆT KÊ” có đối tượng nghiên cứu là các bài toán có dạng liệt kê tổ hợp và lý thuyết về phương pháp sinh tin học.

1. **Thời gian thực hiện:**

Sáng kiến kinh nghiệm được thực hiện từ năm học 2017 – 2018 cho đến thời gian hoàn thành là đầu năm học 2018 – 2019.

1. **Phương pháp nghiên cứu:**

Để hoàn thành sáng kiến kinh nghiệm tôi đã sử dụng nhiều phương pháp khác nhau như:

* Phương pháp nghiên cứu lý thuyết về phương pháp sinh tin học
* Phương pháp thu thập thông tin và tổng hợp, phân tích, so sánh dựa trên lý thuyết về phương pháp sinh tin học.
* Phương pháp tìm tòi và nghiên cứu tài liệu

**PHẦN B. NỘI DUNG**

**I. Bài toán liệt kê và phương pháp sinh tin hoc:**

1. **Bài toán liệt kê:**

Trong thực tế, có một số bài toán yêu cầu chỉ rõ: trong một tập các đối tượng cho trước có bao nhiêu đối tượng thỏa mãn những điều kiện nhất định. Bài toán đó gọi là bài toán đếm cấu hình tổ hợp.

Trong lớp các bài toán đếm, có những bài toán yêu cầu chỉ rõ những cấu hình tìm được thỏa mãn điều kiện đã cho là những cấu hình nào. Bài toán đưa ra danh sách cấu hình có thể có gọi là bài toán liệt kê tổ hợp. Khi giải bài toán liệt kê cần đáp ứng hai yêu cầu sau:

* Không được lặp lại một cấu hình nào
* Không được bỏ sót một cấu hình nào

1. **Phương pháp sinh tin học (Generation):**

***2.1 Định nghĩa:***

* Sinh: Tạo ra dữ liệu
* Phương pháp sinh: Từ dữ liệu ban đầu, tạo ra dữ liệu kế tiếp cho đến khi kết thúc
* Dùng để giải quyết bài toán liệt kê của lý thuyết tổ hợp
* Điều kiện của thuật toán sinh:

1. Có thể xác định một thứ tự tập các cấu hình của tổ hợp (thứ tự của các phép gán trị, thường dùng thứ tự từ điển)
2. Có một cấu hình cuối (điều kiện kết thúc của giải thuật)
3. Có một cách để suy ra được cấu hình tiếp theo
   1. ***Mô tả:***

* *Thuật toán sinh:*

**Procedure** Generate;

**Begin**

<Xây dựng cấu hình đầu tiên>;

Stop:=**false**;

**While not**(stop) **do**

**Begin**

<Đưa ra cấu hình đang có>;

**If <**Cấu hình đang có chưa là cuối cùng> **then**

**<**Sinh\_kế\_tiếp>

**Else** stop:= **True**;

**End;**

**End;**

* *Có thể tóm tắt kỹ thuật sinh như sau:*

1. Xác định trạng thái ban đầu của bài toán
2. Xác định trạng thái kết thúc
3. Xác định một thứ tự cho các trạng thái
4. Tìm giải thuật đi từ trạng thái này sang trạng thái khác

* *Có thể mô tả phương pháp sinh như sau:*

<Xây dựng cấu hình đầu tiên>;

**Repeat**

<Đưa ra cấu hình đang có>;

<Từ cấu hình đang có sinh ra cấu hình kế tiếp nếu còn>;

**Until** <hết cấu hình>;

Kỹ thuật sinh cơ bản là từ cấu hình ban đầu, ta tăng lên một đơn vị theo thứ tự từ điển để có cấu hình tiếp theo, quá trình được lặp lại với cấu hình hiện tại cho đến khi gặp cấu hình cuối cùng.

Ví dụ: Liệt kê dãy nhị phân n (với n=8):

| *Cấu hình ban đầu là:* | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| *Cấu hình tiếp theo là:* | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |
| *……………………* | … | … | … | … | … | … | … | … |
| *Nếu cấu hình hiện tại là:* | **0** | **1** | **0** | **1** | **1** | **0** | **1** | **1** |
|  |  |  |  |  |  |  |  |  |
| *Cấu hình tiếp theo là:* | **0** | **1** | **0** | **1** | **1** | **1** | **0** | **0** |
| *……………………* | … | … | … | … | … | … | … | … |
| *Cấu hình cuối cùng là:* | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |

Dựa vào kỹ thuật sinh cơ bản nêu trên, ta có thể xây dựng thuật toán sinh cụ thể cho từng dạng bài toán cụ thể. Để xây dựng được thuật toán sinh vừa đơn giản vừa tối ưu, ngoài việc đòi hỏi phải hiểu sâu hơn về phương pháp sinh thì còn phụ thuộc vào kỹ năng của mỗi người. Sau đây tôi xin đề xuất một số thuật toán sinh đơn giản để giải các bài toán cơ bản có dạng liệt kê tổ hợp.

**II. Ứng dụng phương pháp sinh để giải bài toán liệt kê:**

1. ***Dạng 1: Tìm tất cả các nghiệm***

***Bài toán 1:*** Liệt kê dãy nhị phân độ dài n

**Input:** file văn bản Nhiphan.Inp chứa số nguyên dương N.

**Output:** file văn bản Nhiphan.Out ghi các dãy nhị phân, mỗi dãy ghi trên một dòng.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ý tưởng:**  -Sử dụng mảng X= (x­­­0, x1, x2,…,xn) trong đó: xi (i=1,2,..,n)dùng để biểu diễn phần tử thứ i  trong dãy nhị phân, x0 do ta thêm vào để xác định việc kết thúc vòng lặp.  - Xác định dãy nhị phân đầu tiên là:  x1=x2=…=xn=0 (và x0=0);  - Dãy nhị phân tiếp theo được sinh ra bằng  cách cộng thêm 1 vào dãy trước đó.  - Dãy nhị phân cuối cùng là:  x1=x2=…=xn=1. | |  |  | | --- | --- | | Nhiphan.Inp | Nhiphan.Out | | 4 | 0000  0001  0010  0011  0100  0101  0110  0111  1000  1001  1010  1011  1100  1101  1110  1111 | |

Nếu tiếp tục cộng thêm 1 vào dãy cuối cùng ta sẽ được dãy x1=x2=…=xn=0 (và x0=1); khi này vòng lặp kết thúc.

***Cách 1:*** Ta xây dựng thuật toán sinh xi như sau:

Ban đầu i:=n; them:=1; {*cộng thêm 1 vào dãy hiện tại*}

**While** them<>0 **do** {*lặp cho đến khi them=0*}

**Begin**

tam:=x[i]+them;

x[i]:=tam **mod** 2; {*tính giá trị mới cho xi*}

them:=tam **div** 2; {*tính giá trị mới cho them*}

i:=i-1; {*them trong phép tính xi được cộng tiếp vào xi-1 cho đến khi them=0*}

**End;**

***Chương trình:***

**Program** ppsinh\_nhiphan1;

**Const** nmax=50;

**Var** x:array[0..nmax] of integer;

n,i,tam,them: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,n); close(f1);

**end;**

**{========================}**

**Procedure**  Ghicauhinh;

**Var** j : integer;

**Begin**

for j :=1 to n do write(f2,x[j], ’ ’); writeln(f2);

**end;**

{======================================}

**BEGIN**

fi:=’Nhiphan.Inp’;

fo:=’Nhiphan.Out’;

Doctep;

Assign(f2,fo); rewrite(f2);

**For** i:=0 **to** n **do** x[i]:=0; {*xây dựng cấu hình đầu tiên*}

**Repeat**

Ghicauhinh;

i:=n; them:=1;

**while** them<>0 **do**

**begin**

tam:=x[i]+them;

x[i]:=tam **mod** 2; them:=tam **div** 2;

i:=i-1;

**end;**

**Until** x[0]<>0;

Close(f2);

**END.**

***Cách 2:*** Bắt đầu từ phần tử cuối cùng (i:=n), nếu gặp xi=1 thì thì ta gán xi:=0; và xét phần tử thứ i-1 cho đến khi gặp phần tử xi=0 thì gán xi:=1;

Ta xây dựng thuật toán sinh như sau:

Ban đầu i :=n;

**While** x[i]=1 **do**

**Begin**

x[i]:=0;

i :=i-1;

**end;**

x[i]:=1;

***Chương trình:***

**Program** ppsinh\_nhiphan2;

**Const** nmax=50;

**Var** x:array[0..nmax] of integer;

n,i: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,n); close(f1);

**end;**

{==========================}

**Procedure**  Ghicauhinh;

**Var** j: integer;

**Begin**

**for** j:=1 **to** n **do** write(f2,x[j], ’ ’); writeln(f2);

**end;**

{======================================}

**BEGIN**

fi:=’Nhiphan.Inp’;

fo:=’Nhiphan.Out’;

Doctep;

Assign(f2,fo); rewrite(f2);

**For** i:=0 **to**  n **do** x[i]:=0; {*xây dựng cấu hình đầu tiên*}

**Repeat**

Ghicauhinh;

i:=n;

**while** x[i]=1 **do**

**begin**

x[i]:=0;

i:=i-1;

**end;**

x[i]:=1;

**Until** x[0]<>0;

Close(f2);

**END.**

***Cách 3:*** Có thể kết hợp phương pháp sinh và kỹ thuật đệ quy để giải bài toán 1 bằng cách xây dựng kỹ thuật sinh sinh như sau:

**Procedure** Sinh(h:integer);

**Begin**

**If** x[h]=0  **then** x[h]:=1 {*gặp xh=0 thì gán xh bằng 1*}

**Else**

**Begin** x[h]:=0;

Sinh(h-1); {*gặp xh=1 thì gán xh=0 và làm việc với h-1*}

**End;**

**End;**

***Chương trình:***

**Program** ppsinh\_nhiphan3;

**Const** nmax=50;

**Var** x:array[0..nmax] of integer;

n,i: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,n); close(f1);

**end;**

**{============================}**

**Procedure**  Ghicauhinh;

**Var** j : integer;

**Begin**

**for** j :=1 **to** n **do**  write(f2,x[j], ’ ’); writeln(f2);

**end;**

{======================================}

**Procedure** Sinh(h:integer);

**Begin**

**If** x[h]=0 **then** x[h]:=1

**else**

**begin** x[h]:=0;

Sinh(h-1);

**end;**

**End;**

{==================================}

**BEGIN**

fi:=’Nhiphan.Inp’;

fo:=’Nhiphan.Out’;

Doctep;

Assign(f2,fo); rewrite(f2);

**For** i:=0 **to** n **do** x[i]:=0; {*xây dựng cấu hình đầu tiên*}

**Repeat**

Ghicauhinh;

Sinh(n);

**Until** x[0]<>0;

Close(f2);

**END.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Bài toán 2:*** Liệt kê dãy k phân độ dài n.  *Input:* file văn bản Kphan.Inp chứa hai số nguyên dương k và n cách nhau ít nhất một dấu cách.  *Output:* file văn bản Kphan.Out ghi các dãy k phân độ dài n, mỗi dãy ghi trên một dòng.  ***Ý tưởng:***Sử dụng mảng X= (x­­­0, x1, x2,…,xn) trong đó: xi (i=1,2,..,n)dùng để biểu diễn phần tử thứ i trong dãy k phân, x0 do ta thêm vào để xác  định việc kết thúc vòng lặp.  - Xác định dãy k phân đầu tiên là:  x1=x2=…=xn=0 (và x0=0);  - Dãy k phân tiếp theo được sinh ra bằng cách cộng thêm 1 vào dãy trước đó.  - Dãy k phân cuối là:x1=x2=…=xn= k-1.  Nếu tiếp tục cộng thêm 1 vào dãy cuối cùng ta sẽ | |  |  | | --- | --- | | KPHAN.INP | KPHAN.OUT | | 3 3 | 000  001  002  010  011  012  020  021  022  100  101  102  110  111  112  120  121  122  200  201  202  210  211  212  220  221  222 | |

được dãy x1=x2=…=xn=0 (và x0=1), khi này vòng lặp kết thúc.

***Cách 1:***  Ta xây dựng thuật toán sinh như sau:

Ban đầu i:=n; them:=1;

**While** them<>0 **do** {*lặp cho đến khi them=0*}

**Begin** tam:=x[i]+them;

x[i]:= tam **mod** k; {*tính giá trị mới cho xi*}

them:= tam **div** k; {*tính giá trị mới cho them*}

i:=i-1; {*them khi tính xi được cộng vào cho xi-1 cho đến khi them=0*}

**end;**

***Chương trình:***

**Program** ppsinh\_kphan1;

**Const** nmax=50;

**Var** x:array[0..nmax] of integer;

n,i,tam,them: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,k,n); close(f1);

**End;**

{==========================}

**Procedure**  Ghicauhinh;

**Var** j : integer;

**Begin**

**for** j :=1 **to** n **do** write(f2,x[j], ’ ’); writeln(f2);

**end;**

{======================================}

**BEGIN**

fi:=’Kphan.Inp’;

fo:=’Kphan.Out’;

Doctep;

Assign(f2,fo); rewrite(f2);

**For** i:=0 **to** n **do** x[i]:=0; {*xây dựng cấu hình đầu tiên*}

**Repeat**

Ghicauhinh;

i:=n; them:=1;

**while** them<>0 **do**

**begin**

tam:=x[i]+them;

x[i]:=tam **mod** k;

them:=tam **div** k;

i:=i-1;

**end;**

**Until** x[0]<>0;

Close(f2);

**END.**

***Cách 2:*** Bắt đầu từ phần tư cuối cùng (i:=n), nếu gặp xi=k-1 thì thì ta gán xi:=0; và xét phần tử thứ i-1 cho đến khi gặp phần tử xi<k-1 thì gán xi:=xi+1;

Ta xây dựng thuật toán sinh như sau:

Ban đầu i :=n;

**While** x[i]=k-1 **do** {*lặp cho đến khi gặp xi<k-1*}

**Begin**

x[i]:=0;

i :=i-1;

**end;**

x[i]:=x[i]+1; {*khi gặp xi<k-1 thì tăng xi lên 1 đơn vị*}

***Chương trình:***

**Program** ppsinh\_kphan2;

**Const** nmax=50;

**Var** x:array[0..nmax] of integer;

n,i: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,n); close(f1);

**end;**

{==========================}

**Procedure**  Ghicauhinh;

**Var** j: integer;

**Begin**

**for** j:=1 **to** n **do** write(f2,x[j], ’ ’); writeln(f2);

**end;**

{======================================}

**BEGIN**

fi:=’Kphan.Inp’;

fo:=’Kphan.Out’;

Doctep;

Assign(f2,fo); rewrite(f2);

**For** i:=0 **to** n **do** x[i]:=0; {*xây dựng cấu hình đầu tiên*}

**Repeat**

Ghicauhinh;

i:=n;

**while** x[i]=k-1 **do**

**begin**

x[i]:=0;

i:=i-1;

**end;**

x[i]:=x[i]+1;

**Until** x[0]<>0;

Close(f2);

**END.**

***Cách 3:*** Có thể kết hợp phương pháp sinh và kỹ thuật đệ quy để giải bài toán bằng cách xây dựng kỹ thuật sinh như sau:

**Procedure** Sinh(h:integer);

**Begin**

**if** x[h] < k-1 **then** x[h]:=x[h]+1

**else**

**begin** x[h]:=0;

Sinh(h-1);

**end;**

**End;**

***Chương trình:***

**Program** ppsinh\_kphan3;

**Const** nmax=50;

**Var** x:array[0..nmax] of integer;

n,i: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,n); close(f1);

**end;**

{==========================}

**Procedure**  Ghicauhinh;

**Var** j: integer;

**Begin**

**for** j:=1 **to** n **do** write(f2,x[j], ’ ’); writeln(f2);

**end;**

{======================================}

**Procedure** Sinh(h:integer);

**Begin**

**If** x[h]< k-1 **then** x[h]:=x[h]+1

**else**

**begin** x[h]:=0;

Sinh(h-1);

**end;**

**End;**

{=====================================}

**BEGIN**

fi:=’Kphan.Inp’;

fo:=’Kphan.Out’;

Doctep;

Assign(f2,fo); rewrite(f2);

**For** i:=0 **to** n **do** x[i]:=0; {*xây dựng cấu hình đầu tiên*}

**Repeat**

Ghicauhinh;

Sinh(n);

**Until** x[0]<>0;

Close(f2);

**END.**

***Bài toán 3:*** TỔ HỢP (Xác định tổ hợp chập k của n phần tử).

*Input:* file văn bản Tohop.inp chứa hai số nguyên dương n, k cách nhau ít nhất một dấu cách.

*Output:* file văn bản Tohop.Out ghi các tổ hợp chập k của tập {1,2,..,n).

***Ý tưởng:***  Một tổ hợp chập k của n là một tập con k phần tử của tập n phần tử. Chẳng hạn tập {1,2,3,4,5} có các tổ hợp chập 2 là: {1,2}, {1,3}, {1,4}, {1,5}, {2,3}, {2,4}, {2,5}, {3,4}, {3,5}, {4,5}. Vì trong tổ hợp không phân biệt thứ tự nên tập {1,2} cũng là tập {2,1} do đó ta coi chúng chỉ là một tổ hợp.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Để đơn giản ta chỉ xét bài toán tìm các tổ hợp của tập các số nguyên từ 1 đến n. Đối với một tập hữu hạn bất kỳ, bằng cách đánh số thứ tự của các phần tử, ta cũng đưa được về bài toán đối với tập số nguyên từ 1 đến n.  Nghiệm của bài toán tìm các tổ hợp chập k của n phải thỏa mãn các điều kiện sau:  Là một tâp X={x1, x2, …,xk} | |  |  | | --- | --- | | TOHOP.INP | TOHOP.OUT | | 3 5 | {1,2,3}  {1,2,4}  {1,2,5}  {1,3,4}  {1,3,5}  {1,4,5}  {2,3,4}  {2,3,5}  {2,4,5}  {3,4,5} | |

Ràng buộc: xi<xi+1 với mọi giá trị i từ 1 đến k-1 (vì tập không phân biệt thứ tự phần tử nên ta sắp xếp các phần tử theo thứ tự tăng dần)

Và ta củng có ràng buộc: 1≤ x1≤ x2 ≤…≤ xk ≤ n.

***Cách 1:*** Ta có thể áp dụng bài toán 2: liệt kê dãy k phân độ dài n, ở đây ta thay bằng dãy n phân độ dài k (với k ≤ n), tuy nhiên ta chỉ lọc lấy những cấu hình thỏa mãn điều kiện: 1 ≤ x1 < x2 < …< xk ≤ n.

***Chương trình:***

**Program** ppsinh\_tohop1;

**Const** nmax=50;

**Var** x:array[0..nmax] of integer;

kt: boolean;

N,k,i,j: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,k,n); close(f1);

**end;**

{==========================}

**Procedure**  Ghicauhinh;

**Var** l: integer;

**Begin**

write(f2,’{’);

**for** l:=1 **to** k-1 **do** write(f2,x[l], ’, ’); writeln(f2,x[k],’}’);

**end;**

{======================================}

**BEGIN**

fi:=’Tohop.Inp’;

fo:=’Tohop.Out’;

Doctep;

Assign(f2,fo); rewrite(f2);

**For** i:=0 **to** k **do** x[i]:=i; {*xây dựng cấu hình đầu tiên*}

kt:=true;

**Repeat**

**If** kt **then** Ghicauhinh; {*nếu thỏa mãn* *x1<x2<…<xk thì ghi vào tệp*}

i:=k; kt:=true;

**while** x[i]=k **do** {*lặp cho đến khi gặp x[i]< k*}

**begin**

x[i]:=1;

i:=i-1;

**end;**

x[i]:=x[i]+1;

**for** j:=1 **to** k-1 **do**

**if** x[j]>=x[j+1] **then** kt:=false; {*kiểm tra điều kiện x1<x2<…<xk*}

**Until** x[0]<>0;

Close(f2);

**END.**

***Cách 2:*** Ta luôn có ràng buộc xi < xi+1 và xi ≤ n-k+i;

* Xét tổ hợp chập k của n với k=4, n=6 ta thấy:

Cấu hình đầu tiên là: {1, 2, 3, **4**}

Cấu hình tiếp theo là: {1, 2, 3, **5**}…

Nếu cấu hình hiên tại là: {**1**, 4, 5, 6}

Thì cấu hình tiếp theo là: {**2, 3, 4, 5**}…

Nếu cấu hình hiện tại là: {2, **3**, 5, 6}

Thì cấu hình tiếp theo là: {2, **4, 5, 6**}

Xét các phần tử từ cuối dãy đến đầu dãy ta thấy: khi gặp một phần tử thứ i có khả năng tăng lên (xi ≤ n-k+i) thì tăng xi lên 1 đơn vị, đồng thời các phần tử từ xi đến xk là một dãy tăng theo cấp số cộng có công sai bằng 1.

Vậy ta có thể xây dựng thuật toán sinh như sau:

i:=k;

**While** (x[i]=n-k+i) **do** i:=i-1; {*lặp cho đến khi gặp xi còn có thể tăng lên*}

x[i]:=x[i]+1; {*tăng xi lên một đơn vị*}

**for** j:=i+1 to k **do** x[j]:=x[j-1]+1; {*từ xi đến xk là một dãy tăng (xj =xj-1 +1)*}

***Chương trình:***

**Program** ppsinh\_tohop2;

**Const** nmax=50;

**Var** x:array[0..nmax] of integer;

N,k,i,j: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,k,n); close(f1);

**end;**

{==========================}

**Procedure**  Ghicauhinh;

**Var** l: integer;

**Begin**

**for** l:=1 **to** k **do** write(f2,x[l], ’ ’); writeln(f2);

**end;**

{======================================}

**BEGIN**

fi:=’Tohop.Inp’;

fo:=’Tohop.Out’;

Doctep;

Assign(f2,fo); rewrite(f2);

**For** i:=0 **to** k **do** x[i]:=i; {*xây dựng cấu hình đầu tiên*}

**Repeat**

Ghicauhinh; {*ghi cấu hình hiện tại vào tệp*}

i:=k; kt:=true;

**while (**x[i]=n-k+i) **do** i:=i-1;

x[i]:=x[i]+1;

**for** j:=i+1 **to**  k **do** x[j]:=x[j-1]+1;

**Until** x[0]<>0;

Close(f2);

**END.**

Nhận xét với ***cách 2*** thì thuật toán ít phức tạp hơn, tuy nhiên việc tìm ra quy luật để sinh ra các giá trị cho xi (i=1,2,..,k) là khó khăn hơn.

***Bài toán 4***: Liệt kê các chỉnh hợp không lặp chập k.

*Input:* file văn bản chinhhop.Inp chứa hai số nguyên dương k, n (1 ≤ k ≤ n) cách nhau ít nhất một dấu cách.

*Output:* file văn bản chinhhop.Out ghi các chỉnh hợp không lặp chập k của tâp {1, 2,…,n).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Ý tưởng:*** Để liệt kê các chỉnh hợp không lặp chập k của tập S={1, 2, …, n} ta có thể đưa về liệt kê các cấu hình (x1, x2, …, xk), ở đây các xi thuộc tập S và khác nhau từng đôi một.  Nghiệm của bài toán là tập X={x1, x2, …, | |  |  | | --- | --- | | CHINHHOP.INP | CHINHHOP.OUT | | 2 3 | 1 2  1 3  2 1  2 3  3 1  3 2 | |

xk) thỏa mán các điều kiện sau: xi ≠ xj với mọi i ≠ j.

Ta tạo một mảng c1, c2, …, cn có kiểu logic. Ở đây ci cho biết giá trị i có còn tự do hay đã bị chọn.

***Cách 1:*** Ta có thể áp dụng bài toán 2: Liệt kê dãy k phân độ dài n, ở đây ta thay bằng dãy n phân độ dài k (k≤ n). Và ta chỉ lọc lấy các cấu hình (x1, x2, …, xk) khác nhau từng đôi một.

***Chương trình:***

**Program** ppsinh\_chinhhop1;

**Const** nmax=50;

**Var** x:array[0..nmax] of integer;

c: array[1..nmax] of boolean;

kt: boolean;

N,k,i,j: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,k,n); close(f1);

**end;**

{==========================}

**Procedure**  Ghicauhinh;

**Var** l: integer;

**Begin**

**for** l:=1 **to** k **do** write(f2,x[l], ’ ’); writeln(f2);

**end;**

{======================================}

**BEGIN**

fi:=’chinhhop.Inp’;

fo:=’chinhhop.Out’;

Doctep;

Assign(f2,fo); rewrite(f2);

**For** i:=0 **to** k **do** x[i]:=i; {*xây dựng cấu hình đầu tiên*}

kt:=true;

**Repeat**

**if** kt **then** Ghicauhinh; {*nếu khác nhau đôi một thì ghi vào tệp*}

**for** j:=1 **to** n **do** c[j]:=false; {*tạo giá trị cho mang c1=c2=…=cn=flase*}

i:=k; kt:=true;

**while** x[i]=k **do** {*lặp cho đến khi gặp x[i]< k*}

**begin**

x[i]:=1;

i:=i-1;

**end;**

x[i]:=x[i]+1;

**for** j:=1 **to** k **do**

**if** c[x[j]]= false **then** c[x[j]]:=true {*nếu chưa gặp xj*}

**else** kt:=false; {*điều kiện xi ≠ xj (mọi i ≠ j)bị sai*}

**Until** x[0]<>0;

Close(f2);

**END.**

***Cách 2:*** Ta thấy với k=4, n=5 thì:

Nếu cấu hình hiệ tại là: 1 **2** 5 4 thì:

Cấu hình tiếp theo là: 1 **3** 2 4 ….

Nếu cấu hình hiện tại là: **1** 5 4 3 thì:

Cấu hình tiếp theo là: **2** 3 4 5 ……

Do đó ta có kỹ thuật sinh như sau:

* Khi xi nhận giá trị h nào đó (1 ≤ h ≤ n) thì c[h]:=true;
* Ban đầu i:=k;

1. Xét tại vị trí i nếu xh có khả năng tăng lên, tức là tồn tại h (xi+1 ≤ h ≤ n) mà c[h]=false thì ta tăng xi. Ngược lại nếu xi không có khả năng tăng lên, tức là không tồn tại h (xi+1 ≤ h ≤ n) mà c[h]=false thì ta quay lại (1) và xét vị trí thứ i-1 (i:=i-1). Cho đến khi gặp vị trí thứ i mà tại đó xi có khả năng tăng lên.
2. Khi xi được tăng lên thì các giá trị của xj (i < j ≤ k) là một dãy tăng có tập giá trị h {1≤ h ≤ n với c[h]=false}, theo thứ tự tương ứng từ nhỏ đến lớn.

Ta có thuật toán sinh như sau:

Stop:=false; i:=k;

**While** (stop=false) **and** (i>0) **do**

**Begin**

c[x[i]]:=false; h:=x[i]+1;

**while** c[h]= true **do** h:=h+1; {*tìm giá trị h mà c[h]=false*}

**if** h<= n **then** {*xi có thể tăng lên*}

**begin** x[i]:=h; {*tăng xi lên bằng h*}

c[h]:=true; stop:= true;

h:=1;

**for** j:= i+1 **to** k **do** {*gán giá trị cho xj với j=(i+1,…, k*)}

**begin**

**while** c[h]=true **do** h:=h+1; {*tìm giá trị h để gán cho xj*}

x[j]:=h; c[h]:=true;

**end**;

**end**

**else** i:=i-1;

**end;**

***Chương trình:***

**Program** ppsinh\_chinhhop2;

**Const** nmax=50;

**Var** x:array[0..nmax] of integer;

c: array[1..nmax] of boolean;

stop: boolean;

N,k,i,j,h: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,k,n); close(f1);

**end;**

{==========================}

**Procedure**  Ghicauhinh;

**Var** l: integer;

**Begin**

**for** l:=1 **to** k **do** write(f2,x[l], ’ ’); writeln(f2);

**end;**

{======================================}

**BEGIN**

fi:=’chinhhop.Inp’;

fo:=’chinhhop.Out’;

Doctep;

Assign(f2,fo); rewrite(f2);

**For** i:=0 **to**  k **do** x[i]:=i; {*xây dựng cấu hình đầu tiên*}

**Repeat**

Ghicauhinh; {*ghi cấu hình hiện vào tệp*}

i:=k; stop:=false;

**while** (stop=false) **and** (i>0) **do**

**begin**

c[x[i]]:=false; h:=x[i]+1;

**while** c[h]=true **do** h:=h+1;

**if** h <= n **then**

**begin** stop:=true;

x[i]:=h; c[h]:=true;

h:=1;

**for** j:=i+1 **to** k **do**

**begin**

**while** c[h]=true **do** h:=h+1;

x[j]:=h; c[h]:=true;

**end;**

**end**

**else** i:=i-1;

**end**;

**Until** x[0]<>0;

Close(f2);

**END.**

***Bài toán 5:*** Bài toán phân tích số.

Hãy tìm tất cả các cách phân tích số n thành tổng các số nguyên dương, các cách phân

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tích là hoán vị của nhau chỉ tính là một cách.  Input: file văn bản Phantich.Inp chứa số nguyên dương n.  Output: file văn bản Phantich.Out ghi các cách phân tích, mỗi cách ghi trên một dòng. | |  |  | | --- | --- | | PHANTICH.INP | PHANTICH.OUT | | 5 | 5 = 1 + 1 + 1 + 1 + 1  5 = 1 + 1 + 1 + 2  5 = 1 + 1 + 3  5 = 1+ 2 + 2  5 = 1 + 4  5 = 5 | |

***Ý tưởng:*** Ta sẽ lưu nghiệm trong mảng x, ngoài ra có một mảng t. Mảng t được xây dựng như sau: ti là tổng các phần tử từ x1 đến xi (ti=x1+x2+…+xi).

Khi liệt kê các dãy x có tổng các phần tử đúng bằng n, để tránh sự trùng lặp ta đưa thêm ràng buộc xi-1 ≤ xi , khi ti=n thì ghi cấu hình (x1, x2, …, xi) vào tệp.

***Chương trình:***

**Program** ppsinh\_phantichso;

**Const** nmax=50;

**Var** t, x: array[-1..nmax] of integer;

i,j,n: integer;

f1,f2: text;

fi,fo: string;

kt: boolean;

{===============================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,n); close(f1);

**End;**

{================================}

**Procedure** Ghicauhinh(k:integer);

**Var** l: integer;

**Begin**

Write(f2,n,’=’,x[1]);

For l:=2 to k-1 do write(f2,’+’,x[l]);

Writeln(f2,’+’,x[k]);

**End;**

{================================}

**BEGIN**

fi:=’phantich.inp’;

fo:=’phantich.out’;

assign(f2,fo); rewrite(f2);

Doctep;

**For** i:=1 **to** n **do**

**Begin**

x[i]:=1;

t[i]:=t[i-1]+x[i];

**end;**

i:=n;

**repeat**

**if** t[i]=n **then** ghicauhinh(i);

**if** t[i] >= n **then**

**begin**

i:=i-1;

x[i]:=x[i]+1;

t[i]:=t[i-1]+x[i];

**end;**

**while** (t[i] < n) **and** (x[0]=0)  **do**

**begin**

i:=i+1;

x[i]:=x[i-1];

t[i]:=t[i-1]+x[i];

**end;**

**Until** x[0]<>0;

Close(f2);

**END.**

1. ***Dạng 2: Tìm một nghiệm.***

***Bài toán 6:*** Bài toán máy rút tiền tự động ATM 1.

Một máy ATM hiện có n tờ tiền, có mệnh giá t1, t2, …, tn. Hãy đưa ra cách trả số tiền bằng S.

*Input:* file văn bản Atm1.Inp có dạng:

* Dòng đầu là hai số n, s.
* Dòng thứ hai gồm n số t1, t2, …, tn.

*Output:* file văn bản Atm1.Out có dạng: nếu có thể trả đúng S thì đưa ra cách trả, nếu không ghi -1.

|  |  |
| --- | --- |
| Atm1.Inp | Atm1.Out |
| 11 490  20 200 10 20 20 50 50 50 50 100 100 | 200 20 20 50 100 100 |

***Ý tưởng:*** Nghiệm của bài toán là một dãy nhị phân độ dài n, trong đó thành phần thứ i bằng 1 nếu tờ tiền thứ i được sử dụng để trả, bằng 0 trong trường hợp ngược lại.

X=(x1, x2, …, xn) là nghiệm nếu: x1×t1 + x2×t2 + … + xn×tn = S. Chương trình cần biến kt để kiểm tra xem tìm được hay chưa, ban đầu kt:=false, nếu tìm được thì kt:=true và kết thúc việc tìm kiếm.

***Chương trình:***

**Program** ppsinh\_atm1;

**Const** nmax=50;

**Var** x,t: array[0..nmax] of integer;

i,j,n,s,tong: integer;

kt: boolean;

fi,fo: string;

f1,f2: text;

{=========================}

**Procedure** Doctep;

**Var** l: integer;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,n,s);

For l:=1 to n do read(f1,t[i]);

Close(f1);

**End;**

{=============================}

**Procedure** ghicauhinh;

**Var** h: integer;

**Begin**

**If** kt=false **then** write(f2,-1)

**Else**

**For** h:=1 **to** n **do**

**If** x[i] <> 0 **then** write(f2,t[i],’ ’);

**End;**

{=======================================}

**BEGIN**

fi:=’Atm1.Inp’;

fo:=’Atm1.Out’;

assign(f2,fo); rewrite(f2);

Doctep;

tong:=0; kt:=false;

**for** i:= 1 **to** n **do** x[i]:=0;

**Repeat**

**while** x[i]=1 **do**

**begin**

x[i]:=0;

tong:=tong - t[i]; {*khi này xi=0 nên loại ti ra khỏi tong*}

i:=i-1;

**end**;

x[i]:=1; tong:=tong+t[i]; {*khi này xi=1 nên ti được cộng vào tổng*}

**if** tong= s **then** kt:=true;

**Until** (x[0] <> 0) **or** (kt=true);

Ghicauhinh;

Close(f2);

**END.**

1. ***Dạng 3: Nghiệm tối ưu thỏa mãn điều kiện.***

***Bài toán 7:*** Bài toán máy rút tiền tự động Atm2.

Một máy ATM hiện có n tờ tiền, có mệnh giá t1, t2, …, tn. Hãy tìm ra cách trả ít tờ nhất với số tiền đúng bằng S.

*Input:* file văn bản Atm2.Inp có dạng:

* Dòng đầu là hai số n, s.
* Dòng thứ hai gồm n số: t1, t2, …, tn.

*Output:* file văn bản Atm2.Out có dạng: nếu có thể trả đúng số tiền S thì đưa ra số tờ ít nhất cần trả và đưa ra cách trả, nếu không ghi -1.

|  |  |
| --- | --- |
| Atm2.Inp | Atm2.Out |
| 11 390  20 200 10 20 20 50 50 50 50 100 100 | 5  200 20 20 50 100 |

***Ý tưởng:*** Như ta đã biết nghiệm của bài toán là một dãy nhị phân độ dài n, trong đó thành phần thứ i bằng 1 nếu tờ tiền thứ i được sử dụng để trả, bằng 0 trong trường hợp ngược lại. Nếu bài toán có nhiều nghiệm thì ta chọn nghiệm có số phần tử có giá trị bằng 1 ít nhất.

***Chương trình:***

**Program** ppsinh\_atm2;

**Const** nmax=50;

**Var** x,t,c: array[0..50] of integer;

tong,dem,i,j,k,n,s: integer;

fi,fo: string;

f1,f2: text;

{==========================}

**Procedure** Doctep;

**Begin**

Assign(f1,fi); reset(f1);

Readln(f1,n,s);

**For** i:=1 **to** n **do** read(f1,t[i]);

Close(f1);

**End;**

{===============================}

**Procedure** luucauhinh;

**Var** l: integer;

**Begin**

**If** (tong=s)**and(**dem<k) **then**

**Begin**

k:=dem;

**for** l:=1 **to** n **do** c[l]:=x[l];

**end;**

**end;**

{===================================}

**BEGIN**

fi:=’Atm2.Inp’;

fo:=’Atm2.Out’;

assign(f2,fo); rewrite(f2);

Doctep;

dem:=0; tong:=0;

k:=high(integer);

**Repeat**

i:=n;

**while** x[i]=1  **do**

**begin**

x[i]:=0; tong:=tong – t[i];

dem:=dem-1; i:=i-1;

**end;**

x[i]:=1; dem:=dem+1;

tong:=tong+t[i];

luucauhinh;

**Until** x[0] <> 0;

**If** k > n **then** write(f2,-1)

**Else**

**begin** writeln(f2,k);

**For** j:=1  **to** n **do**

**If** c[j]=1 **then** write(f2,t[j],’ ’);

**End;**

Close(f2);

**END.**

**PHẦN C. KẾT LUẬN VÀ KIẾN NGHỊ**

**KẾT LUẬN**

Có thể nói rằng phương pháp sinh tin học là phương pháp đơn giản nhất, dể hiểu nhất và dể áp dụng nhất khi giải các bài toán dạng liệt kê tổ hợp. Với mục đích ban đầu đặt ra đề tài đã làm được:

* Giới thiệu cho giáo viên và học sinh về bài toán liệt kê và phương pháp sinh tin học.
* Học sinh (chủ yếu là học sinh giỏi tin và những học sinh có đam mê học môn lập trình) hiểu rõ hơn, sâu hơn về phương pháp sinh tin học. Ứng dụng phương pháp sinh để giải các bài toán dạng liệt kê tổ hợp, đặc biệt là giải các bài toán trong kỳ thi học sinh giỏi.
* Việc biết được nhiều phương pháp khác nhau để giải bài toán trên máy tính giúp học sinh thích thú hơn, đam mê học môn tin học lập trình hơn.

Qua các nội dung được trình bày ở trên, đặc biệt là một số kỹ thuật sinh tôi đã đề xuất, hy vọng các giáo viên và học sinh sẽ lựa chọn và áp dụng vào ôn tập và bồi dưỡng học sinh giỏi đạt kết quả cao.

Mặc dù bản thân đã cố gắng, nỗ lực rất nhiều, tuy nhiên có thể có một số thuật toán còn có hạn chế và chưa thật sự tối ưu, rất mong nhận được nhiều ý kiến đóng góp từ quý thầy cô và đồng nghiệp.

**KIẾN NGHỊ**

Bài viết sáng kiến kinh nghiệm “*ứng dụng phương pháp sinh để giải bài toán liệt kê*” đã giới thiệu tương đối đầy đủ và chi tiết về phương pháp sinh tin học và đã áp dụng để giải tất cả các dạng bài toán liệt kê tổ hợp cơ bản nhất, hay gặp nhất.

Đây là tài liệu tốt có thể sử dụng cho các giáo viên và học sinh ở trường trung học phổ trong việc ôn tập và bồi dưỡng học sinh giỏi bộ môn tin học.

Trong quá trình sử dụng tài liệu để bồi dưỡng học sinh giỏi, giáo viên có thể mở rộng bằng cách bổ sung thêm các dạng bài toán mới để tạo thành một bộ tài liệu tốt hơn nữa phục vụ cho quá trình ôn thi học sinh giỏi của mình.

**TÀI LIỆU THAM KHẢO**

1. Sách giáo khoa Tin học 11. NXB Giáo dục – Năm 2004
2. Lê Minh Hoàng, bài giảng chuyênđề: giải thuật và lập trình – Đại học Sư phạm Hà Nội, 1999 – 2002
3. Trang xemtailieu.com