THUẬT TOÁN NÂNG CAO

THUẬT TOÁN THAM LAM

Tháng 9 năm 2020

I. TỔNG QUAN VỀ THUẬT TOÁN THAM LAM

Phương pháp tham lam thường dùng để giải quyết các bài toán tối ưu.

Quan niệm của thuật toán này là: tối ưu từng bước ➔ tối ưu toàn cục.

Ý tưởng: Giải thuật tham lam giải bài toán mà lời giải phụ thuộc vào việc xác định một trật tự xử lý. Trật tự có thể là:

+ Từ lớn đến nhỏ

+ Từ nhỏ đến lớn

+ Hoặc thỏa điều kiện p nào đó.

Nội dung kỹ thuật tham tham (Greedy Method)

Kỹ thuật tham lam thường được vận dụng để giải bài toán tối ưu tổ hợp bằng cách xây dựng một phương án tối ưu X. Phương án X được xây dựng bằng cách lựa chọn từng thành phần xi của X cho đến khi hoàn chỉnh (đủ n thành phần). Với mỗi xi , ta sẽ chọn xi tối ưu theo các tiêu chí cho trước(chi phí, khoảng cách, thời gian,…). Với cách này thì có thể ở bước cuối cùng ta không còn gì để chọn mà phải chấp nhận một giá trị cuối cùng còn lại. Áp dụng kỹ thuật tham lam sẽ cho một giải thuật thời gian đa thức.

Mô tả tổng quát của thuật toán:

```
Function thamlam(c: tập hợp);
    Begin
        {s: tập các đại diện được chọn là lời giải của bài toán}
        s:= ∅ ;
        While (s chưa có lời giải) and (C <> ∅ ) do
            Begin
                chọn x thuộc C nhờ hàm chọn
                C:=C\{x};
                IF S ∪ {x} có triển vọng then
                    S:= S ∪ {x};
            End;
        IF S có lời giải then return S
        Else Return 0;
    End;
```

Lưu ý:

- Thuật toán tham lam không phải lúc nào cũng cho lời giải tối ưu.
- Khi sử dụng thuật toán ta cần chứng minh thuật toán cho lời giải tối ưu.

I GIỚI THIỆU MỘT SỐ BÀI TOÁN

Bài toán 1 : Bài tóan đổi tiền

Cho 5 loại tiền 1, 2, 5, 10, 20 đồng. Cần đổi 1 số tiền là S sao cho số tờ cần dùng là ít nhất.

.1.1 Mô tả thuật toán:

Cho s là số tiền nhập vào. Kiểm tra giá trị của số tiền:

Bước 1: Nếu giá trị tiền >=20 thì số tờ giấy loại 20 đồng là s div 20. Thay đổi giá trị của s=s div 20, tiếp tục kiểm tra các bước tiếp theo.

Bước 2: Nếu giá trị tiền >=10 thì số tờ giấy loại 10 đồng là s div 10. Thay đổi giá trị của s=s div 10, tiếp tục các bước tiếp theo.

Bước 3: Nếu giá trị tiền >=5 thì số tờ giấy loại 5 đồng là s div 5. Thay đổi giá trị của s=s div 5, tiếp tục các bước tiếp theo.

Bước 4: Nếu giá trị tiền >=2 thì số tờ giấy loại 2 đồng là s div 2. Thay đổi giá trị của s=s div 2, tiếp tục các bước tiếp theo.

Bước 5: Nếu giá trị tiền >=1 thì số tờ giấy loại 1 đồng là s.

.1.2 Chương trình:

```
use crt;
var n: integer;
 Procedure Doitien(s: integer);
                Begin
                    if s>=20 then
                            begin
                                    writeln('Số tờ loại 20 là ', s div 20);
                                    s:=s div 20;
                             end;
                     if s>=10 then
                            begin
                                    writeln('Số tờ loại 10 là ', s div 10);
                                     s:=s div 10;
                            end;
   if s>=5 then
                            begin
                                     writeln('Số tờ loại 5 là ', s div 5);
                                      s:=s div 5;
                            end;
            if s>=2 then
                            begin
                                     writeln('Số tờ loại 2 là ', s div 2);
```

```
                    s:=s div 2;
                end;
  if s>=1 then
                    writeln('Số tờ loại 1 là ', s );
      End;
Begin
      Write (' Nhap vao so tien can doi:'); readln (n);
      Doitien(n);
      Readln;
End.
```

.1.3 Test chương trình:

Cho số tiền 56 đồng.

số tờ giấy 20 đồng là 2 tờ

số tờ giấy 10 đồng là 1 tờ

số tờ giấy 5 đồng là 1 tờ

số tờ giấy 1 đồng là 1 tờ

Bài toán 2 : Bài Toán Trộn Tệp

Viết chương trình trộn 2 tệp văn bản có chứa các số nguyên được sắp tăng thành một tệp văn bản chứa các số nguyên được sắp tăng.

.1.4 Mô tả thuật toán:

Gọi f, g là 2 tệp được nhập vào, h là tệp chứa kết quả đã xếp tăng từ 2 tệp f và g.

Bước 1: Đọc1 phần tử đầu tiên (nếu có) từ tệp f vào biến x và 1 phần tử đầu tiên (nếu có) của tệp g vào biến y.

Bước 2: So sánh x với y để chọn một phần tử nhỏ rồi ghi vào tệp kết quả h.

Bước 3: Sau đó ta đọc tiếp một phần tử từ tệp cung cấp phần tử được chọn vào biến tương ứng rồi lặp lại thủ tục chọn phần tử nhỏ nói trên. Qúa trình này kết thúc khi nào một trong 2 tệp f hoặc g được duyệt xong.

.1.5 Chương trình:

```
uses crt;
const BL = ' '; {Dấu cách}
const fn  = ' Tep1.txt' ;
            gn = ' Tep2.txt';
            hn = ' Tep3.txt';
procedure Sinh(fn : string ; n :integer) ;
                  var f : text ;
                        i, x : integer ;
```

```
                begin
                        assign(f,fn) ; rewrite(f) ;
                        x := random(n) ; write(f,x,BL) ;
                        for i  := 2 to n do
                                begin
                                        x := x + random(n) ;
                                        write(f,x,BL) ;
                                        if i mod 10 = 0 then writeln(f) ;
                                end ;
                        close(f) ;
                end ;
procedure Tron2tep(fn,gn,hn : string) ;
                var f,g,h : text ;
                        ef, eg : Boolean ;
                        x, y, d : integer ;
        begin
                        assign(f,fn) ;
                        assign(g,gn) ; reset(g) ;
                        assign(f,fn) ; rewrite(h);
                        ef := SeekEof(f) ;
                        if NOT ef then read(f,x);
                        eg := SeekEof(g) ;
                        if NOT ef then read(g,y);
                        d := 0;
         while (NOT ef) AND (NOT eg) do
                        if x< y then
                                begin
                                        inc(d) ;
                                        write(h,x,BL) ;
                                        if d mod 10 = 0 then writeln(h) ;
                                        if NOT ef then read(f,x) ;
                        end else
                                begin
                                        inc(d) ;
                                        write(h,y,BL) ;
                                        if d mod 10 = 0 then writeln(h) ;
                                        eg := SeekEof(g) ;
                                        if NOT ef then read(g,y) ;
                        end ;
         while (NOT ef) do
                        begin
                                        inc(d) ;
                                        write(h,x,BL) ;
                                        if d mod 10 = 0 then writeln(h) ;
                                        ef := SeekEof(f) ;
                                        if NOT ef then read(f,x) ;
                        end ;
         while (NOT eg) do
                        begin
```

```
                                    inc(d) ;
                                    write(h,y,BL) ;
                                if d mod 10 = 0 then writeln(h) ;
                                    eg := SeekEof(g) ;
                                    if NOT ef then read(g,y) ;
                        end ;
            close(f) ; close(g) ; close(h) ;
             if hn = fn then erase(f)
                        else if hn = gn then erase(g) ;
             rename(h,hn) ;
           end ;
BEGIN
                        Randomize ;
                        Sinh(fn,100) ;
                        Sinh(gn,200) ;
                        Tron2tep(fn,gn,hn) ;
                Writeln('OK') ; readln ;
END.
```

.1.6 Ta cần lưu ý mấy điểm sau đây

Khi đọc xong phần tử cuối cùng của một tệp thì tệp đó chuyển sang trạng thái kết thúc (EOF) do đó nếu ta tổ chức vòng lặp WHILE trong thủ tục trộn 2 tệp theo điều kiện NOT EOF(f) AND NOT EOF(g) thì phần tử cuối của các tệp đó sẽ không đươc so sánh trong khi ta muốn tôn trọng nguyên tắc giải phóng. Điều này được thực hiện bằng nguyên tắc săn đuổi như sau:

- Dùng biến logic ef ghi nhận trạng thái hết tệp f trước một nhịp. Điều đó có nghĩa khi ef= FALSE biến x vẫn đang chứa giá trị chưa xử lí (chưa so sánh với y và do đó chưa được ghi vào tệp h). Chú ý rằng dù ef=FALSE nhưng có thể ta vẫn có EOF(f) =TRUE. Một biến eg tương tự cũng tạo được cho tệp g.

.1.7 Test chương trình:

Cho 2 tệp sắp tăng được nhập vào như sau:
Tệp f: 3 4 5 6 6 7 8 30
Tệp g: 1 3 3 8 9 10 12 40
Tệp kết quả: 1 3 3 3 4 5 6 6 7 8 9 10 12 30 40

Bài toán 3: Bài toán băng nhạc

.1.8 Mô tả bài toán

Có n bài hát, với chiều dài mỗi bài tính theo phút được biết trước cần được ghi vào một băng nhạc có dung lượng đủ chứa toàn bộ các bài đã cho. Băng sẽ được lắp vào một máy phát nhạc tự động theo yêu cầu của mỗi người nghe. Tính trung bình, các bài hát trong

một ngày được người nghe lựa chọn với số lần như nhau. Để phát bài thứ i trên băng, máy phải quay để bỏ qua i-1 bài ghi trước đó. Thời gian quay băng bỏ qua mỗi bài và thời gian phát bài đó được tính là như nhau. Hãy tìm cách ghi các bài trên băng sao cho tổng thời gian quay băng trong mỗi ngày là thấp nhất.

.1.9 Dữ liệu vào và ra

Dữ liệu vào được ghi trong tệp văn bản tên **'Bangnhac.inp'**:

Dòng đầu tiên ghi số lượng bài hát, dòng tiếp theo ghi chiều dài mỗi bài. Mỗi đơn vị dữ liệu cách nhau bởi dấu cách. Dữ liệu Test trong tệp BANGNHAC.INP:

```
10
5 2 4 5 3 2 4 4 6 3
```

cho biết có 10 bài hát, bài thứ nhất có chiều dài 5 phút, …

Dữ liệu ra được ghi trong tệp văn bản tên **'Bangnhac.out'**:

Trật tự ghi bài trên băng và thời gian tìm và phát bài đó theo trật tự này. Dòng cuối cùng ghi tổng số thời gian sử dụng băng nếu mỗi bài được phát một lần.

Kết quả Test trong tệp BANGNHAC.OUT:

```
Bài thứ 2: thời gian tìm và phát = 2
Bài thứ 6: thời gian tìm và phát = 4
Bài thứ 5: thời gian tìm và phát = 7
Bài thứ 10: thời gian tìm và phát = 10
Bài thứ 3: thời gian tìm và phát = 14
Bài thứ 7: thời gian tìm và phát = 18
Bài thứ 8: thời gian tìm và phát = 22
Bài thứ 1: thời gian tìm và phát = 27
Bài thứ 4: thời gian tìm và phát = 32
Bài thứ 9: thời gian tìm và phát = 38
Tổng thời gian sử dụng băng = 174
```

.1.10 Giải thuật:

Trước khi trình bày ý tưởng giải thuật, ta xét ví dụ sau:

Giả sử ta có 3 bài hát với số phút lần lượt như sau: a = (6,3,5); n=3. Ta xét vài tình huống ghi băng để rút ra kết luận cần thiết.

Trật tự ghi	Thời gian tìm và phát mỗi bài theo trật tự này
1 2 3	(6) + (6+3) + (6+3+5) = 29 phút
1 3 2	(6) + (6+5) + (6+5+3) = 33 –
2 1 3	(3) + (3+6) + (3+6+5) = 26 –
2 3 1	(3) + (3+5) + (3+5+6) = 25 – (phương án tối ưu)
3 1 2	(5) + (5+6) + (5+6+3) = 30 –
3 2 1	(5) + (5+3) + (5+3+6) = 27 –

Vậy phương án tối ưu sẽ là (2, 3, 1): ghi bài 2 rồi đến bài 3, cuối cùng ghi bài 1. Tổng thời gian theo phương án này là 25 phút.

Để có được phương án tối ưu ta chỉ cần ghi băng theo trật tự tăng dần của chiều dài các bài hát. Bài toán được cho với giả thiết băng đủ lớn để ghi được toàn bộ các bài.

Bước 1: Sắp xếp các bài hát theo thứ tự tăng dần của thời gian phát.

Bước 2:

Tính thời gian tìm và phát cho lần lượt từng bài theo thứ tự đã sắp (tăng).

Tính tổng thời gian sử dụng băng = Tổng thời gian tìm và phát n bài.

Trường hợp dung lượng băng hạn chế trong M phút: Ta phải tính tổng thời gian sử dụng băng từ bài đầu tiên đến bài thứ i (i = 1.. n); So sánh tổng này với M:

Nếu Tổng <= M, bài 1 đến bài i được chọn, tiếp tục tính cho bài thứ i+1 nếu i còn bé hơn n.

Nếu Tổng > M: bài thứ i không được chọn, ngừng tại đây.

Chương trình:

```
uses crt;
const   MN = 200;
        fn = 'Bangnhac.inp';
        gn = 'Bangnhac.out';
var     a, id: array[1..MN] of integer;
        f, g: text;
        n: integer;
procedure Doc;
   var  i, k: integer;
```

```
begin
        assign(f, fn);
        reset(f);
        read(f, n)
        for i:= 1 to n do read(f, a[i]);
        close(f);
end;
procedure  InitID;
   var i: integer;
begin
   for i:=1 to n do id[i]:=i;
end;
procedure IDQuickSort(d,c: integer);
   var i, j, x, k: integer;
begin
   i:=d; j:=c;
   x:=a[id[(i+j) div 2]];
   while i<=j do
        begin
          while a[id[i]]< x do  inc(i);
          while a[id[j]]> x do  dec(j);
           if  i<=j then
                begin
                  k:=id[i]; id[i]:=id[j]; id[j]:=k;
                  inc(i);  dec(j);
                end;
        end;
        if d<j then IDQuickSort(d, j);
        if i<c then IDQuickSort(i, c);
end;
procedure Test;
   var i, t: integer;
        tt: longint;
begin
        clrscr;
        Doc;
        InitID;
        IDQuickSort(1, n);
        assign(g, gn); rewrite(g);
        t:=0; tt:=0;
        for i:=1 to n do
            begin
                t:= t + a[id[i]];
                tt:= tt + t;
                writeln(g,'Bai thu ', id[i], ':  thoi gian tim va phat = ',t);
            end;
        writeln(g, 'Tong thoi gian su dung bang = ', tt);
    close(g);
    write('OK'); readln;
```

```
end;
        BEGIN
                Test;
        END.
```

Trường hợp dung lượng băng hạn chế trong M phút, *procedure Test* được sửa lại như sau:

```
procedure Test;
  Const M = 150;
   var i, t: integer;
        tt,: longint;
begin
        clrscr;
        Doc;
        InitID;
        IDQuickSort(1, n);
        assign(g, gn); rewrite(g);
        t:=0; tt:=0;
        for i:=1 to n do
            begin
                t:= t + a[id[i]];
                tt:= tt + t;
                If  (tt > M)  then  begin
tt:= tt - t;
exit;
                                                    end;
writeln(g,'Bai thu ', id[i], '  : thoi gian tim va phat =',t);
                end;
        writeln(g, 'Tong thoi gian su dung bang = ', tt);
    close(g);
    write('OK'); readln;
end;
        BEGIN
                Test;
        END.
```

Bài toán 4: Bài toán chọn việc

.1.11 Mô tả bài toán

Có n công việc cần thực hiện trên một máy tính, mỗi việc đòi hỏi đúng 1 giờ chạy máy. Với mỗi việc ta biết thời hạn phải nộp kết quả thực hiện sau khi chạy máy và tiền thù lao thu được nếu nộp kết quả đúng hạn. Chỉ có một máy tính trong tay, hãy lập lịch thực hiện các công việc để thực hiện trên máy sao cho tổng số tiền thu được là lớn nhất.

Dữ liệu vào và ra

Dữ liệu vào file “Viec.inp”:

Dữ liệu vào được ghi trong một tệp văn bản tên ‘Viec.inp’, mỗi việc được mô tả bằng 2 số nguyên dương: Số thứ nhất là thời hạn giao nộp, số thứ hai là thù lao được hưởng. Các số cách nhau bởi dấu cách. Ví dụ

```
2 100 1 10 2 15 1 27 2 50 3 10 5 20 3
5
```

cho biết việc thứ nhất phải nộp vào lúc 2 giờ, thù lao sẽ nhận là 100,…Trong tệp không ghi số lượng công việc. Thời điểm bắt đầu ca làm việc là 0 giờ.

Dữ liệu ra file **“Viec.out”**:

Với dữ liệu vào là file ‘Viec.inp’ ở trên thì lịch thực hiện được ghi trong một tệp văn bản tên ‘Viec.out’:

```
Gio thu 1 thuc hien viec 5
Gio thu 2 thuc hien viec 1
Gio thu 3 thuc hien viec 6
Gio thu 4 thuc hien viec 7
Tong so tien thu duoc 180
```

(dòng cuối cùng của ‘Viec.out’ ghi số tiền thu được).

3. Giải Thuật

Ý tưởng giải thuật

Ta sẽ ưu tiên cho những việc có thù lao cao do đó ta sắp các việc giảm dần theo tiền thù lao. Dùng một mảng làm chỉ số cho mảng a để đảm bảo rằng sau khi sắp xếp thứ tự của các công việc không thay đổi so với trong tệp ‘Viec.inp’.

Với mỗi việc k ta đã biết thời hạn giao nộp việc đó là h=t[k]. Ta xét trục thời gian b. Nếu giờ h trên trục đó đã bận do việc khác thì ta tìm từ thời điểm h trở về trước một thời điểm thực hiện việc k đó. Nếu tìm được một thời điểm m như vậy, ta đánh dấu bằng mã số của việc đó trên trục thời gian b, b[m]:=k.

Sau khi xếp việc xong, có thể trên trục thời gian còn những thời điểm rỗi, ta dồn chúng về phía trước nhằm thu được một lịch làm việc trù mật, tức là không có giờ trống. Ví dụ, nếu b= (3, 0, 4, 0, 1), tức là:

giờ thứ 1 thực hiện việc 3

giờ thứ 2 máy rỗi

giờ thứ 3 thực hiện việc 4

giờ thứ 4 máy rỗi

giờ thứ 5 thực hiện việc 1

ca làm việc kéo dài 5 tiếng, trong đó chỉ sử dụng máy 3 tiếng, thì sau khi dồn mảng ta có b= (3, 4, 1) ứng với lịch làm việc như sau:

giờ thứ 1 thực hiện việc 3

giờ thứ 2 thực hiện việc 4

giờ thứ 3 thực hiện việc 1

ca làm việc kéo dài 3 tiếng.

- Chương trình sẽ không chọn những việc không có thù lao nếu như không thể xếp được vào ca làm việc.

Giải thuật chi tiết

Đọc dữ liệu từ tệp ra, mảng a ghi thù lao của các việc, mảng t ghi thời hạn giao nộp của các việc tương ứng. Lấy giờ cuối ca hmax =max(t[i]), với i=1...n

Sắp xếp việc giảm dần theo thù lao: Mảng chỉ số cho mảng a là id, khởi tạo id[i]=i với i=1..n

Thuật toán QuickSort: (d=1, c=n)

+ B1:Cho i=d, j=c, x:=a[id[(i+j) div 2]]. Trong khi i<=j thì nếu a[id[i]]>x thì inc(i), nếu a[id[j]]<x thì dec(j).

+ B2: Nếu d<j thì lặp lại B1 với d=d, c=j

Nếu i<c thì lặp lại B1 với d=i, c=c.

Xếp việc:

+ Dùng mảng b làm trục thời gian. Khởi tạo b[i]=0 với i=0..hmax

+ Vòng lặp i=1→n.

Lấy chỉ số việc có thù lao cao thứ i: k=id[i]

Lấy thời hạn giao nộp tương ứng: h= t[k]

Trong khi b[h] <>0 (tức là đã có việc được chọn làm ở thời giờ h ta trở về trước một thời điểm có thể thực hiện được việc k này): dec(h)

Nếu h>0 (có thể làm được việc k tại giờ h) thì đánh dấu trên trục thời gian: b[h]=k.

Dồn việc:

+ Bắt đầu k=1

+ Trong khi k<=hmax và b[k] <>0 thì inc(k)

+ Nếu k> hmax, nghĩa là trong trục thời gian b không có thời điểm rỗi thì kết thúc dồn việc tại đây.

+ Ngược lại, vòng lặp i=k+1→hmax:

Nếu b[i] <>0 thì gán b[k]=b[i] và inc(k)

Sau khi loại bỏ hết các giờ rỗi phải gán lại hmax=k-1 vì k-1 mới chính là giờ cuối ca.

Ghi tệp: Ghi vào tệp g nội dung như sau:

Writeln(g,'Gio thu ' , i, 'thuc hien viec', b[i]);

Chương trình

```
Program ChonViec;
Uses crt;
Const    MN=200;
     Fn= 'Viec.inp';
     Gn= 'Viec.out';
Var  a,id,t: array [1..MN] of integer; {với a là thù lao, t: thời
hạn giao nộp}
     b: array [0..MN] of integer;
     n: integer; {số việc}
     hmax: integer; {giờ cuối ca. Ta quy định giờ đầu ca là 0}
     f,g: Text;
Procedure Doc;
     Var i,k: integer;
     Begin
          Assign(f, Fn);
          Reset(f);
          n:=0; hmax:=0;
```

```
          While not SeekEof(f) do
          begin
          inc(n);
          read(f,t[n],a[n]);
          if hmax<t[n] then hmax:=t[n];
          end;
          close(f);
     End;
Procedure InitID;
     Var i:integer;
     Begin
          For i:=1 to n do id[i]:=i;
     End;
```
{Sắp xếp mảng a giảm dần bằng phương pháp sắp xếp nhanh. Ở đây chỉ sắp giá trị của các phần tử mảng còn chỉ số của các phần tử vẫn giữ nguyên}
```
Procedure IDQuickSort(d,c:integer);
     Var i,j,x,k:integer;
     Begin
          i:=d; j:=c;
          x:=a[id[(i+j) div 2]];
          While i<=j do
               Begin
                    While a[id[i]]>x do  inc(i);
                    While a[id[j]]<x do dec(j);
                    If i<=j then
                        begin
                         k:= id[i]; id[i]:=id[j]; id[j]:=k;
                         inc(i); dec(j);
                          end;
               End;
          If d<j then IDQuickSort(d,j);
          If i<c then IDQuickSort(i,c);
     End;
Procedure XepViec;
     Var i,k,h:integer;
     Begin
          For i:=0 to hmax do b[i]:=0;
          For i:=1 to n do
               Begin
                    k:=id[i]; {Việc nào}
                    h:=t[k]; {Thời gian giao nộp}
                    while b[h]<>0 do dec(h);
                    if h>0 then b[h]:=k;
               End;
     End;
Procedure DonViec;
     Var i,k:integer;
     Begin
          k:=1;
```

```
          While (k<=hmax) and (b[k]<>0) do inc(k);
          If k>hmax then exit;
          For i:=k+1 to hmax do
               If b[i]<>0 then
                   begin
                    b[k]:=b[i];
                    inc(k);
                   end;
          hmax:=k-1;
     End;
Procedure GhiTep;
     Var i:integer;
          c:longint; {Tổng số tiền thu được}
     Begin
          c:=0;
          assign(g,Gn);
          rewrite(g);
          for i:=1 to hmax do
               begin
                    writeln('Gio thu ',i,'  thuc hien viec ',
b[i] );
                    c:=c+a[b[i]];
               end;
          writeln('Tong so tien thu duoc: ',c);
          close(g);
     End;
Procedure Test;
     Begin
          Clrscr;
          Doc;
          InitID;
          IDQuickSort(1,n);
          XepViec;
          DonViec;
          GhiTep;
          Write('Test xong chuong trinh!');
          Readln;
     End;
BEGIN
     Test;
END.
```

Bài toán 5: Bài toán cái ba lô(Knapsack Problem)

Mô tả bài toán

Cho một cái ba lô có thể đựng một trọng lượng W và n loại đồ vật, mỗi đồ vật i có một trọng lượng g_i và một giá trị v_i. Tất cả các loại đồ vật đều có số lượng không hạn chế(có nghĩa là có thể chọn tùy ý số lượng một vật). Tìm một cách lựa chọn các đồ vật đựng

vào ba lô, chọn các loại đồ vật nào, mỗi loại lấy bao nhiêu sao cho tổng trọng lượng không vượt quá W và tổng giá trị là lớn nhất.

Phân tích bài toán

Theo yêu cầu của bài toán thì ta cần những đồ vật có giá trị cao mà trọng lượng lại nhỏ để sao cho có thể mang được nhiều "đồ quý". Từ đó ta quan tâm đến yếu tố "đơn giá" của từng loại đồ vật tức là tỷ lệ giá trị/trọng lượng. Đơn giá của một đò vật càng cao thì đồ vật đó càng quí.

Tiêu chí ưu tiên chọn các vật vào Balo ở đây là các đồ vật có đơn giá càng cao thì ưu tiên được chọn trước. Từ đó ta có kỹ thuật greedy áp dụng cho bài toán Balô như sau:

1: Tính đơn giá cho các loại đồ vật (= giá trị/trọng lượng).

2. Xét các loại đồ vật theo thứ tự đơn giá từ lớn đến nhỏ.

3. Với mỗi đồ vật được xét sẽ lấy một số lượng tối đa mà trọng lượng còn lại của ba lô cho phép.

4. Xác định trọng luợng còn lại của ba lô và quay lại bước 3 cho đến khi không còn có thể chọn được đồ vật nào nữa.

Minh hoạt giải thuật:

Ta có một ba lô có trọng lượng là 37 và 4 loại đồ vật với trọng lượng và giá trị tương ứng được cho trong bảng sau:

Loại đồ vật	**Trọng lượng**	**Giá trị**
A	15	30
B	10	25
C	2	2
D	4	6

Từ bảng đã cho ta tính đơn giá cho các loại đồ vật và sắp xếp các loại đồ vật này theo thứ tự đơn giá giảm dần ta có bảng sau:

Loại đồ vật	**Trọng lượng**	**Giá trị**	**Đơn giá**
B	10	25	2.5
A	15	30	2.0
D	4	6	1.5
C	2	2	1.0

Theo đó thì thứ tự ưu tiên để chọn đồ vật là là B, A, D và cuối cùng là C.

Vật B được xét đầu tiên và ta chọn tối đa 3 cái vì trọng lượng mỗi cái là 10 và ba lô có trọng lượng 37.

Sau khi đã chọn 3 vật loại B, trọng lượng còn lại trong ba lô là 37 - 3*10 = 7.

Ta xét đến vật A, vì A có trọng lượng 15 mà trọng lượng còn lại của balô chỉ còn 7 nên không thể chọn vật A.

Xét vật D và ta thấy có thể chọn 1 vật D, khi đó trọng lượng còn lại của ba lô là 7-4 = 3.

Cuối cùng ta chọn được một vật C.

Như vậy chúng ta đã chọn 3 cái loại B, một cái loại D và 1 cái loại C. Tổng trọng lương là 3*10 + 1*4 + 1*2 = 36 và tổng giá trị là 3*25+1*6+1*2 = 83.

Chương trình

```
Uses crt;
const mn=200;
fn='balo.inp';
gn='balo.out';
var
a,v,id : array[1..mn] of integer;
{Mang a chua trong luong; mang v chua gia tri; mang id chua so hieu cac mat hang}
gdv : array[1..mn] of real;
{ Mang gdv chua don gia cua ca mat hang}
f,g:text;   { hai file dau vao va ra}
w,n:integer; {n: so mat hang; m: trong luong balo}
procedure doc; {thu tuc doc du lieu tu file balo.inp}
var i,k:integer;
Begin
assign(f,fn);
reset(f);
readln(f,n,w); { so dau tien chua so do vat (n); so tiep theo chua trong luong ba lo (w)}
For i:=1 to n do
   begin
   read(f,a[i], v[i]); { doc va ghi vao file trong luong va gia tri cua mat hang thi i}
   gdv[i]:= v[i]/a[i];
   end;
close(f);
end;
procedure initID; {thu tuc khoi tao mang ID}
var i:integer;
begin
```

```
for i:=1 to n do id[i]:=i
end;
procedure IDQuicksort(d,c:integer);{ Sap xep cac mat hang theo thu tu uu tien chon tuc la
tang dan theo don gia}
var i,j,k:integer;
x:real;
begin
i:=d;
j:=c;
x:=gdv[id[(i+j) div 2]];
while i<=j do
 BEGIN
     while gdv[id[i]] > x do inc(i);
     while gdv[id[j]]<x do dec(j);
     if i<=j then
       begin
           k:=id[i];
           id[i]:=id[j];
           id[j]:=k;
       end;
     end;
     if d< j then IDquicksort(d,j);
     if i< c then IDquicksort(i,c);
     end;

     PROCEDURE XEPBALO;
     var
     i,j:integer;
     t,tam:integer; { tong trong luong con xep duoc vao ba lo}
     tt:real; {tong gia tri da lay}
     BEGIN
      assign(g,gn);
      rewrite(g);
      i:=1; tt:=0; t:=w;
      while (i<=n) and ( t>=0) do
           begin
             tam:=0; {Chua trong luong can lay o moi vat}
             j:=i;
             {Voi moi vat lay toi da trong luong co the}
             while A[id[j]]<=t Do
             Begin
                 Tam:=Tam+A[id[i]];
                 t:=t-A[id[i]];
                 tt:=tt+v[id[i]];
             End;
             If Tam>0 then
               Writeln(g,'Lay  vat', j , ' Voi trong luong la:', Tam);
               i:=i+1;
           End;
```

```
If (i>n) and (t=w) then
Writeln('Khong co phuong an nao!')
Else
writeln(g,'Tong gia tri la:', tt:5:5);
close(g);
readln;
end;
PROCEDURE TEST;
BEGIN
 doc;
initID;
IDquicksort(1,n);
xepbalo;
end;
BEGIN {main program}
test;
end.
```

Bài toán 6: Cây bao trùm tối thiểu (Minimum Spanning Trees)

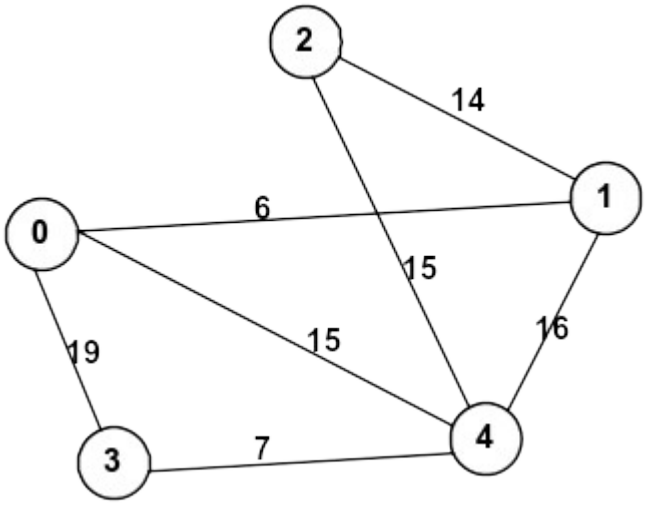
Mô tả bài toán

Giả sử ta có một đồ thị liên thông vô hướng G = (V, E), V là tập hợp các đỉnh (Vertices), E là tập hợp các cạnh (Edges).

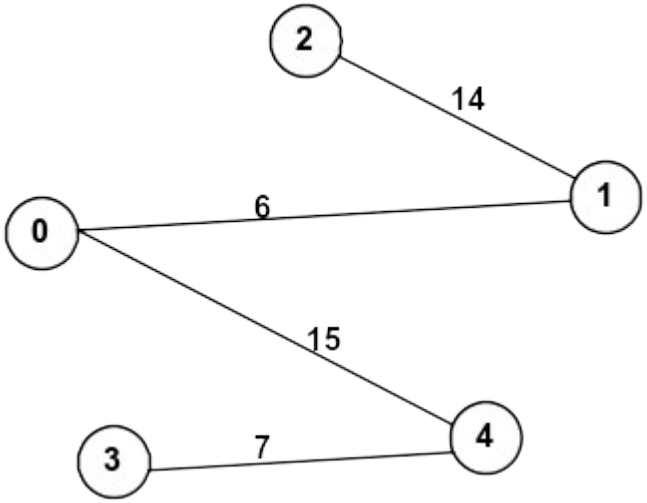
Bài toán tìm cây bao trùm tối tiểu (MST) là tìm một tập hợp T chứa các cạnh của G sao cho V cùng với tập hợp cạnh này cũng là một đồ thị liên thông, tức (V,T) là một đồ thị liên thông. Hơn nữa tổng độ dài các cạnh trong T là nhỏ nhất có thể được.

Các vid dụ của bài toán này trong thực tế là bài toán thiết lập mạng truyền thông, ở đó các đỉnh là các thành phố còn các cạnh của cây bao trùm là đường nối mạng giữa các thành phố, hay bài toán xây dựng hệ thống đường sắt nối liền các thành phố.

Ví dụ: Cho đồ thị liên thông có trọng số như sau:



Thì các cây bao trùm tối thiểu tìm được là :



Với tổng trọng số là 42.

.1.12 Tư tưởng chính của giải thuật Kruskal

- Duyệt các cạnh từ trọng số nhỏ đến lớn
- Chọn các cạnh khi ghép vào cây nó không tạo thành chu trình
- Kết quả cuối cùng sẽ là cây MST được tạo từ n-1 cạnh.

Giải thuật Kruskal được trình bày như sau:

E(1) tập hợp các cạnh của cây MST.

E(2) tập hợp các cạnh của đồ thị.

Begin Of Algorithm

E1=∅;E2=E;

Num_edges=0;

While Num_edges < N – 1 do

Chọn 2 đỉnh V(i) V(j) sao cho trọng số E2(ij) nhỏ nhất

If V(i),V(j) không tạo thành chu trình then

```
                    E1=E1∪(i,j)
                end (If)
                inc(Num_edges);
            end (While)
        End Of Algorithm.
```

Áp dụng giải thuật Kruskal cho đồ thị trên ta có thứ tự chọn được các cạnh cho MST như sau:

- Đầu tiên khởi tạo T=Ø
- Tiếp theo ta có cạnh (0,1)= 6 , T={(0,1)}
- Tiếp theo ta có cạnh (3,4)= 7 nhỏ , T={(0,1),(3,4)}
- Tiếp theo ta có cạnh (1,2)= 14, T={(0,1),(3,4),(1,2)}
- Tiếp theo ta có cạnh (0,4)= 15 , T={(0,1),(3,4),(1,2)}
- Vậy cây bao trùm cần tìm là T={(0,1),(3,4),(1,2)}, với tổng trong số là 42

Cài đặt giải thuật

Để giải bài toán này ta biểu diễn đồ thị dưới dạng một ma trận kề, trong đó phần tử E(i,j) là trong số của cạnh nối đỉnh i với đỉnh j. Vì đồ thị là vô hướng nên ma trận biểu diễn nó đối xứng qua đường chéo chính.

Chương trình nguồn được cài đặt bằng ngôn ngữ C, tệp dữ liệu đầu vào và tệp dữ liệu kết quả như sau:

input.dat

```
0,6,0,19,15
6,0,14,0,16
0,14,0,0,15
19,0,0,0,7
15,16,15,7,0
```

output.dat

```
THUAT TOAN TIM CAY BAO TRUM TOI THIEU (MST)
Cac cap canh do thi (da sap xep):

  0   1   6
  3   4   7
  1   2  14
  0   4  15
  2   4  15
  1   4  16
  0   3  19

Cac cap canh trong MST la:
  E(0,1) = 6
  E(3,4) = 7
  E(1,2) = 14
  E(0,4) = 15
Tong chieu dai cua MST = 42
```

```
/* Bao cao mon Giai Thuat nang cao */
/* Thuat toan CAY BAO TRUM TOI THIEU (MST) */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define CN 50                   /* So dinh cua do thi */
#define CM 150                  /* So canh cua do thi */

int N,M;
int U[CN];

/* Khai bao cac ham */
void makeset (int i);
int find (int i);
void merge (int p, int q);
int equal (int p, int q);
void initial (int n);
void test_univ (void);
int readfile(FILE *in,int W[CN][CN]);

int main()
{
  FILE *in,*out;
  int G[CN][CN];
  int E1[CN-1][3];              /* Tap cac canh trong MST */
  int E2[CM][3];                /* Tap cac canh cua do thi */
  int num_edges = 0;            /* So canh trong MST */
  int next_edge = 0;            /* Canh ke tiep chua xet */
  int weight = 0;               /* Trong so */
  int a, b, c, i, j, k;         /* Cac bien trung gian */

  readfile(in,G);               /* Doc do thi tu file vao ma tran*/

  /* Khoi tao tap cac canh cua do thi*/
  k = 0;
  for (i = 0; i < N-1; i++)
    for (j = i+1; j < N; j++)
        if ((G[i][j]) >0)
        {  E2[k][0] = i;         /* Dinh dau cua canh */
           E2[k][1] = j;         /* Dinh ke tiep cua canh */
           E2[k][2] = G[i][j];   /* Trong so cua canh */
           k++;
        }
  M=k;
```

```
/* Sap xep cac canh (theo chieu khong giam) - bubblesort */
  for (i = M - 1; i > 0; i--)
    for (j = 0; j < i; j++)
          if (E2[j][2] > E2[j+1][2])
          {  a = E2[j][0];
             b = E2[j][1];
             c = E2[j][2];
             E2[j][0] = E2[j+1][0];
             E2[j][1] = E2[j+1][1];
             E2[j][2] = E2[j+1][2];
             E2[j+1][0] = a;
             E2[j+1][1] = b;
             E2[j+1][2] = c;
          }

  initial (N);

/* Khoi tao cac canh trong MST */
  for (i = 0; i < N - 1; i++)
    for (j = 0; j < 3; j++)
          E1[i][j] = -1;          /* '-1' nghia la rong */

/* Tim cay MST */
  while (num_edges < N - 1)
  {  a = E2[next_edge][0];
    b = E2[next_edge][1];

    i = find(a);
    j = find(b);

    if (!equal(i, j))
    {  merge (i, j);
          E1[num_edges][0] = E2[next_edge][0];
          E1[num_edges][1] = E2[next_edge][1];
          E1[num_edges][2] = E2[next_edge][2];
          num_edges++;
    }

    next_edge++;
  }

/* Ghi ket qua chuong trinh vao tep ket qua */
  if ((out = fopen("output.dat", "wt")) == NULL)
  {
    fprintf(stderr, "Cannot open input file.\n");
    return 1;
  }
```

```
/* Hien thi cac cap canh sau khi sap xep */
  fprintf(out,"%s\n","    THUAT TOAN TIM CAY BAO TRUM TOI THIEU (MST)");
  fprintf(out," %3s\n\n","Cac cap canh do thi (da sap xep):");
  for (i = 0; i < M; i++)
  {  for (j = 0; j < 3; j++)
          fprintf(out,"  %3d", E2[i][j]);
     fprintf(out,"\n");
  }

/* Hien thi cac cap canh trong MST */
  fprintf(out,"\nCac cap canh trong MST la:\n");
  for (i = 0; i < N - 1; i++)
  {  fprintf(out,"   E(%d,%d) = %d\n", E1[i][0], E1[i][1], E1[i][2]);
     weight = weight + E1[i][2];
  }
  fprintf(out,"Tong chieu dai cua MST = %d\n", weight);

  return (0);
}

/* Khoi tao tap hop dinh */
void makeset (int i)
{  U[i] = i;
}

/* Tim kiem */
int find (int i)
{  int j;
  j = i;
  while (U[j] != j)
     j = U[j];
  return (j);
}

/* Noi hai dinh */
void merge (int p, int q)
{  if (p < q)
     U[q] = p;
  else
     U[p] = q;
}

/* So sanh bang */
int equal (int p, int q)
{  if (p == q)
     return (1);
  else
     return (0);
}
```

```
/* Khoi tao */
void initial (int n)
{  int i;
   for (i = 0; i < n; i++)
      makeset(i);
}

/* Doc do thi tu file input.dat vao ma tran ke */
int readfile(FILE *in,int W[CN][CN])
{
   int i,j;
   char *vc,*vt;
   div_t x;

   if ((in = fopen("input.dat", "rt")) == NULL)
   {
      fprintf(stderr, "Cannot open input file.\n");
      return 1;
   }

   i=0;
   while (!feof(in))
   {
     fgets(vc,100,in);
     x=div(strlen(vc)+1,2);
     N=x.quot;
     vt=strtok(vc,",");
     for (j=0;j<N;j++)
       {
            W[i][j]=atoi(vt);
            vt=strtok(NULL,",");
       }
     i++;
    }

   fclose(in);
   return 0;
}
```

II MỘT SỐ BÀI TOÁN

Bài số 1: Bài toán người qua cầu (Tuyển MicroSoft)

.1.13 Phát biểu bài toán:

Một đàn con nít đi chơi đêm giao thừa với 1 chiếc đèn lồng. Gặp một cây cầu mỏng manh. Các em phải đưa nhau qua cầu. Cầu yếu, chỉ chịu được sức nặng của 2 em mỗi lượt,

ngoài ra, cầu trơn và gập gềnh nên buộc phải soi đèn mới đi được. Mỗi em qua cầu với thời gian khác nhau. Hãy tìm cách để các em qua cầu nhanh nhất. Giả thiết cho n em với thời gian vượt cầu của mỗi em lần lượt là t1, t2,…,tn.

.1.14 Ví dụ cụ thể xác định thuật toán tối ưu

Có 4 em với thời gian vượt cầu lần lượt là 1, 5, 2 và 12 phút. Ta có 2 phương án sau đây:

Phương án 1: Chọn em thứ 1 có thời gian vượt cầu là nhỏ nhất cầm đèn dẫn từng người qua cầu.

- Em thứ 1 dẫn em thứ 3 mất 2 phút.
- Em thứ 1 cầm đèn về mất 1 phút
- Em thứ 1 dẫn em thứ 2 mất 5 phút
- Em thứ 1 cầm đèn về mất 1 phút.
- Em thứ 1 dẫn em thứ 4 mất 12 phút

Tổng thời gian qua cầu của 4 em là 21 phút.

Phương án 2:

- Em thứ 1 dẫn em thứ 3 mất 2 phút.
- Em thứ 1 cầm đèn về mất 1 phút
- Em thứ 2 dẫn em thứ 4 mất 12 phút
- Em thứ 3 cầm đèn về mất 2 phút.
- Em thứ 1 dẫn em thứ 3 mất 2 phút

Tổng thời gian qua cầu của 4 em là 19 phút.

Từ ví dụ trên ta rút ra nhận xét là:

- Một em quá chậm đi cùng em quá nhanh sẽ làm 'hại em đi nhanh, do đó khi qua cầu nên cử hai em chậm nhất.
- Khi cầm đèn chạy về nên giao cho em đi nhanh nhất.

Ý tưởng của giải thuật:

Gọi nơi tập hợp các em nhỏ trước khi qua cầu là T (bên trái cầu), nơi cần đến là P (bên phải cầu).

Trước hết tìm hai em đi nhanh nhất đoàn là a và b.

Lăp các bước 1-6 cho đến khi T rỗng:

1. a và b qua P,

2. a cầm đèn về T,

3. a giao đền cho 2 em chậm nhất x và y ở T,

4. x và y qua P,

5. x và y giao đèn cho b ở P,

6. b trở về T.

7. a và b qua P.

Chú ý: Bạn cần lưu ý trường hợp n=1. Ngoài ra mỗi khi đến P bạn cần kiểm tra ngay xem còn ai ở T không, mỗi khi ở T bạn lại phải kiểm tra xem còn bao nhiêu người chưa qua cầu, có như vậy vòng lặp mới kết thúc đúng lúc.

Chuẩn bị dữ liệu đầu vào.

Dữ liệu vào được lưu vào tập tin Quacau.inp có cấu trúc:

Dòng đầu ghi số người n

n dòng tiếp theo ghi thời gian qua cầu của người thứ i (i=1..n).

Kết quả ra lưu vào tập tin Quacau.out có cấu trúc:

Dòng đầu ghi tổng thời gian qua cầu của n người.

Các dòng tiếp theo ghi cách qua cầu.

Giải thuật chi tiết:

Bước 1: Khởi tạo:

♦ For i:=1 → n

▪ b[i]=false; {người thứ i chưa qua cầu}

Bước 2: Kiểm tra số người qua cầu:

Nếu n=1 thì Người 1 qua, với tổng thời gian là a[1]

Ngược lại:

♦ Tổng:=0; m:=n; {m là số người chưa qua cầu}

♦ min:=true; {hai người có thời gian nhỏ nhất chưa qua cầu}

♦ Trong khi m>0 thì còn làm:

- Nếu min=true Thì

  + Tìm 2 người (q1,q2) có thời gian nhỏ nhất để qua cầu

  Ngược lại:

  + Tìm 2 người (q1,q2) có thời gian lớn nhất để qua cầu

- Nếu a[q1]>a[q2] Thì Tổng:=Tổng+a[q1]

  Ngược lại: Tổng:=Tổng+a[q2];

- m:=m-2; b[q1]:=true; b[q2]:=true;

- Nếu m>0 Thì {vẫn còn người chưa qua cầu}

  + Tìm 1 người (q1) có thời gian nhỏ nhất để quay lại.

  + Tổng:=Tổng+a[q1];

  + m:=m+1; b[q1]:=false;

- min:=not min;

✍ Lưu kết quả trong quá trình thực hiện.

Chương trình

```
Const  max=100;
Var          a:array[1..max] of integer;
             b:array[1..max] of boolean;
             i,j,n,Tong:integer;
             f,f1:text; st:string;
Procedure    Nap;
     Begin
             assign(f,'Quacau.inp');
             reset(f); readln(f,n);
             For i:=1 To n Do
                  Begin
                       readln(f,a[i]); b[i]:=false;
                  End;
             close(f);
             assign(f,'Quacau.ou1');
             assign(f1,'Quacau.out');
             rewrite(f); rewrite(f1);
     End;
Procedure    Tim(min,dk:boolean;var q:integer);
```

```
        Begin
                q:=0;
                For i:=1 To n Do
                        If (b[i]=dk) Then
                                If q=0 Then q:=i
                                Else
                                        If  min xor (a[i]>a[q]) Then q:=i
        End;
Procedure   Quacau;
        Var     m,q1,q2:integer; min:boolean;
        Begin
                Tong:=0;m:=n;min:=true;
                While m>0 Do
                        Begin
                                Tim(min,false,q1); b[q1]:=true;
                                Tim(min,false,q2); b[q2]:=true;
                                If a[q1]>a[q2] Then Tong:=Tong+a[q1]
                                Else Tong:=Tong+a[q2];
                                m:=m-2;
                                writeln(f,q1,' va ',q2,' qua');
                                If m>0 Then
                                        Begin
                                                Tim(true,true,q1); b[q1]:=false;
                                                Tong:=Tong+a[q1]; m:=m+1;
                                                writeln(f,q1,' lai');
                                        End;
                                min:=not min;
                        End;
        End;
Begin
        Nap;
If n=1 Then
                Begin
                Tong:=a[1];
                writeln(f,'1 qua');
                End
        Else Quacau;
        close(f); reset(f);
        writeln(f1,Tong);
        writeln('Tong thoi gian la: ',Tong);
        While not eof(f) Do
                Begin
                        readln(f,st); writeln(f1,st); writeln(st);
                End;
        close(f);close(f1);
        readln;
End.
```

Test chương trình

Cho dữ liệu đầu vào trong tập tin Quacau.inp như hình bên.

Sau khi thực hiện chương trình kết quả cho ra ở tập tin Quacau.out.

Quacau.inp
7 34 6 2 55 4 20 4

Quacau.out
105 3 va 5 qua 3 lai 4 va 1 qua 5 lai 3 va 5 qua 3 lai 6 va 2 qua 5 lai 3 va 5 qua 3 lai 7 va 3 qua

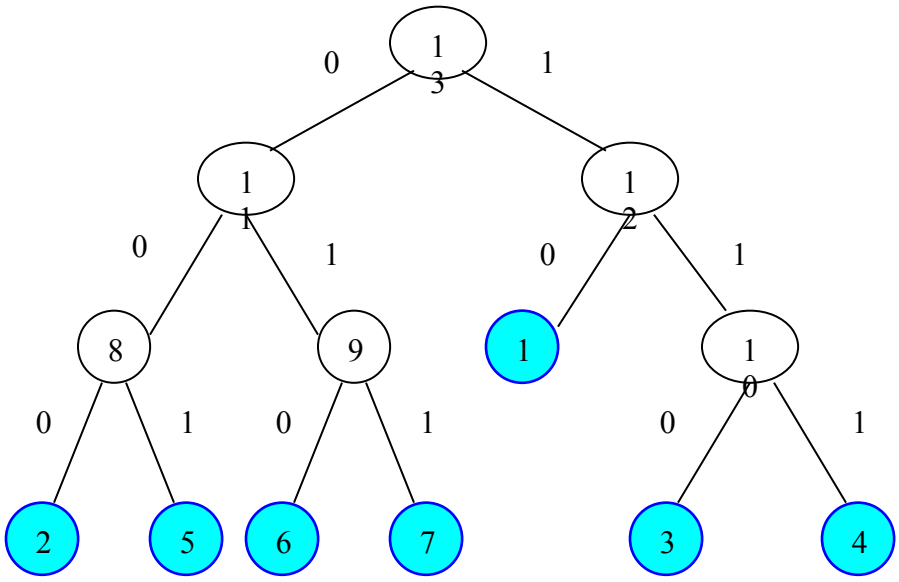
Bài số 2: Bài toán mã Hubbman

Mô tả bài toán:

Cho chuỗi s = "nang ngoc nu"

Ký tự:	n	a	g		o	c	u	
Tần suất:		4	1	2	2	1	1	1
Nút (đỉnh):	1	2	3	4	5	6	7	

Ta xây dựng cây Hubbman như sau:



Ta có mã Hubbman là:

Vị trí (Lá):	1	2	3	4	5	6	7
Ký tự:	n	a	g		o	c	u
Mã:	10	000	110	111	001	010	011

Thuật toán:

- Cho chuỗi s = 'nang ngoc nu'
- Lập bảng tần suất:
- n = số ký tự xuất hiện trong chuỗi (số lá) = 7

Ký tự:	n	a	g		o	c	u
Tần suất:	4	1	2	2	1	1	1
Vị trí:	1	2	3	4	5	6	7

- Xây dựng cây Hubbman:
- For h:= n +1 → n + n - 1 {Tạo thêm h nút mới của cây}
  - min2(i, j); {tìm 2 nút chưa xét (từ 1 đến h) có trọng số nhỏ nhất}
  - d[h]:=d[i]+d[j]; {trọng số của nút mới}
  - R[h]:= i; L[h]: =j;
- h:= n + n + 1; {h là gốc cây}
- Duyệt cây h để tìm mã:

```
Procedure   Duyetcay(h: integer);
        Var t: integer;
        Begin
                t:=h;    x[t]:='';
While   (t >n) do    {trong khi chưa đến lá}
                    begin
if L[t] >0 then  x[L[t]] := x[t] + '0';
                if R[t] >0 then  x[R[t]] := x[t] + '1';
                        t:= t-1;
                    end;
        End;
```

- Tạo mã:
- Sau khi tìm được mã, ta duyệt chuỗi s và thay các ký tự bằng mã tương ứng.
- Giải mã:
  - ss:=''; t:=h; {gốc cây}
  - For i:=1 → length(s1) {s1 là đoạn mã cần giải}
    - if s1[i]='0' then t:= L[t];
      else t:= R[t];
    - if t<= n then
      ss:=ss + kt[t];
      t:= h; {về lại gốc}
  - Giaima:=ss;

Chương trình:

```
Const MN=255
Type    mc1=array[0..MN] of char;
        mi1=array[0.MN] of integer;
        mb1=array90..MN] of byte;
d: mi1; {trong so}
n: int;   {so la}
h: int; {goc dinh cuoi}
L, R: mi1;
kt: mc1; {ki tu cua mang}
daxet: mb1;
s, kq1, kq2: string;
x: array[1..MN] of string;  {ma cua la}
i, j: integer;
vitri: array[1..MN] of integer; {Thu tu cua ki tu  C trong bang tan suat}

Procedure tansuat(s: string);
Var     v: array[#0..#256] of integer;
        C: char;
        Begin
          Fillchar(v,sizeof(v),0);
          For i:=1 to length(s) do
                Int(v[s[i]]);
          n:=0;
  for c:= #0  to #255 do
     if v[c]<>0 then
       begin
         inc(n);
         kt[n]:=c;
         vitri[c]:=n;
         d[n]:=v[c];
        end;
         end;

Function min:integer;
Var     m,k,imin: integer;
        Begin
           M:=maxint;
           For i:=1 to h do
             If daxet[i]=0 then
                  If d[i]<m then
                    Begin
                          imin:=i;
                          m:=d[i];
                      end;
           daxet[imin]:=1;
           min:=imin;
        End;
```

```
Procedure min2(var i,j: integer);
        Begin
          i:=min;
          j:=min;
        end;
Procedure  Hubbman;
        Begin
          Fillchar(daxet, sizeof(daxet),0)
          For h:= n+1 to (n+n-1) do
                Begin
                  min2(i,j);
                  d[h]:=d[i] + d[j];
                  R[h]:=i; L[h]:=j;
                 End;
          h:= n + n –1;
        end;
Procedure Duyetcay(h: integer);
Var     t: integer;
        Begin
           t:=h;
           x[t]:=’’;
           while t > n do
             begin
if L[t] >0 then  x[L[t]] := x[t] + ‘0’;
                 if R[t] >0 then  x[R[t]] := x[t] + ‘1’;
                 t:= t-1;
              end;
        end;
Function Taoma(s: string): String;
Var     ss: string;
        Begin
           ss:=’’
           for i:=1 to length(s) do
                 ss:= ss + x[vitri[s[i]]];
           taoma:=ss;
        end;
Function Giaima(s1: string): string;
Var     ss: string;
        t: integer;
        Begin
           ss:=’’; t:=h;
           for i:= 1 to length(s1) do
             begin
                 if s1[i]=’0’ then t:= L[t];
                 else t:= R[t];
                 if  t<= n then
                   begin
                         ss:=ss + kt[t];
                         t:= h;
```

```
                end;
            end;
        giaima:= ss;
    end;
BEGIN
    s:='nang ngoc nu';
    tansuat(s);
    Hubbman;
    Duyetcay(h);
    Kq1:=taoma(s);
    Kq2:=giaima(kq1);
    Readln;
END.
```

Bài toán 3: Bài toán sắp Topo

.1.15 Mô tả bài toán

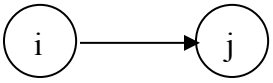
Để hoàn thành một công trình, người ta cần thực hiện nhiều công việc, giả sứ có n công việc. Với mỗi công vệic, ta phải biết công việc đó phải thực hiện sau những công việc nào. Hãy liệt kê một trật tự các công việc để thực hiện theo lịch đó thì sẽ hoàn thành được công trình.

Phân tích bài toán:

Ta có thể lập một đồ thị có hướng G =(V, E) để minh hoa cho các yêu cầu theo công việc.

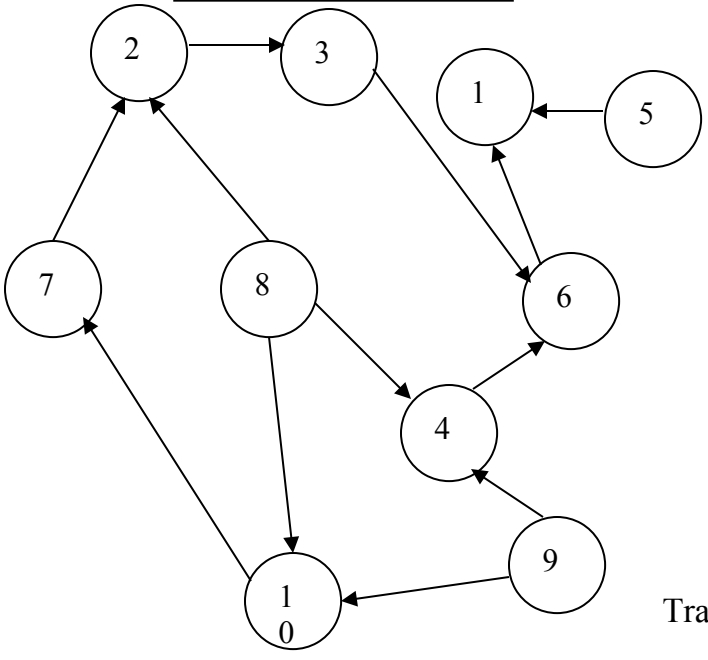
V: Tập các công việc {1, 2,…, n}, n là số công việc cần thực hiện.

E: Tập các cung đi từ việc (đỉnh) i đến j {i,j: 1..n}



: Việc i phải hoàn thành trước j.

Giả sử ta có đồ thi sau:



Biểu diễn ma trận của đồ thị:

i \ j	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0
5	1	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0
7	0	1	0	0	0	0	0	0	0	0
8	0	1	0	1	0	0	0	0	0	1
9	0	0	0	1	0	0	0	0	0	1
10	0	0	0	0	0	0	1	0	0	0

C[i,j] = $\begin{cases} 1; & \text{if } i \to j \\ 0; & \text{else} \end{cases}$

Theo đồ thị trên: Việc 1 thực hiện trước việc 5 và 6; Việc 3 thực hiện trước việc 2, …

Ta có thể mô tả lại đồ thị trên bằng ma trận C[i,j]; i, j: 1..n;

C[i,j] = 1 nếu tồn tại cung đi từ i đến j; C[i,j] = 0 cho các trường hợp còm lại.

Theo ma trận trên , ta tính được bậc vào (BV) cho đỉnh (việc) j: $BV[j] = \sum_{i=1}^{n} C[i, j]$.

Thuật toán:

- Đọc dữ liệu từ tập tin f: 'matran.inp' ma trận c[i,j] như sau:

0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0

- Tính bậc vào BV cho đỉnh j (j=1..n): $BV[j] = \sum_{i=1}^{n} C[i, j]$
- Lặp n lần việc sau:

  Tìm đỉnh trọc j: Đỉnh j là trọc khi:

  - Bậc vào BV[j] = 0

  - Đỉnh j chưa được xếp vào danh sách trình tự công việc cần thực hiện: Daxep[j] = 0;

  - Nếu không tìm được j: Ngừng, Bài toán vô nghiệm (tồn tại chu trình).

  - Nếu tìm được j: Tháo khớp đỉnh j;

  - Xếp đỉnh (việc) j vào trình tự các công việc cần thực hiện.

  - Tháo khớp đỉnh j được thực hiện: For i:= 1 to n do

    Nếu C[j,i] = 1 thì dec(BV[i]);

- Dữ liệu ra tệp g: 'Ketqua.out' như sau:

1 thuc hien viec 5

2 thuc hien viec 8

3 thuc hien viec 9

4 thuc hien viec 4

5 thuc hien viec 10

6 thuc hien viec 7

7 thuc hien viec 2

8 thuc hien viec 3

9 thuc hien viec 6

10 thuc hien viec 1

Chương trình

```
Const n = 10;
Var      c: array[1..n, 1..n] of byte;
BV: array[1..n] of integer;  {bac vao}
Daxep: array[1..n] of byte;
Kq: array[1..n] of byte; {trình tự thực hiện công việc}
m: integer;  {so viec da xep}
          i, j: integer;
          f, g: text;
Procedure Doc;
          Begin
                    Assign(f,'Matran.inp');
                    Reset(f);
                    For i:= 1 to n do
                      Begin
     For j:= 1 to n do              Read(f,C[i,j]);
                        Readln(f);
                       End;
                    Close(f);
          End;
Prcedure Bacvao;
          Begin
                    Fillchar(BV, sizeof(BV),0);
For j:=1 to n do
                       For i:= 1 to n do
                            BV[j]:= BV[j] + c[i,j];
          End;

Function Dinhtroc: integer;  {Tim va danh dau dinh troc}
  Begin
```

```
        For i:=1 to n do
          If (Daxep[i] = 0) and  (BV[i])=0 then
              Begin
                 Dinhtroc:=i; Daxep[i]:=1;
                 Exit;
               End;
        Dinhtroc:=0;
  End;
Procedure Thaokhop(j: integer);
  Begin
        For i:=1 to n do
           If c[j,i] = 1 then  dec(BV[i]);
  End;

Procedure TopoSort;
    Begin
        Assign(g,'Ketqua.out'); Rewrite(f);
Fillchar(daxep, sizeof(daxep),0);
m:=0;
For i:=1 to n do
Begin
        j:=Dinhtroc;
        if j= 0 then    {ton tai chu trinh}
                exit;
        inc(m); kq[m]: =j;   {xep viec j vao trinh tu vi tri m}
        Writeln(g, m,' thuc hien viec', kq[m]);
thaokhop(j);
 end;
  Close(f);
     end;
BEGIN
        Doc;
        Bacvao;
        TopoSort;
        Readln;
END.
```

III MỘT SỐ BÀI TOÁN LUYỆN TẬP

Bài số 1: Phủ đoạn thẳng

Cho một tập N đoạn thẳng (đánh số từ 1 đến N) với các đầu mút có tọa độ nguyên $[L_i, R_i], i = 1, 2, \dots, N$ và một đoạn thẳng $[P, Q]$.

Yêu cầu: Tìm một số ít nhất đoạn thẳng trong tập đã cho phủ kín đoạn $[P, Q]$ (tức là mỗi điểm $x \in [P, Q]$, x phải thuộc vào ít nhất một trong số các đoạn được chọn).

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

- Dòng đầu tiên chứa 3 số nguyên N, P và Q $(N \leq 10^5, 0 < P \leq Q \leq 10^9)$
- N dòng tiếp theo mỗi dòng chứa hai số L_i, R_i $(0 < L_i \leq R_i \leq 10^9)$

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

- Dòng 1: Đưa ra số nguyên K là số đoạn ít nhất chọn được hoặc đưa ra -1 nếu không chọn được các đoạn thỏa mãn.
- Dòng 2: Ghi K số thể hiện số hiệu của các đoạn thẳng được chọn.

Ví dụ:

COVER.INP	COVER.OUT
3 4 6 1 2 3 5 4 7	1 3

Subtasks:

- *Có 60% số test ứng với 60% số điểm của bài có* $N \leq 10^4, 0 < L_i \leq R_i \leq 10^5$
- *Có 40% số test ứng với 40% số điểm của bài có* $N \leq 10^5, 0 < L_i \leq R_i \leq 10^9$

Bài số 2: APALIN

Cho dãy A gồm N số nguyên dương. Dãy *A* được gọi là dãy palindrome nếu như với mọi i từ 1 đến N ta có $A_i = A_{N-i+1}$. Bạn được yêu cầu tìm cách thực hiện ít bước biến đổi nhất để dãy A trở thành dãy palindrome. Với mỗi bước biến đổi, bạn có thể thay thế 2 phần tử liên tiếp bằng tổng của chúng. Như vậy sau mỗi bước, số phần tử của dãy giảm đi 1.

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

- Dòng đầu tiên: chứa số nguyên dương N ($1 \leq N \leq 10^6$).
- Dòng thứ hai: chứa N số nguyên dương biểu diễn dãy A ($A_i \leq 10^9$).

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

- Một số nguyên duy nhất là số bước ít nhất cần thực hiện.

Ví dụ:

APALIN.INP	APALIN.OUT
3 12 3	1

Bài số 3: LAMP

Nhà Hưng xếp 2*N* bóng đèn trang trí trên 2 hàng và *N* cột. Mỗi bóng đèn có thể ở trạng thái tắt hoặc bật. Ban đầu tất cả các bóng đèn đều ở trạng thái tắt, Hưng muốn bật một số bóng đèn lên để tạo thành một mẫu trang trí đẹp.

Hai dãy đèn trang trí này được thiết kế đặc biệt, mỗi bước Hưng có thể đổi trạng thái của một chuỗi liên tiếp nhiều (hoặc một) bóng đèn trên cùng hàng hoặc cùng cột.

Yêu cầu

Cho trước mẫu của trạng thái bật tắt các bóng đèn mà Hưng mong muốn, bạn hãy giúp Hưng tìm cách chuyển trạng thái với ít bước chuyển nhất để có được mẫu mong muốn đó.

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

- Dòng đầu chứa duy nhất một số nguyên dương $N \leq 10000$;
- Mỗi dòng trong hai dòng tiếp theo chứa một chuỗi *N* số nhị phân 0 hoặc 1 mô tả mẫu trang trí Hưng mong muốn. 1 chỉ đèn bật còn 0 chỉ đèn tắt.
- Hai số liên tiếp trên cùng một dòng được ghi cách nhau bởi dấu cách.

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

- Một số nguyên duy nhất là số bước chuyển tối thiểu tìm được.

Ví dụ

LAMP.INP	LAMP.OUT
5 11011 11011	3

Bài số 4: UAV thông minh

Một cuộc đua dành cho các máy bay không người lái thông minh (UAV cỡ nhỏ) được tổ chức trên một đường băng dài gồm N vạch cách đều nhau, vạch cách vạch 10 mét. Các vạch được đánh số từ 1 đến N. Trên mỗi vạch đều có đặt một bộ cảm biến có nhiệm vụ gửi về trung tâm điều khiển (TTĐK) của Ban tổ chức cuộc thi (BTC) số hiệu của vạch khi UAV đứng hay hạ cánh tại vạch này. Cuộc đua cho mỗi UAV được tiến hành như sau: UAV vào đứng tại vạch 1, có thời gian một giây để nạp dữ liệu mà BTC cung cấp. Dữ liệu gồm một số nguyên dương L và N số nguyên X_i (i=1,…,N) với ý nghĩa: X_i là giá trị của vạch i. Ngay sau đó, UAV phải thực hiện hành trình bằng cách liên tục di chuyển như sau:

- Trong hành trình lượt đi, UAV (đứng tại vạch 1) cần bay đến được vạch N theo quy tắc: nếu đang đứng tại vạch i ($1 \leq i < N$) thì nó sẽ chỉ được phép hạ cánh tại vạch j (j>i) mà X_j lẻ (ấn định rằng X_N=1001) đồng thời vạch j cách vạch i không quá L vạch (tức là, $1 \leq j-i \leq L$). Tổng số lần UAV đứng hay hạ cánh trong hành trình lượt đi, bao gồm cả tại vạch 1 và vạch N, được ký hiệu bởi U.
- Trong hành trình lượt về, bắt đầu với số điểm được BTC cung cấp bằng X_N=1001, UAV từ vạch N bay tiếp về vạch 1 theo quy tắc: Nếu đang đứng ở vạch i ($1 < i \leq N$) thì nó sẽ chỉ được phép hạ cánh tại vạch k (k<i) mà X_k chẵn (ấn định rằng X_1=1000). UAV sẽ nhận được thêm X_k điểm nếu vạch k cách vạch i đúng L vạch (tức là, i-k=L) và bị trừ X_k điểm trong trường hợp trái lại. Tổng số điểm mà UAV thu được trong hành trình lượt về, được ký hiệu bởi V.

Lưu ý:

- Các UAV được lập trình sẵn để tiếp nhận và xử lý dữ liệu của BTC rồi tự động thực hiện toàn bộ hành trình (lượt đi và về).
- Dữ liệu mà BTC đưa ra luôn đảm bảo để các UAV có thể thực hiện được cuộc đua.

Yêu cầu: Hãy lập trình cho UAV để nó đạt được giá trị nhỏ nhất của U và lớn nhất của V.

Dữ liệu: Vào từ file văn bản **UAV.INP** có cấu trúc:

- Dòng đầu ghi số nguyên L ($10 \le L \le 100$);
- Dòng thứ hai ghi số nguyên N ($L < N \le 500000$) ;
- Dòng thứ ba ghi lần lượt các số nguyên $X_2,\ldots,X_{N-1}$ ($0 \le X_i \le 106$, i=2,…,N-1). Các số cách nhau bởi dấu cách.

Kết quả: Ghi ra file văn bản **UAV.OUT** 2 số nguyên trên 2 dòng: Dòng đầu là số U, dòng sau là số V.

Ví dụ:

UAV.INP	**UAV.OUT**
10 19 6 3 15 2 30 7 8 6 10 2 4 9 9 15 0 18 10	4 1999

Ràng buộc: 50% số test ứng với 50% số điểm của bài có $N \le 1000$.

KẾT LUẬN

Chuyên đề này là cơ hội để tìm hiểu kỹ hơn về giải thuật nâng cao nói chung và đi sâu nghiên cứu về giải thuật tham lam, một phương pháp để giải lớp các bài tóan tối ưu đạt hiệu quả cao.

Đã đạt được những kết quả sau:

- Nâng cao hiểu biết về giải thuật
- Tìm hiểu và trình bày rõ ràng về giải thuật tham lam
- Áp dụng giải thuật tham lam để giải một số bào tóan tối ưu như bài tóan cái ba lô, bài tóan băng nhạc, bài tóan Sắp Topo, bài tóan chọn việc, bài tóan cây bao trùm tối thiểu, …

Có thể nói giải thuật tham lam rất có ích trong việc giải lớp các bài tóan tối ưu, tuy nhiên giải thật này cũng có một số nhược điểm sau:

- Thuật toán tham lam không phải lúc nào cũng cho lời giải tối ưu.
- Khi sử dụng thuật toán ta cần chứng minh thuật toán cho lời giải tối ưu.
- Có một số bài toán mà kỹ thuật tham lam có thể không cho một lời giải tối ưu ví dụ: bài toán phủ đỉnh

TÀI LIỆU THAM KHẢO

1. Tìm đường trong mê cung – TSKH Nguyễn Xuân Xuân Huy, NXB Giáo dục, 1996.
2. Cấu trúc dữ liệu và thuật tóan – Đinh Mạnh Tường, NXB Khoa học và kỹ thuật 2001.
3. Cơ sở toán trong lập trình – Đỗ Đức Giáo, NXB khoa học và kỹ thuật, 1998.
4. Bài Tập Cấu trúc dữ liệu và giải thuật- Lê Minh Trung, NXB Thống kê 2000
5. Nguyễn Đình Thúc (2001), *Lập trình tiến hoá*, NXB Giáo dục
6. Đặng Văn Chuyết và Nguyễn Tuấn Anh (2000), *Cơ sở lý thuyết truyền tin,* NXB giáo dục