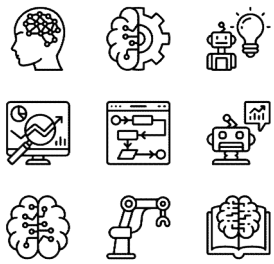
Computer Science for Practicing Engineers

Phương pháp quy hoạch động



TS. Huỳnh Bá Diệu
Email: dieuhb@gmail.com
Phone: 0914146868

Phương pháp qui hoạch động (dynamic programming)

Nội dung:

1. Quy hoạch động là gì?
2. Quy hoạch động vs chia để trị
3. Các bước trong giải bài toán bằng quy hoạch động
4. Ví dụ các bài toán giải bằng quy hoạch động

Phương pháp qui hoạch động (dynamic programming)

Giải bài toán bằng cách chia bài toán lớn thành các bài toán nhỏ

Giải các bài toán nhỏ và ghi nhớ kết quả.

Khi gặp một bài toán nhỏ đã giải thì dùng lại kết quả, không giải lại từ đầu.

Qui hoạch động thường được dùng khi có các bài toán con chồng nhau.

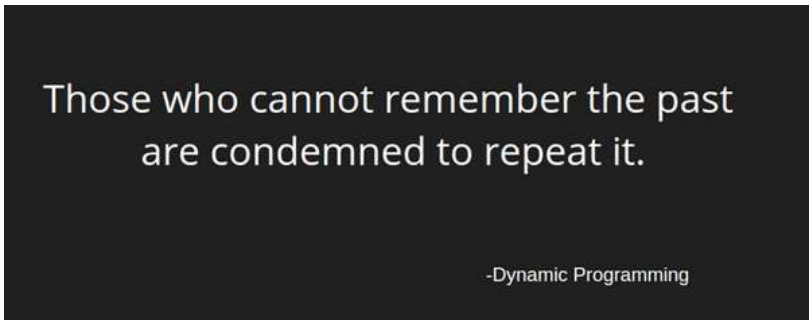
Phương pháp qui hoạch động (dynamic programming)

Jonathan Paulson explains Dynamic Programming in Quora:

Writes down "1+1+1+1+1+1+1+1 =" on a sheet of paper.
"What's that equal to?"
Counting "Eight!"
Writes down another "1+" on the left.
"What about that?"
"Nine!" " How'd you know it was nine so fast?"
"You just added one more!"

Phương pháp qui hoạch động (dynamic programming)

The core idea of Dynamic Programming is to avoid repeated work by remembering partial results and this concept finds it application in a lot of real life situations.



Phương pháp qui hoạch động (dynamic programming)

Some famous Dynamic Programming algorithms are:

- Unix diff for comparing two files
- Bellman-Ford for shortest path routing in networks
- TeX the ancestor of LaTeX
- WASP - Winning and Score Predictor

Phương pháp qui hoạch động (dynamic programming)

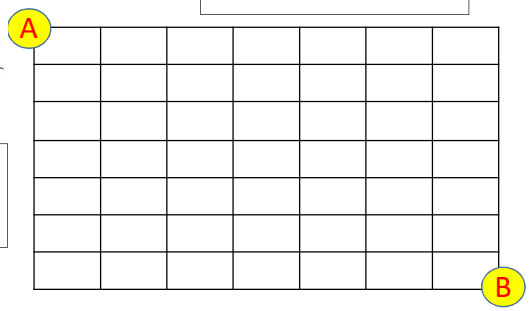
Các bước trong giải bài toán bằng quy hoạch động (4 bước)

- Characterize the structure of an optimal solution.
- Recursively define the value of an optimal solution.
- Compute the value of an optimal solution, typically in a bottom-up fashion.
- Construct an optimal solution from the computed information.

Phương pháp qui hoạch động: Bài toán con kiến

Bài toán **con kiến**

n cột: 0-→n

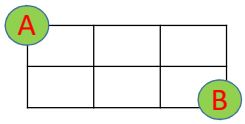


m dòng
0→m

Qui luật khi di chuyển từ A→ B: chỉ đi qua phải hoặc đi xuống

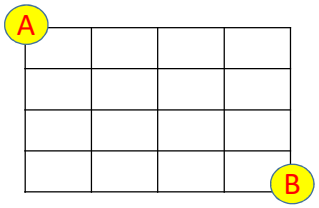
Phương pháp qui hoạch động: Bài toán con kiến

Bài toán **con kiến**



Qui luật khi di chuyển từ A→ B: chỉ đi qua phải hoặc đi xuống

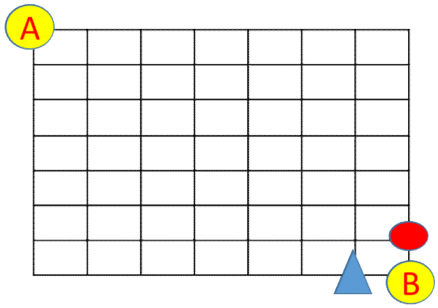
Có bao nhiêu cách cho mỗi trường hợp?



Phương pháp qui hoạch động: Bài toán con kiến

Nhận xét:

Nếu m=0 hoặc n=0 thì chỉ có 1 cách là đi thẳng hoặc đi xuống ➔sc=1



$$sc(m,n) = \begin{cases} 1 \text{ nếu } n=0 \text{ hoặc } m=0 \\ sc(m,n-1) + sc(m-1,n) \end{cases}$$

Nếu m> 0 và n> 0 thì sc(m,n) = sc(m,n-1) + sc(m-1,n)

Phương pháp qui hoạch động: Bài toán con kiến

Bài toán **con kiến**

```
long long ck(int m,int n) {
    if(m==0 && n==0 ) return 0;
    else
      if(m==0|| n==0) return 1;
      else return ck(m, n-1) + ck(m-1, n);
}
```

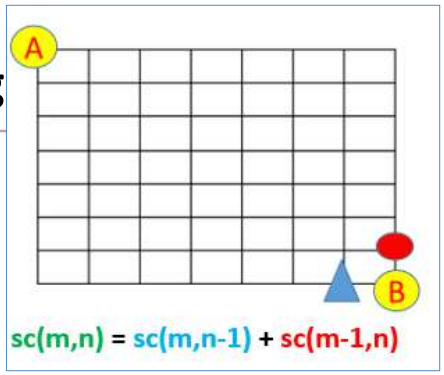
Phương pháp qui hoạch động: Bài toán con kiến

DP_ConKien.cpp

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  long long ck(int m,int n)
4  {
5      if(m==0 && n==0 ) return 0;
6      else
7          if(m==0|| n==0) return 1;
8          else return ck(m, n-1) + ck(m-1, n);
9  }
10
11
22 int main()
23 {
24     int m=30, n=40, t1,t2;
25     t1=clock();  long long kq=ck(m,n);      t2=clock();
26     cout<<"\n So cach di tren luoi "<<m<< ", n"<<n<<" la: "<<kq;
27     cout<<"\n thoi gian tinh toan la = "<<(t2-t1);
28 }
```

Hãy chạy thử m=30, n=40. Cho biết thời gian thực hiện trên máy tính???

Phương pháp qui hoạch động



Bài toán **con kiến**

C(6,7) = C(6,6) + C(5,7)

C(6,6)= C(6,5) + **C(5,6)**

C(5,7) = **C(5,6)** + C(4,7)

Giá trị **C(5,6)** phải tính 2 lần khi tính C(6,7)

Giải pháp: lập bảng tính, tính toán và lưu kết quả (để tránh tính lại nhiều lần)

Nếu tính rồi thì có thể dùng lại kết quả.

Dùng bảng có kích thước **m + 1** hàng, **n+1** cột

Dòng 0: cho các giá trị =1, Cột 0: cho các giá trị =1, Ô(0,0): =0

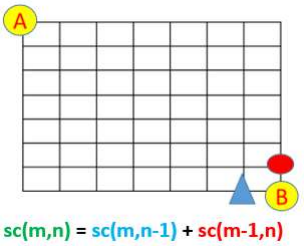
Các ô khác tính theo công thức, lặp từ hàng 1 đến hàng m, cột 1 đến n.

Phương pháp qui hoạch động: Bài toán con kiến

Bài toán **con kiến** C(6,7)

Thử điền kết quả vào bảng???

	0	1	2	3	4	5	6	7
0	0	1	1	1	1	1	1	1
1	1							
2	1							
3	1							
4	1							
5	1							
6	1							



Phương pháp qui hoạch động: Bài toán con kiến

Bài toán **con kiến** C(6,7)

	0	1	2	3	4	5	6	7
0	0	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8
2	1	3	6	10	15	21	28	36
3	1	4	10	20	35	56	84	120
4	1	5	15	35	70	126	210	330
5	1	6	21	56	126	252	462	792
6	1	7	28	84	210	462	924	1716

Phương pháp qui hoạch động: Bài toán con kiến

Bài toán **con kiến** C(6,7)

	0	1	2	3	4	5	6	7
0	0	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8
2	1	3	6	10	15	21	28	36
3	1	4	10	20	35	56	84	120
4	1	5	15	35	70	126	210	330
5	1	6	21	56	126	252	462	792
6	1	7	28	84	210	462	924	1716

Phương pháp qui hoạch động: Bài toán con kiến

```
long long ck_dp(int m, int n)   {
    long long **a;  a=new long long*[m+1];
    for(int i=0; i<=m; i++) a[i]= new long long [n+1];
    for(int i=1; i<=n; i++) a[0][i] =1;
    for(int i=1; i<=m; i++) a[i][0]=1;
    for(int i=1; i<=m; i++)
        for(int j=1; j<=n; j++)
            a[i][j] =a[i][j-1]+ a[i-1][j];
    return a[m][n];
  }
```

Phương pháp qui hoạch động: Bài toán con kiến

Nhận xét về cách tính toán khi tính A[i][j]

	0	1	2	3	4	5	6	7
0	0	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8
2	1	3	6	10	15	21	28	36
3	1							
4	1							
5	1							
6	1							

Phương pháp qui hoạch động: Bài toán con kiến

Chuyển từ mảng 2 chiều về mảng 1 chiều

	0	1	2	3	4	5	6	7
0	0	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8
2	1	3	6	10	15	21	28	36
3	1							
4	1							
5	1							
6	1							

Phương pháp qui hoạch động: Bài toán con kiến

Chuyển từ mảng 2 chiều về mảng 1 chiều

B1: Cấp phát bộ nhớ cho mảng a gồm n+ 1 phần tử

B2: Khởi gán a[i]= 1; *với i=0 đến n*

B3: Cho i chạy từ 1 đến m

cho j chạy từ 1 đến n

a[j]= a[j-1] + a[j];

B4: Return a[n];

Phương pháp qui hoạch động: Bài toán con kiến

```
long long ck_dp1(int m, int n){
      long long *a;
      a= new long long [n+1];
      for(int i=0; i<=n; i++) a[i] =1;
      for(int i=1; i<=m; i++)
          for(int j=1; j<=n; j++)
              a[j] =a[j-1]+ a[j];
      return a[n];
}
```

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Cho hai xâu X, Y có độ dài là m,n. **longest common subsequence**

Tìm độ dài chuỗi con chung dài nhất.

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

longest common subsequence

Cho hai xâu X, Y có độ dài là m,n.

Tìm độ dài chuỗi con chung dài nhất.

Ví dụ:

X= ABC

Y = MAGXCM

Chuỗi con chung dài nhất có độ dài 2 là AC.

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Gọi LCS(m, n) là độ dài chuỗi con chung dài nhất của hai chuỗi X và Y, với m là độ dài chuỗi X và n là độ dài chuỗi Y

Nhận xét:

Nếu m hoặc n=0 thì độ dài =0: LCS(m,0) = LCS(0,n) =0

Nếu ký tự cuối cùng của X và ký tự cuối cùng của Y giống nhau:

LCS(m,n)= 1+ LCS(m-1, n-1)

[******A] [******A]**

Nếu ký tự cuối hai chuỗi không giống nhau:

LCS(m,n)= max(LCS(m-1, n) , LCS(m, n-1))

[******A] [******B]**

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Cho hai xâu X, Y có độ dài là m,n. Tìm độ dài chuỗi con chung dài nhất của hai xâu X và Y.

YÊU cầu:

Viết chương trình nhập 2 chuỗi X, Y và in ra độ dài chuỗi con chung dài nhất

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

```
string X, Y;
int LCS(int m, int n){
   if(m==0||n==0) return 0;
   else if(X[m-1] == Y[n-1]) return 1+ LCS( m-1, n-1);
        else return max(LCS( m-1, n),LCS( m, n-1)) ;
}
```

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

```
#include<bits/stdc++.h>
using namespace std;
string X, Y;
int main() {
        X="DBAXDA"; Y= "ABADCA";
        int m=X.length();  int n=Y.length();  long t1, t2, kq;
        t1=clock();  kq=LCS(m,n);      t2=clock();
        cout<<"\n Do dai chuoi con chung dai nhat la :"<<kq;
        cout<<"\n thoi gian tinh toan la = "<<(t2-t1);
}
```

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Tính thời gian chạy trên máy tính cho hai trường hợp sau????

1. / X= “SDKFNSCABADCA"; Y="DBSFOSAXDA";

2. / X= “SDKFNSLDFSDJFLKSDFLSFLSOOIREIWEFOJCASNCMCABADCA";
Y="DBSKJFWEJFOSDJFDOIJFMXCSJFSOJDFSJDOFJSODJFOSLKSJDFOSAXDA";

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Khử đệ qui bằng cách lập bảng.

Tạo bảng m+1 hàng và n+1 cột

Thảo luận nhóm

Tương tự ví dụ 1, các bạn về thảo luận cách thức để giảm thời gian thực hiện thuật toán!!!

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Y= "ABADCA"; X="DBANDA";

	Y	A	B	A	D	C	A
X							
D							
B							
A							
N							
D							
A							

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Y= "ABADCA"; X="DBANDA";

	Y	A	B	A	D	C	A
X	0	0	0	0	0	0	0
D	0						
B	0						
A	0						
N	0						
D	0						
A	0						

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Y= "ABADCA"; X="DBANDA";

	Y	A	B	A	D	C	A
X	0	0	0	0	0	0	0
D	0						
B	0						
A	0						
N	0						
D	0						
A	0						

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Y= "ABADCA"; X="DBANDA";

	Y	A	B	A	D	C	A
X	0	0	0	0	0	0	0
D	0	0	0	0	1+0	1	1
B	0	0	1+0	1	1	1	1
A	0	1+ 0	1	1 + 1	2	2	1 + 1
N	0	1	1	2	2	2	2
D	0	1	1	2	1 +2	3	3
A	0	1 +0	1	1+1	3	3	1+ 3

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

	Y	P	H	Q	K	T	A	K
X								
P								
K								
H								
Q								
A								
P								
K								

Thử cho hai chuỗi sau:

Y= “PHQKTAK"

X=“PKHQAPK";

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

```
int LCS_DP(int m, int n){
      int **a;  a=new int*[m+1];
      for(int i=0; i<=m; i++) a[i]= new int [n+1];
      for(int i=0; i<=m; i++)
          for(int j=0; j<=n; j++) a[i][j] =0;
      for(int i=1; i<=m; i++)
          for(int j=1; j<=n; j++)
              if(X[i-1] == Y[j-1]) a[i][j]=1 + a[i-1][j-1];
              else a[i][j]= max(a[i][j-1], a[i-1][j]);
      return a[m][n];
}
```

Phương pháp qui hoạch động: **Chia bi vào hộp**

Chia bi vào hộp

Cho m viên bi và n cái hộp (được đánh số từ 1 đến n). Cần ***bỏ hết*** m viên bi vào n hộp, sao cho số viên bi trong hộp thứ i ***không ít hơn*** số bi trong hộp thứ i+1. Yêu cầu **đếm có bao nhiêu cách**.

Ví dụ: m=4, n =5

4	0	0	0	0
3	1	0	0	0
2	2	0	0	0
2	1	1	0	0
1	1	1	1	0

Phương pháp qui hoạch động: **Chia bi vào hộp**

Nhận xét:

+ Nếu số hộp nhiều hơn số bi (n>m) thì các hộp ở vị trí m +1 trở đi sẽ không có viên bi nào, vì phải bỏ hết vào m hộp đầu tiên.

+ Nếu m>0 mà không có hộp nào thì số cách là bằng 0

+ Nếu n>0 mà m bằng 0 thì có 1 cách (không có viên bi nào trong hộp)

4	0	0	0	0
3	1	0	0	0
2	2	0	0	0
2	1	1	0	0
1	1	1	1	0

Phương pháp qui hoạch động: **Chia bi vào hộp**

Gọi C(m,n) là số cách bỏ m viên bi vào n hộp

$$C(m,n) = \begin{cases} 0 \text{ nếu } m > 0, n = 0 \\ 1 \text{ nếu } m = 0 \\ C(m,m) \text{ nếu } m < n \\ \boldsymbol{C(m,n-1) + C(m-n,n)} \end{cases}$$

Bỏ cái hộp, thì hộp mới cũng

Xây dựng hàm tính C(m,n)

long tinh (int m, int n) { }

4	0	0	0	0
3	1	0	0	0
2	2	0	0	0
2	1	1	0	0
1	1	1	1	0

bỏ mỗi hộp 1 viên, lấy số bi còn lại bắt đầu chia lại

Phương pháp qui hoạch động: **Chia bi vào hộp**

Chia bi vào hộp

Gọi C(m,n) là số cách bỏ m viên bi vào n hộp

$$C(m,n) = \begin{cases} 0 \text{ nếu } m > 0, n = 0 \\ 1 \text{ nếu } m = 0 \\ C(m,m) \text{ nếu } m < n \\ \boldsymbol{C(m,n-1) + C(m-n,n)} \end{cases}$$

4	0	0	0	0
3	1	0	0	0
2	2	0	0	0
2	1	1	0	0
1	1	1	1	0

TEST với các bộ dữ liệu:

m=10, n=7 ; m=40, n=30; m=160, n=130;

Phương pháp qui hoạch động: **Chia bi vào hộp**

```
long long chia_bi(int m, int n)
{
        if(m>0 && n==0) return 0;
        else if(m==0) return 1;
        else if (m<n) return chia_bi(m,m);
        else return chia_bi(m,n-1) + chia_bi(m-n,n);
}
```

Phương pháp qui hoạch động: **Chia bi vào hộp**

```
int main()
{
        int m=12, n=10, t1,t2;   // m=60, n=46;
        t1=clock();  long long kq=chia_bi(m,n); t2=clock();
        cout<<"\n So cach chia "<<m<<" vien bi vao "<<n<<" hop la: "<<kq;
        cout<<"\n thoi gian tinh toan la = "<<(t2-t1);
}
```

Phương pháp qui hoạch động: **Chia bi vào hộp**

m/n	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							
7							
8							

c(1,1)= c(1,0) + c(0,1)
c(2,1)= c(2,0) + c(1,1)
c(2,2)= c(2,1) + c(0,2)
C(3,1)= c(3,0)+ c(2,1)
c(3,2)= c(3,1) + c(1,2)
c(3,3)= c(3,2) + C(0,3)
c(4,1)=c(4,0) + c(3,1)
c(4,2)= c(4,1) + c(2,2)
c(4,3)= C(4,2) + C(1,3)
c(4,4)= = C(4,3)= C(0, 4)

$$C(m,n) = \begin{cases} 0 \text{ nếu } m > 0, n = 0 \\ 1 \text{ nếu } m = 0 \\ C(m,m) \text{ nếu } m < n \\ C(m,n-1) + C(m-n,n) \end{cases}$$

Phương pháp qui hoạch động: **Chia bi vào hộp**

m/n	0	1	2	3	4	5	6
0	0	1	1	1	1	1	1
1	0						
2	0						
3	0						
4	0						
5	0						
6	0						
7	0						
8	0						

$$C(m,n) = \begin{cases} 0 \text{ nếu } m > 0, n = 0 \\ 1 \text{ nếu } m = 0 \\ C(m,m) \text{ nếu } m < n \\ C(m,n-1) + C(m-n,n) \end{cases}$$

Phương pháp qui hoạch động: **Chia bi vào hộp**

m/n	0	1	2	3	4	5	6
0	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1
2	0	1	2	2	2	2	2
3	0	0+1	1+1	2+1	3	3	3
4	0	0+1	1+2	3+1	4+1	5	5
5	0						
6	0						
7	0						
8	0						

Dựa vào công thức, hãy điền cho bảng ???

C(4,2) = C(4,3) + C(2,2)

$$C(m,n) = \begin{cases} 0 \text{ nếu } m > 0, n = 0 \\ 1 \text{ nếu } m = 0 \\ C(m,m) \text{ nếu } m < n \\ C(m,n-1) + C(m-n,n) \end{cases}$$

Phương pháp qui hoạch động: **Chia bi vào hộp**

Thuật toán khử đệ qui

Cấp phát mảng gồm m + 1 hàng , n+1 cột

Cho giá trị cột 0 =0;

Cho giá trị hàng 0 =1

Cho i chạy từ 1 đến m

cho j chạy từ 1 đến n

nếu j> i thì a[i][j] =a[i][i]

ngược lại a[i][j] = a[i][j-1] + a[i-j][j]

Trả về a[m][n]

Phương pháp qui hoạch động: **Chia bi vào hộp**

```
11  long long chia_bi_dp(int m, int n)
12  {
13          long long **a;
14          a=new long long*[m+1];
15          for(int i=0; i<=m; i++) a[i]= new long long [n+1];
16          for(int j=0; j<=n; j++) a[0][j] =1;
17          for(int i=1; i<=m; i++) a[i][0]=0;
18          for(int i=1; i<=m; i++)
19               for(int j=1; j<=n; j++)
20                  if(j>i) a[i][j] =a[i][i];
21                  else a[i][j] = a[i][j-1] + a[i-j][j];
22          return a[m][n];
23  }
```

Phương pháp qui hoạch động Chuỗi con đối xứng dài nhất

Chuỗi con đối xứng dài nhất

Cho chuỗi S chứa các ký tự.

Tìm độ dài chuỗi con đối xứng dài nhất

S= “AXBBA” d=4

S= AMBAB d=3

S= “ABCDACSBSC” d=

Phương pháp qui hoạch động Chuỗi con đối xứng dài nhất

Chuỗi con đối xứng dài nhất

Cho chuỗi S chứa các ký tự. Tìm độ dài chuỗi con đối xứng dài nhất.

S= “AXBBA” d=4

S= AMBAB d=3

S= “ABCDACSBSC” d=

S= “SKHSDKSHDKSHDKHUEWYWWHDINIIWDIWDHD”

Phương pháp qui hoạch động **Chuỗi con đối xứng dài nhất**

Gọi L(i, j) là độ chuỗi con đối xứng dài nhất từ vị trí i đến vị trí j

Ta có L(i,j) = 0 nếu i>j

L(i,j) = 1 nếu i=j

L(i,j) = 2 + L(i+1, j-1) nếu S[i] = S[j]

L(i,j) = max(L(i+1, j) , L(i, j-1)) nếu S[i] <> S[j]

Phương pháp qui hoạch động **Chuỗi con đối xứng dài nhất**

```
long dx(int i, int j){
        if(i>j) return 0;
        else if(i==j) return 1;
        else if (S[i]==S[j]) return 2 + dx(i+1,j-1);
        else return max(dx(i+1,j), dx(i,j-1));
}
```

Dựa vào công thức đệ qui, hãy hoàn thiện chương trình và cho biết thời gian chạy khi tính trên chuỗi S???

string S= “SKHSDKSHDKSHDKHUEWYWWHDINIIWDIWDHD”;

Phương pháp qui hoạch động **Chuỗi con đối xứng dài nhất**

[*] DP_Chuoi_con_doi_xung_dai_nhat.cpp

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  string S;
4  long dx(int i, int j)
5  {
6      if(i>j) return 0;
7      else if(i==j) return 1;
8      else if (S[i]==S[j]) return 2 + dx(i+1,j-1);
9      else return max(dx(i+1,j), dx(i,j-1));
10 }
11 int main()
12 {
13     S= "ABCDACSBSC";
14     int      n= S.length();
15     long t1,t2, kq;
16     t1=clock();  kq=dx(0,n-1);  t2=clock();
17     cout<<"\n Do dai chuoi con doi xung dai nhat cua S la: "<<kq;
18     cout<<"\n thoi gian tinh toan la = "<<(t2-t1);
19 }
```

S= “SKHSDKSHDKSHDKHUEWYWWHDINIIWDIWDHD”

Phương pháp qui hoạch động **Chuỗi con đối xứng dài nhất**

S= "ABCDACSBSC";

Thảo luận nhóm về cách khử đệ quy dựa vào bảng!!!

i

j

	A	B	C	D	A	C	S	B	S	C
A	1									
B		1								
C			1							
D				1						
A					1					
C						1				
S							1			
B								1		
S									1	
C										1

Phương pháp qui hoạch

$L(i,j) = 2 + L(i+1, j-1)$ nếu $S[i] = S[j]$

$L(i,j) = \max(L(i+1, j), L(i, j-1))$ nếu $S[i] <> S[j]$

Lập bảng để tính cho **S= "ABCDACSBSC";**

i

	A	B	C	D	A	C	S	B	S	C
A	1									
B	0	1								
C	0	0	1							
D	0	0	0	1						
A	0	0	0	0	1					
C	0	0	0	0	0	1				
S	0	0	0	0	0	0	1			
B	0	0	0	0	0	0	0	1		
S	0	0	0	0	0	0	0	0	1	
C	0	0	0	0	0	0	0	0	0	1

j

Chuỗi con đối xứng dài nhất : S= "ABCDACSBSC";

i

	A	B	C	D	A	C	S	B	S	C
A	1	1	1	1	2+1	3	3	5	5	5
B	0	1	1	1	1	3	3	2+3	5	5
C	0	0	1	1	1	2+1	3	3	3	2+3
D	0	0	0	1	1	1	1	1	3	5
A	0	0	0	0	1	1	1	1	3	5
C	0	0	0	0	0	1	1	1	3	2+3
S	0	0	0	0	0	0	1	1	2+1	3
B	0	0	0	0	0	0	0	1	1	1
S	0	0	0	0	0	0	0	0	1	1
C	0	0	0	0	0	0	0	0	0	1

j

Phương pháp qui hoạch động **Chuỗi con đối xứng dài nhất**

Thuật toán

1. **n= độ dài chuỗi S**
2. **Khởi tạo bảng A có kích thước n*n**
3. **Cho các phần tử trên đường chéo chính =1**
4. **Cho i từ n-1 đến 1**

   Cho j chạy từ i+1 đến n-1

   nếu S[i-1]==S[j-1] thì A[i][j] = 2+ A[i+1][j-1]

   ngược lại A[i][j] = max(A[i+1][j] , A[i][j-1])
5. **return A[1] [n]**

Phương pháp qui hoạch động **Chuỗi con đối xứng dài nhất**

```
int dx_dp(int n){
      int **a;  a=new int*[n+1];
      for(int i=0; i<=n; i++) a[i]= new int [n+1];
      for(int i=1; i<=n; i++) a[i][i] =1;
      for(int i=n-1; i>0; i--)
          for(int j=i+1; j<=n; j++)
              if (S[i-1]==S[j-1]) a[i][j] = 2 +a[i+1][j-1];
              else a[i][j] = max(a[i+1][j], a[i][j-1]);
      return a[1][n];
}
```

Phương pháp qui hoạch động: Nhân dãy ma trận

Nhân 2 ma trận

Cho ma trận A gồm 2 hàng 3 cột, ma trận B gồm 3 hàng 4 cột

C= A*B thì C có 2 hàng 4 cột

Số phép nhân cần thực hiện là 2*3*4= 24

Tổng quát: A(m,n) , B(n,k) thì C là C(m,k) (số phép nhân là m*n*k)

```
for(int i=0; i<m; i++)
for(int j =0; j<k; j++)
{
      int t=0; for(int p=0; p<n; p++) t=t+ A[i][p]* B[p][j];
      C[i][j]=t;
}
```

Phương pháp qui hoạch động: Nhân dãy ma trận

Cho n ma trận có kích thước r_i c_i trong dãy phép nhân các ma trận

M= $A_1 * A_2 * A_3 * A_4 * A_5 * **** A_{n-2} * A_{n-1} * A_n$

Hãy đưa ra thứ tự nhân các ma trận sao cho số phép toán thực hiện là ít nhất.

Ví dụ M= $A_1 * A_2 * A_3 * A_4$, Các thứ tự nhân có thể có là:

$((A_1 * A_2) * A_3) * A_4$

$(A_1 * A_2) * (A_3 * A_4)$

$(A_1 * (A_2 * A_3)) * A_4$

$A_1 * (A_2 * (A_3 * A_4))$

$A_1 * ((A_2 * A_3) * A_4)$

Phương pháp qui hoạch động: Nhân dãy ma trận

Ví dụ M= $A_1 * A_2 * A_3 * A_4$

Các thứ tự nhân có thể có là:

A1	A2	A3	A4
3 *5	5*7	7*2	2*4

	Nhân ngoặc 1	Nhân ngoặc 2	Nhân cuối	Số phép nhân
$((A_1 * A_2) * A_3) * A_4$	**105**	**42**	**24**	**171**
$(A_1 * A_2) * (A_3 * A_4)$	**105**	**56**	**84**	**245**
$(A_1 * (A_2 * A_3)) * A_4$	**70**	**30**	**24**	**124**
$A_1 * (A_2 * (A_3 * A_4))$	**56**	**140**	**60**	**256**
$A_1 * ((A_2 * A_3) * A_4)$	**?**	**>**		**?**

Phương pháp qui hoạch động: nhân dãy ma trận

Dữ liệu vào : MMATRIX.inp Dữ liệu ra MMATRIX.out

n
d_0
d_1
d_2
d_3
...
d_n

Số phép nhân phải thực hiện ít nhất

4
3
5
7
2
4

124

Phương pháp qui hoạch động: Nhân dãy ma trận

Cho A_i có kích thước $d_{i-1}d_i$

Gọi F(i, j) là số các phép toán ít nhất khi nhân các ma trận từ ma trận i đến ma trận j.

Ta có:

F(i, i) = 0

F(i, i+1) = $d_{i-1} * d_i * d_{i+1}$

F(i, j) = min (F(i, k) + F(k+1, j) + $d_{i-1} * d_k * d_j$) với k từ i +1 đến j-1

i k k+ 1 j

Phương pháp qui hoạch động: Nhân dãy ma trận

1. Khởi tạo bảng hai chiều; tại F[i][i] : for (i=1 ; i<= n; i++) F[i,i]=0;
2. Điền bảng

```
for(int L=2; L<=n; L++)
   for(i=1;i<=n-L+1;i++)    {
        j=i+L-1;   F[i][j]= MAXINT;
        for(int k=i; k<j; k++)       {
         int q= F[i][k]+F[k+1][j]+d[i-1]*d[k]*d[j];
         if(q<F[i][j]) { F[i][j]=q; S[i][j]=k;}
        }
   }
```

3. return F[1][n];

Phương pháp qui hoạch động: Nhân dãy ma trận

```
int tinh(int n) {
  int i,j,k;
  for (i=1 ; i<= n; i++) F[i][i]=0;
  for(int L=2; L<=n; L++)
    for(i=1;i<=n-L+1;i++)    {
        j=i+L-1; F[i][j]= MAXINT;
        for(int k=i; k<j; k++) {
         int q= F[i][k]+F[k+1][j]+d[i-1]*d[k]*d[j];  if(q<F[i][j]) { F[i][j]=q; S[i][j]=k;}
        }
     }
       return F[1][n];
}
```

Phương pháp qui hoạch động: nhân dãy ma trận

```
void inkq(int i,int j){
  if(i==j)  printf("A%d",num++);
  else   {
    printf("(");
    inkq(i,S[i][j]);   printf(" x ");
    inkq(S[i][j]+1,j); printf(")");
  }
}
#define MAXINT 2000000;
int d[]= {3,5,7,2,4};
int n=5;
```

```
int F[100][100];
int S[100][100];
int num=1;
int main(){
  cout<<"\n -So phep tinh it nhat la:";
  cout <<tinh(n)<<"\n";
  cout<<"\n Thu tu nhan nhau sau:";
  inkq(1,n-1);
}
```

Phương pháp qui hoạch động: Nhân dãy ma trận

```
bang F:
   0  105  100  124    0
   0    0   70  110  160
   0    0    0   56  110
   0    0    0    0   40
   0    0    0    0    0
bang S:
   0    1    1    3    0
   0    0    2    3    3
   0    0    0    3    3
   0    0    0    0    4
   0    0    0    0    0
----So phep tinh it nhat la: 124

Thu tu nhan nhau sau:((A1 x (A2 x A3)) x A4)
--------------------------------
```

int d[]= {3,5,7,2,4};
int n=5;

```
bang F:
   0  105  100  124  226    0
   0    0   70  110  232  310
   0    0    0   56  198  264
   0    0    0    0   72  180
   0    0    0    0    0  216
   0    0    0    0    0    0
bang S:
   0    1    1    3    3    0
   0    0    2    3    3    3
   0    0    0    3    3    3
   0    0    0    0    4    5
   0    0    0    0    0    5
   0    0    0    0    0    0
----So phep tinh it nhat la: 226

Thu tu nhan nhau sau:((A1 x (A2 x A3)) x (A4 x A5))
--------------------------------
```

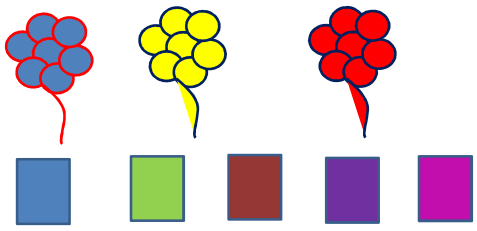
int d[]= {3,5,7,2,4,8};
int n=6;

```
//int d[]={6, 7,23,44,23,56,88,4,12,34, 8};
//int n=10;
```

Phương pháp qui hoạch động: Cắm hoa

Cho n bình hoa và k bó hoa được đánh số thứ tự từ nhỏ đến lớn. Giá trị thẩm mĩ khi cắm hoa j vào bình i là v[i][j]. Mỗi bình chỉ có thể cắm 1 hoa và mỗi hoa chỉ có thể cắm trong 1 bình. Hoa có thứ tự j phải cắm trước hoa thứ j+1. Hãy cắm các hoa vào các bình sao cho tổng thẩm mĩ là lớn nhất.

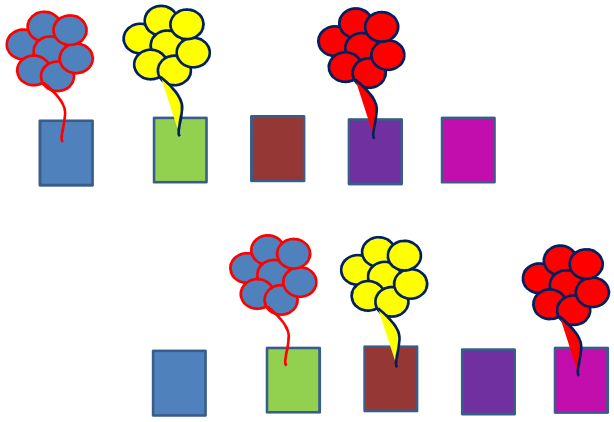
Ví dụ: 3 hoa 5 bình



	b1	b2	b3	b4	b5
H1	3	5	4	7	4
H2	9	6	7	5	9
H3	2	5	11	7	9

Phương pháp qui hoạch động: Cắm hoa

Ví dụ: 3 hoa 5 bình



	b1	b2	b3	b4	b5
H1	3	5	4	7	4
H2	9	6	7	5	9
H3	2	5	11	7	9

Phương pháp qui hoạch động: Cắm hoa

Gọi L[i][j] là tổng thẩm mĩ khi xét đến hoa i và bình j.

Ta có:

- nếu số hoa nhiều hơn số bình (i>j) thì không có cách cắm hợp lý, nên tổng thẩm mĩ = - MaxINT

- Nếu số hoa bằng số bình: thì chỉ có 1 cách cắm và tổng thẩm mĩ là tổng từ v[i][1] đến v[i][i]

Phương pháp qui hoạch động: Cắm hoa (tt)

Ngược lại số hoa ít hơn số bình thì có 2 trường hợp xảy ra:

+ Cắm hoa i vào bình j: Tổng giá trị thẩm mĩ là L(i-1,j-1)+v(i,j) *(bằng tổng giá trị trước khi cắm cộng với giá trị thẩm mĩ khi cắm hoa i vào bình j)*

+ Không cắm hoa i vào bình j (có thể cắm vào bình trước j): giá trị thẩm mĩ của cách cắm là như cũ : L(i,j-1)

LẤY MAX HAI GIÁ TRỊ NÀY

Phương pháp qui hoạch động: Cắm hoa

```
int cam(int i, int j) {
        if(i>j) return –MAXINT;
        else
          if(i==j) {
                        int k=0;  for (int h=1; h<=j; h++) k=k+ v[i][h]];
                        return k;
                   }
        else  return max(cam(i-1, j-1) + v[i][j], cam(i, j-1) );
}
```

Gọi đệ qui: **cam(n, k);**

Phương pháp qui hoạch động: Cắm hoa

Thuật toán:

1. Khởi tạo bảng F gồm k hàng , n cột và cho L[i][j]:= -maxint;
2. Tính toán bảng

```
for (i=1 ; i<= k; i++)
for (j=1 ; j<= n; j++)
        if( i== j) then L[i][j]=sum(i);
        else if (i<j) L[i][j]=max(L[i-1][j-1]+v[i][j],L[i][j-1]);
```

3. return L[k][n];

Phương pháp qui hoạch động: Cắm hoa

Dữ liệu vào HOA.INP

k n
$v_{1,1}$ $v_{1,2}$ $v_{1,3}$ … $v_{1,n}$
$v_{2,1}$ $v_{2,2}$ $v_{2,3}$ … $v_{2,n}$

$v_{k,1}$ $v_{k,2}$ $v_{k,3}$ … $v_{k,n}$

3 5
3 5 4 7 4
9 6 7 5 9
2 5 11 7 9

Dữ liệu ra HOA.OUT

Tổng thẩm mĩ

21

Phương pháp qui hoạch động: Cắm hoa

Dữ liệu vào HOA.INP

k n
$v_{1,1}$ $v_{1,2}$ $v_{1,3}$ … $v_{1,n}$
$v_{2,1}$ $v_{2,2}$ $v_{2,3}$ … $v_{2,n}$

$v_{k,1}$ $v_{k,2}$ $v_{k,3}$ … $v_{k,n}$

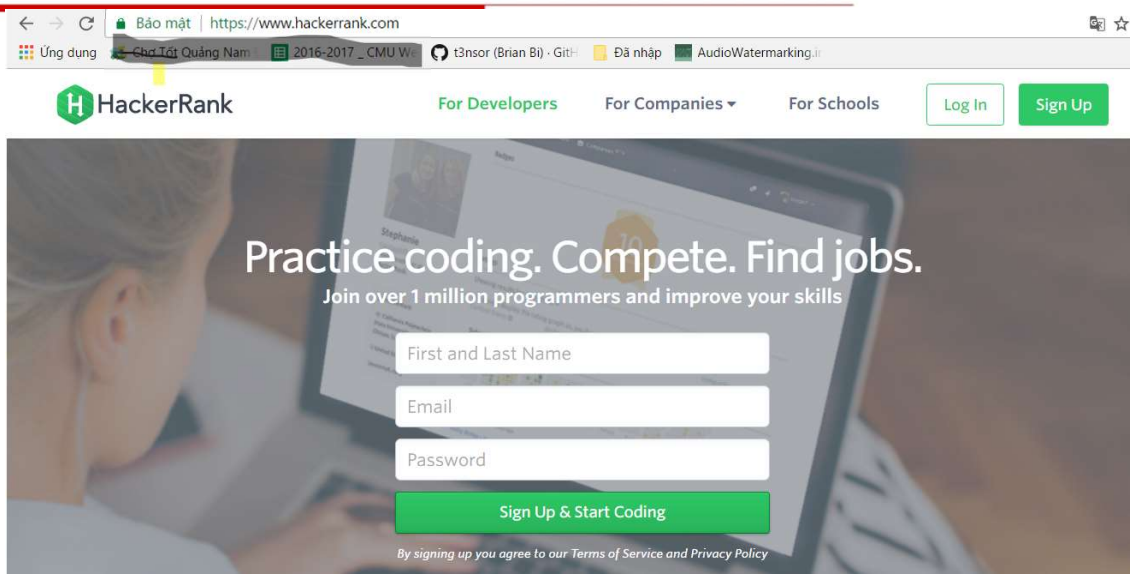
3 5
3 5 4 7 4
9 6 7 5 9
2 5 11 7 9

Dữ liệu ra HOA.OUT

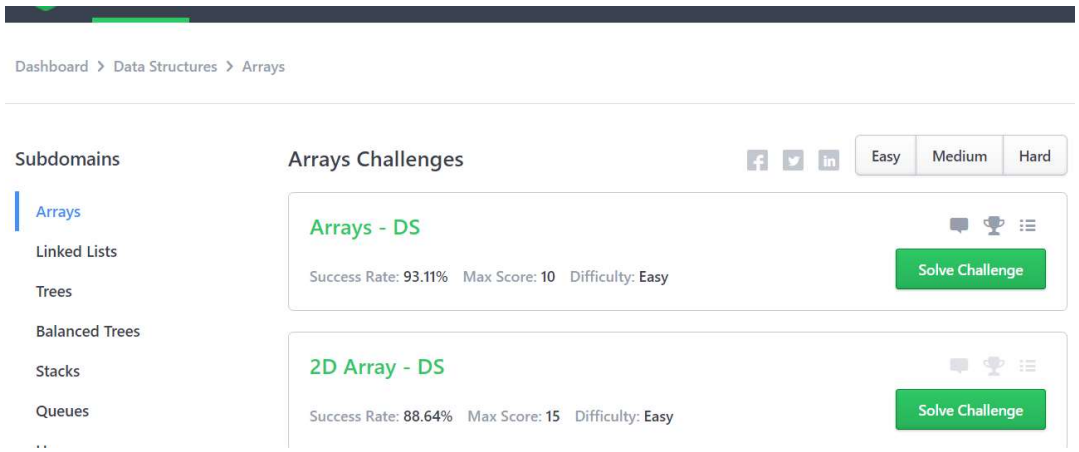
Tổng thẩm mĩ

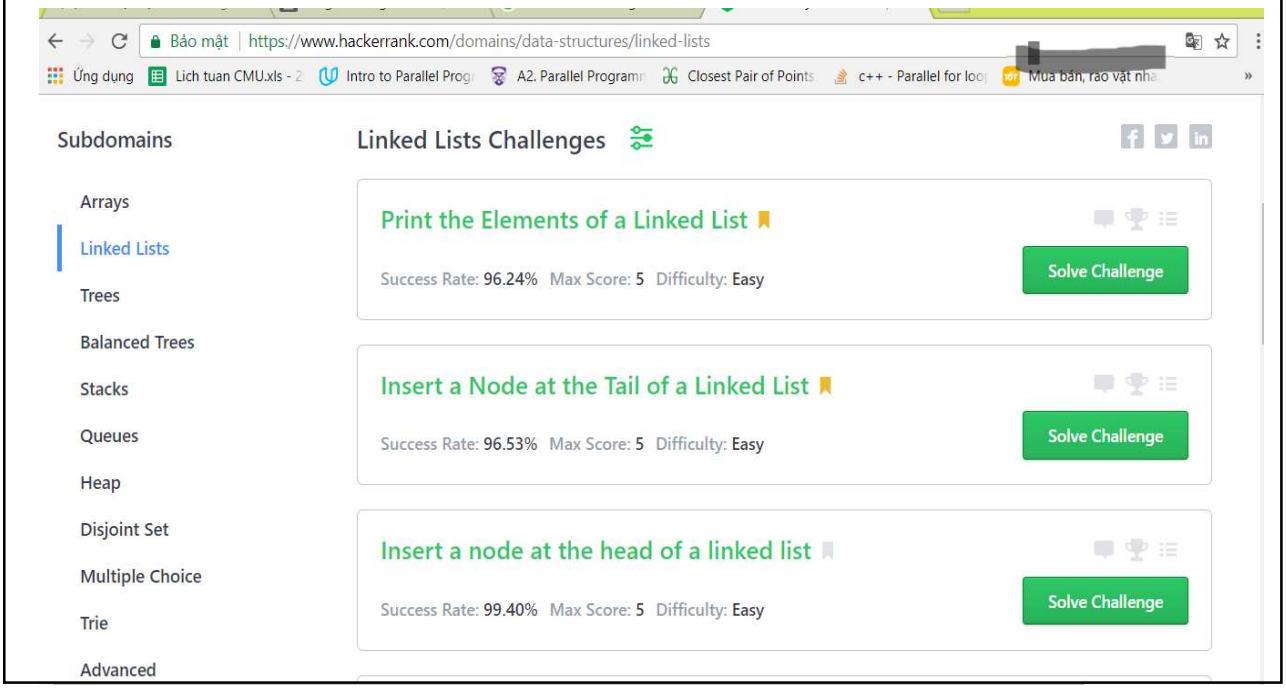
21

Đăng kí luyện tập và làm bài tại hackerrank



Bài tập luyện tập tại hackerrank





Bài tập thảo luận nhóm

Cho ma trận A có kích thước m*n chứa các số nguyên. Hãy tìm ma trận con B của A có kích thước 3*3 sao cho tích các số trong ma trận con là lớn nhất.
Dữ liệu được lấy từ file matran.inp, kết quả ghi ra file matran.out
+ Dòng đầu gồm hai số m, n
+ m dòng tiếp theo, mỗi dòng n số tương ứng với a[i][j]
Kết quả ghi ra file gồm 3 dòng, mỗi dòng gồm 3 số là ma trận cần tìm.

Tên file chương trình **matran.cpp**

Thảo luận về giải pháp để giải quyết bài toán và cho biết độ phức tạp của thuật toán!!!

Tài liệu đọc thêm

https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_dynamic_programming.htm

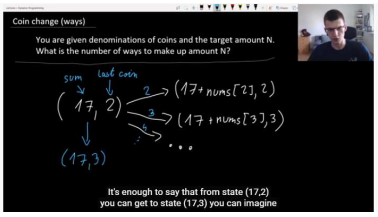
https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/tutorial/

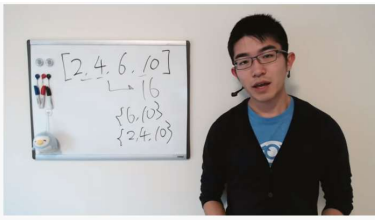
https://codeforces.com/blog/entry/43256

Link YouTube

https://www.youtube.com/watch?v=YBSt1jYwVfU&t=4s
https://www.youtube.com/watch?v=nqlNzOcnCfs

MIT: https://www.youtube.com/watch?v=OQ5jsbhAv_M





Dynamic Programming Interview Question #1 - Find Sets Of Numbers That Add Up To