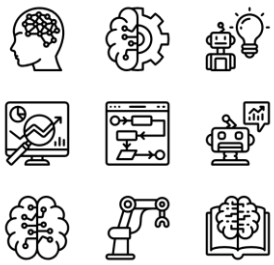
Computer Science for Practicing Engineers

Phương pháp quy hoạch động



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

1

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

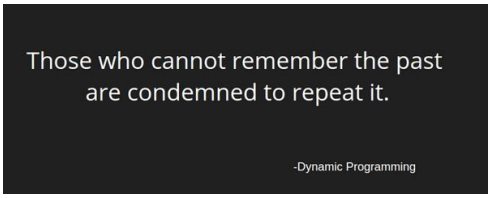
2

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.

Jonathan Paulson explains Dynamic Programming in his amazing Quora answer here.

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!"



"So you didn't need to recount because you remembered there were eight! Dynamic Programming is just a fancy way to say remembering stuff to save time later!"

3

Phương pháp qui hoạch động (dynamic programming)

In programming, Dynamic Programming is a powerful technique that allows one to solve different types of problems in time $O(n^2)$ or $O(n^3)$ for which a naive approach would take exponential time.

The intuition behind dynamic programming is that we trade space for time, i.e. to say that instead of calculating all the states taking a lot of time but no space, we take up space to store the results of all the sub-problems to save time later.

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

4

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.

5

Phương pháp qui hoạch động (dynamic programming)

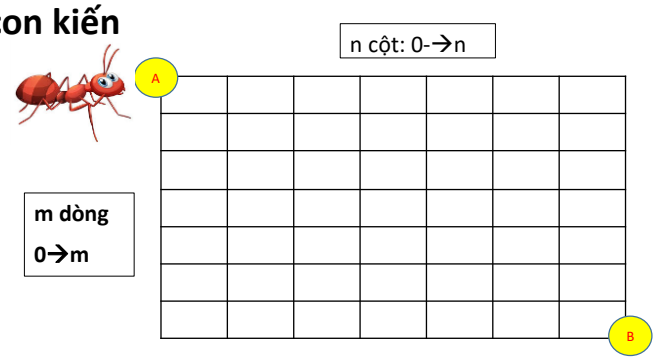
Các bước trong giải bài toán bằng quy hoạch động (4 bước)

- 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.

6

Phương pháp qui hoạch động: Bài toán con kiến

Bài toán **con kiến**

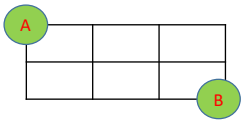


Qui luật khi di chuyển từ A→ B: chỉ đi qua phải hoặc đi xuống

7

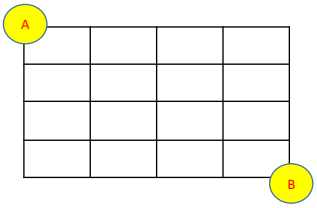
Phương pháp qui hoạch động: Bài toán con kiến

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?

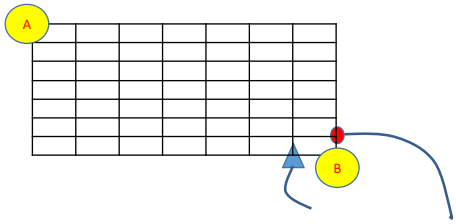


8

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



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

9

Phương pháp qui hoạch động: Bài toán con kiến

Bài toán **con kiến**

```
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);
  }
```

10

Phương pháp qui hoạch động: Bài toán con kiến

```
package cspe_dp_conkien;
public class CSPE_DP_CONKIEN {
    public static 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);
    }
    public static void main(String[] args) {
        int m=5, n=9;
        long t1=0, t2=0;
        t1=System.currentTimeMillis();
        System.out.println("\n So cach di het bang "+ m +"*"+ n+ " la:"+ ck(m,n));
        t1=System.currentTimeMillis();
        System.out.println("\n Thoi gian thuc hien= "+ (t2-t1));
    }
}
```

11

Phương pháp qui hoạch động: Bài toán con kiến

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)** được tính lại 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.

12

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	1							
3	1							
4	1							
5	1							
6	1							

13

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

14

Phương pháp qui hoạch động: Bài toán con kiến

```
long ck_dp(int m, int n)
  {
     long [][]a;
     a= new long[m+1][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];
  }
```

15

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							

16

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							

17

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];

18

Phương pháp qui hoạch động: Bài toán con kiến

```
long ck_dp1(int m, int n)
  {
     long []a;
     a= new 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];
  }
```

19

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Bài toán ví dụ: LCS(m, n) với m là độ dài chuỗi X và n là độ dài chuỗi Y

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.

20

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.
LCS(m, n) 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]**

longest common subsequence

21

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

Bài toán ví dụ: LCS(m, n) với m là độ dài chuỗi X và n là độ dài chuỗi Y

Cho hai xâu X, Y có độ dài là m,n.

Tìm độ dài chuỗi con chung dài nhất.

YÊU cầu:

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

22

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

```
X= "ABADCA";                    Y="DBAXDA";
int LCS(int m, int n)
  {
    if(m==0||n==0) return 0;
    else
      if(Y.charAt(m-1)==X.charAt(n-1))
        return 1+ LCS( m-1, n-1);
      else
        return Math.max(LCS( m-1, n),LCS( m, n-1)) ;
  }
```

23

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

```
public class CSPE_MyP08_QHD_LCS {
  String X, Y;
    int LCS(int m, int n)   {      }
      CSPE_MyP08_QHD_LCS()  {  X= "ABADCA"; Y="DBAXDA";}
  int LCS()   {  return LCS(Y.length(),X.length() );   }
  public static void main(String[] args) {
 CSPE_MyP08_QHD_LCS m = new CSPE_MyP08_QHD_LCS();
    System.out.println(" do dai chuoi con chung dai nhat la: " + m.LCS());
  }
}
```

24

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

```
public class CSPE_MyP08_QHD_LCS{
   String X, Y;
  CSPE_MyP08_QHD_LCS()
   {
        X= “SDKFNSCABADCA";
       Y="DBSFOSAXDA";
       //X= “SDKFNSLDFSDJFLKSDFLSFLSOOIREIWEFOJCASNCMCABADCA";
       //Y="DBSKJFWEJFOSDJFDOIJFMXCSJFSOJDFSJDOFJSODJFOSLKSJDFOSAXDA";
   }
```

Tính thời gian chạy trên máy tính????

25

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

26

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

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

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

27

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

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

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

28

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

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

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

29

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

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

	X	A	B	A	D	C	A
Y	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

30

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

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

Thử cho hai chuỗi sau:

X= “PHQKTAK"
Y=“PKHQAPK";
??????

31

Phương pháp qui hoạch động: Chuỗi con chung dài nhất

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

32

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

33

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

34

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

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

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

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

Xây dựng hàm tính C(m,n)

long tinh (int m, int n) { }

35

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;

36

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

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

37

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							

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

38

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

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

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						

39

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

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

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				
3	0						
4	0						
5	0						
6	0						
7	0						
8	0						

40

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]

41

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=

42

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”

43

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

Chuỗi con đối xứng dài nhất

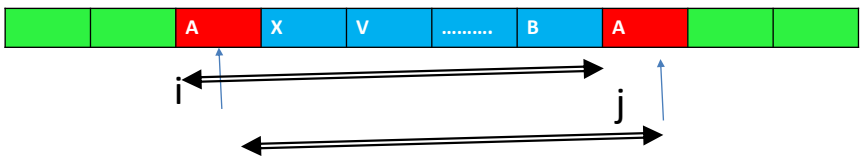
Gọi L(i, j) là độ chuỗi con đối xứng dài nhất từ vị trí i đến vị trí j

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]



44

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

```
public class CSPE_MyP09_QHD_CHUOI_CON_DOI_XUNG_DAI_NHAT {
  String  S;
  CSPE_MyP09_QHD_CHUOI_CON_DOI_XUNG_DAI_NHAT () { S= "ABCDACSBSC";  }
  int L_max(int i, int j)  {     }
  int L_max() { return  L_max(0, S.length()-1);}
   public static void main(String[] args) {
      CSPE_MyP09_QHD_CHUOI_CON_DOI_XUNG_DAI_NHAT
                  m= new CSPE_MyP09_QHD_CHUOI_CON_DOI_XUNG_DAI_NHAT ();
      System.out.println("\n Chuoi doi xung co do dai la: "+ m.L_max());
  }
}
```

45

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

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

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

46

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

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

47

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

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

j

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

48

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

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

	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	1	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	**3+2**
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

i

j

49

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 0**

   Cho j chạy từ i+1 đến n-1

   nếu S[i]==S[j] 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[0] [n-1]**

50

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

```
int L_DP()
{
        int n= S.length();
        int [][] a;
        A= new int[n][n];
         for(int i=0; i<n; i++) a[i][i]=1;
         for(int i=n-1; i>=0; i--)
             for(int j=i+1; j<n; j++)
                 ----
        return a[0][n-1];
}
```

51

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

```
X= "ABADCA";                    Y="DBAXDA";
int LCS_dp(int m, int n)  {
      int a[100][100];
     for(int i=1; i<=m; i++) a[i][0]=0;
      for(int i=1; i<=n; i++) a[0][i] =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-1][j], a[i][j-1]);
      return a[m][n];
 }
```

52

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

A(m,n) , B(n,k) thì C là C(m,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;
}
```

53

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:

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

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

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

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

54

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

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

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

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

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

Nhân ngoặc 1	Nhân ngoặc 2	Nhân cuối	Số phép nhân
105	42	24	171
105	56	84	245
70	30	24	124
56	140	60	256

55

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

56

Phương pháp qui hoạch động: nhân dãy ma trận

Dữ liệu vào : MMATRIX.inp

n
d_0
d_1
d_2
d_3
...
d_n

Dữ liệu ra MMATRIX.out

Số phép nhân phải thực hiện ít nhất

4
3
5
7
2
4

124

57

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-1];

58

Phương pháp qui hoạch động: Nhân dãy ma trận

```
int tinh() {
      int i,j,k,m;
      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-1];
}
```

59

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 20000;
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: "<<tinh()<<"\n";
   cout<<"\n Thu tu nhan nhau sau:";
   inkq(1,n-1);
}
```

60

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)
--------------------------------
Process exited after 0.04084 seconds with return value 0
Press any key to continue . . .
```

```
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))
--------------------------------
Process exited after 0.05666 seconds with return value 0
Press any key to continue . . .
```

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

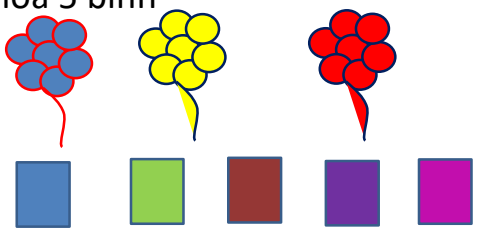
```
int d[]= {3,5,7,2,4,8};
int n=6;
```

61

Phương pháp qui hoạch động: Cắm hoa

Cho n bình hoa và k bó hoa được đánh số thứ tự từ lớn đến nhỏ. 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

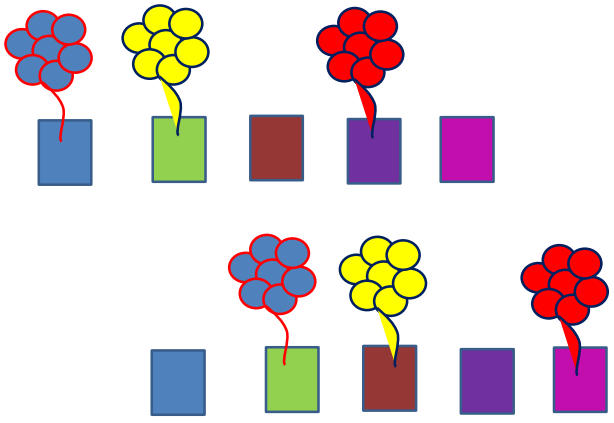


	b1	b2	b3	b4	b5
H1	3	5	4	7	4
H2	9	6	7	5	9
H3	2	5	11	7	9

62

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

63

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, 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]

- 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

64

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);**

65

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];

66

Phương pháp qui hoạch động: Cắm hoa

Dữ liệu vào HOA.INP

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

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

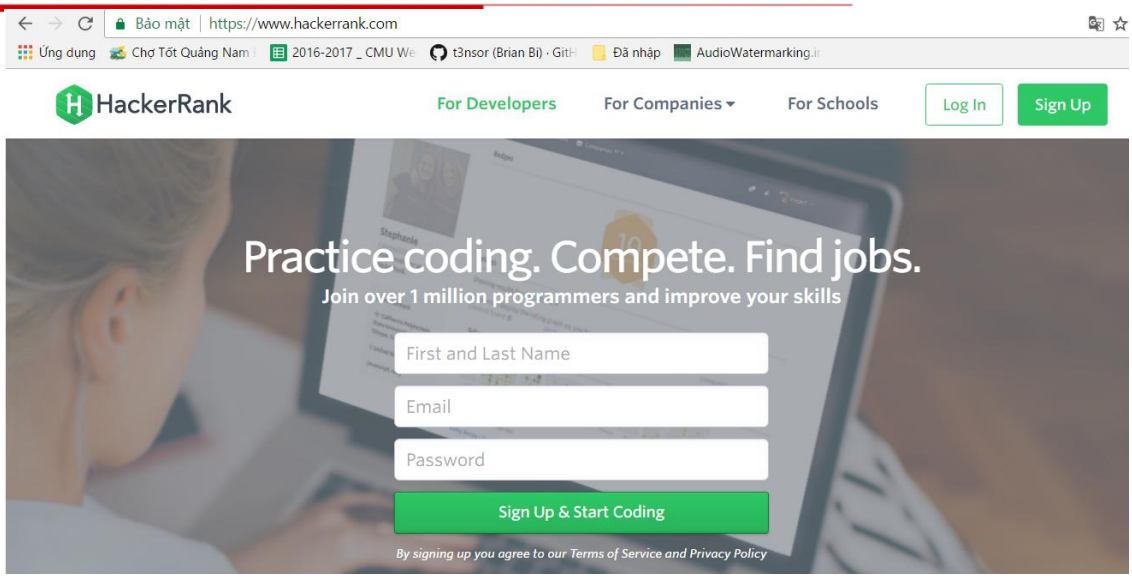
3 5
3 5 4 7 4
9 6 7 5 9
2 5 11 7 9

Dữ liệu ra HOA.OUT

Tổng thẩm mĩ

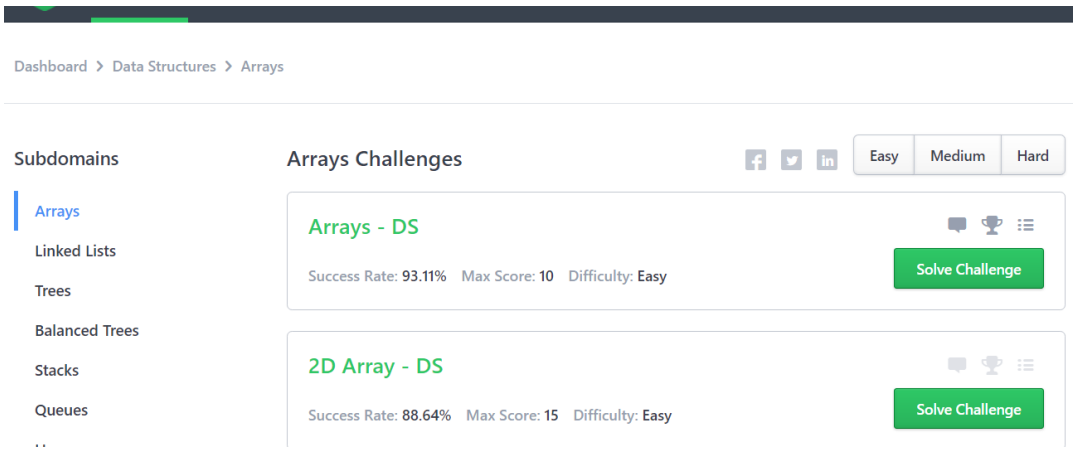
67

Đăng kí luyện tập và làm bài tại hackerank

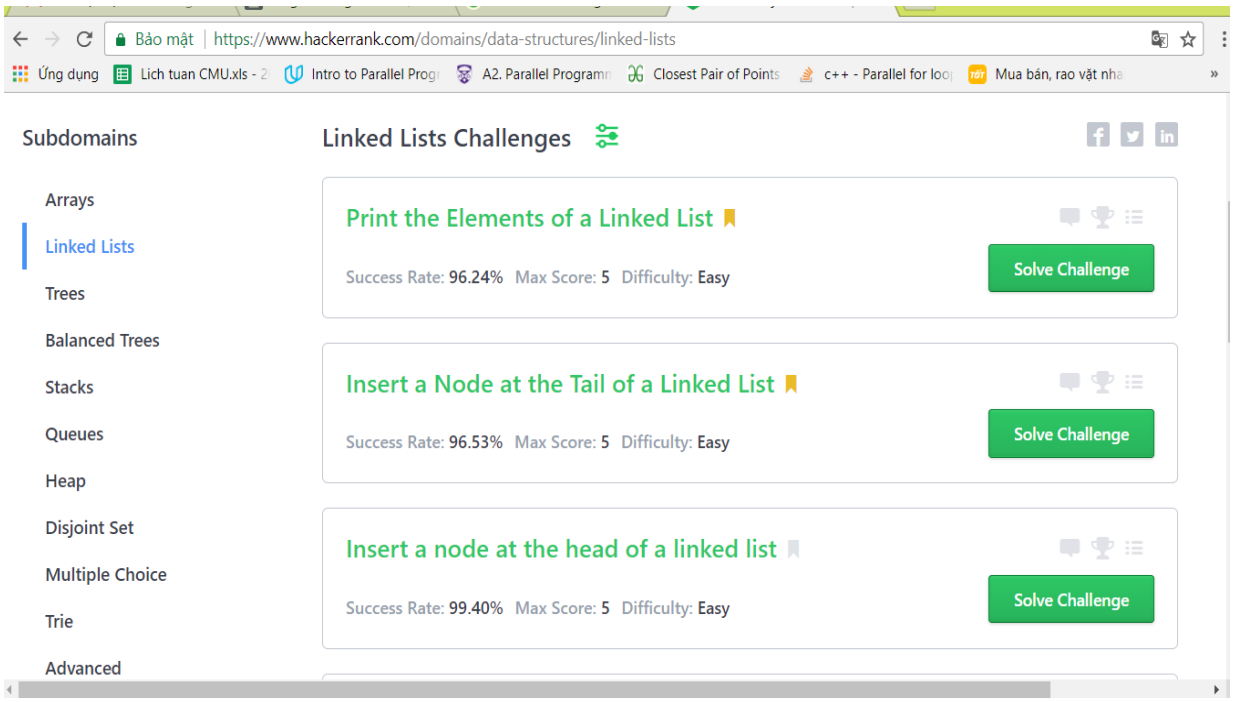


68

Bài tập luyện tập tại hackerank



69



70

Bài tập về nhà

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** hoặc **matran.java**

71

Tài liệu đọc thêm

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

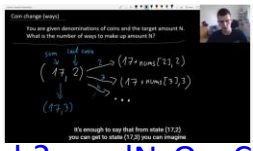
https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/tutorial/

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

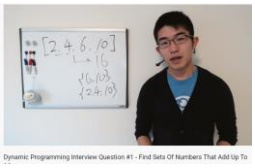
72

Link YouTube

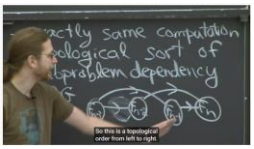
https://www.youtube.com/watch?v=YBSt1jYwVfU&t=4s



https://www.youtube.com/watch?v=nqlNzOcnCfs



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



73