Analysis and Design of Algorithms

Lecture 9,10

Dynamic Programming

Lecturer: Ha Dai Duong
duonghd@mta.edu.vn

Nội dung

1. Lược đồ chung
2. Bài toán tính số Fibonaci
3. Bài toán cái túi
4. Bài toán dãy con có tổng lớn nhất
5. **Bài toán tìm xâu con chung dài nhất**
6. Đường đi ngắn nhất - TT Floyd
7. Cây nhị phân tìm kiếm tối ưu

Bài toán

- Cho hai xâu
  - $X = (x_1, x_2, \dots, x_m)$ và
  - $Y = (y_1, y_2, \dots, y_n)$
- Hãy tìm xâu con chung dài nhất của hai dãy X và Y.
- Ví dụ
  - X = KHOA HOC
  - Y = HOA HONG

HOA HO

Ý tưởng thuật toán

- Phân rã:
  - m: chiều dài xâu X, n: chiều dài xâu Y
  - Với mỗi $0 \le i \le m$ và $0 \le j \le n$ gọi C[i, j] là độ dài của dãy con chung dài nhất của hai dãy

    $\mathbf{X_i}=x_1x_2\ldots x_i$ và $\mathbf{Y_j}=y_1y_2\ldots y_j$

    (Qui ước X_0 = rỗng, Y_0= rỗng)
  - Khi đó C[m,n] là chiều dài xâu con chung dài nhất của X và Y.
- Bài toán con: C[0,j]=0 j=1..n, C[i,0] = 0 i=1..m

2/2/2017 4

Tổng hợp

- Với $i > 0, j > 0$ tính C[i, j]
  - Nếu $x_i = y_j$ thì dãy con chung dài nhất của X_i và Y_j sẽ thu được bằng việc bổ sung x_i (cũng là y_j) vào dãy con chung dài nhất của hai dãy X_{i-1} và Y_{j-1}

    ➡ **C[i,j] = C[i-1,j-1]+1**
  - Nếu $x_i \neq y_j$ thì dãy con chung dài nhất của X_i và Y_j sẽ là dãy con dài hơn trong hai dãy con chung dài nhất của (X_{i-1} và Y_j) và của (X_i và Y_{j-1})

    ➡ **C[i,j] = Max{C[i-1,j], C[i,j-1]}**

2/2/2017 5

Cài đặt

```
Procedure LCS(X,Y)
{
  For i =1 to m do c[i,0]=0;
  For j =1 to n do c[0,j ]=0;
  For i =1 to m do
       for j = 1 to n do
          If xi = yj then { c[i,j]=c[i-1,j-1]+1; b[i,j]='↖' }
          else
             If c [i-1,j]≥ c[i,j-1] then { c[i,j]=c[i-1,j]; b[i,j]='↑';}
             else { c[i,j]=c[i,j-1]; b[i,j]='←';}
  }
```

2/2/2017 6

Minh họa

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
H								
O								
A								
H								
O								
N								
G								

Khởi tạo

- Y= KHOAHOC, X= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0							
O	0							
A	0							
H	0							
O	0							
N	0							
G	0							

Lặp

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0						
O	0							
A	0							
H	0							
O	0							
N	0							
G	0							

Lặp …

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0	1					
O	0							
A	0							
H	0							
O	0							
N	0							
G	0							

Lặp …

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0	1	?				
O	0							
A	0							
H	0							
O	0							
N	0							
G	0							

Lặp …

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0	1	1				
O	0							
A	0							
H	0							
O	0							
N	0							
G	0							

Lặp …

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0	1	1	1	1	1	1
O	0							
A	0							
H	0							
O	0							
N	0							
G	0							

Lặp …

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0	1	1	1	1	1	1
O	0	0						
A	0							
H	0							
O	0							
N	0							
G	0							

Lặp …

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0	1	1	1	1	1	1
O	0	0	?					
A	0							
H	0							
O	0							
N	0							
G	0							

Lặp …

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0	1	1	1	1	1	1
O	0	0	1					
A	0							
H	0							
O	0							
N	0							
G	0							

Lặp …

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0	1	1	1	1	1	1
O	0	0	1	2				
A	0							
H	0							
O	0							
N	0							
G	0							

Kết thúc

- X= KHOAHOC, Y= HOAHONG

		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0	1	1	1	1	1	1
O	0	0	1	2	2	2	2	2
A	0	0	1	2	3	3	3	3
H	0	0	1	2	3	4	4	4
O	0	0	1	2	3	4	5	5
N	0	0	1	2	3	4	5	5
G	0	0	1	2	3	4	5	5

Kết thúc

- X= KHOAHOC, Y= HOAHONG

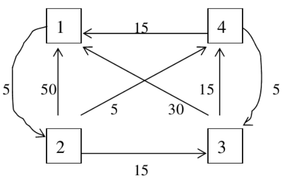
		K	H	O	A	H	O	C
	0	0	0	0	0	0	0	0
H	0	0	1	1	1	1	1	1
O	0	0	1	2	2	2	2	2
A	0	0	1	2	3	3	3	3
H	0	0	1	2	3	4	4	4
O	0	0	1	2	3	4	5	5
N	0	0	1	2	3	4	5	5
G	0	0	1	2	3	4	5	5

Nội dung

1. Lược đồ chung
2. Bài toán tính số Fibonaci
3. Bài toán cái túi
4. Bài toán dãy con có tổng lớn nhất
5. Bài toán tìm xâu con chung dài nhất
6. **Đường đi ngắn nhất - TT Floyd**
7. Cây nhị phân tìm kiếm tối ưu

Bài toán

- Đồ thị G=(V,E)
  - Đơn đồ thị liên thông (vô hướng hoặc có hướng)
  - Có trọng số.
  - V: Tập đỉnh
  - E: Tập cạnh
- Tìm đường đi ngắn nhất từ giữa một cặp đỉnh nào đó của G.



Thuật toán Floyd

- Tư tưởng:
  - Nếu k nằm trên đường đi ngắn nhất từ i đến j thì đường đi từ i đến k và từ k đến j cũng ngắn nhất (Nguyên lý Bellman).
- Phân rã:
  - **n** là số đỉnh của G
  - Gọi **d[i,j]** là đường đi ngắn nhất từ đỉnh i đến đỉnh j
  - Qui ước **p_k[i,j]** với (k=0..n) lưu giá trị từ 0 .. k (đỉnh) thể hiện đường đi ngắn nhất từ i đến j có qua đỉnh p_k[i,j]

Thuật toán Floyd

- Phân rã:
  - **n** là số đỉnh của G, **d[i,j]**, **p_k[i,j]**
  - p_k[i,j] = 0 đường đi ngắn nhất từ i đến j không đi qua p_k[i,j],
  - p_k[i,j] !=0 đường đi ngắn nhất từ i đến j đi qua p_k[i,j]
  - Khi k = n thì p_k[i,j] cho biết đường đi cần tìm.
- Bài toán con:
  - d[i,j] = a[i,j]
  - p_0[i,j] = 0

Tổng hợp

- Nếu d[i,j] là đường đi ngắn nhất từ i đến j đã xét ở bước k-1 (đã xét đi qua từ đỉnh 1 đến đỉnh k-1).
- Ở bước k:

  d[i,j] = min (d[i,j], d[i,k]+d[k,j])

Cài đặt

- Biểu diễn đồ thị G qua ma trận trọng số cạnh

  a= $(a_{uv})_{nxn}$;

$$a_{uv} = \begin{cases} trọng\ số\ của\ (u,v);(u,v) \in E; \\ \infty;(u,v) \notin E; \end{cases}$$

- Khởi tạo

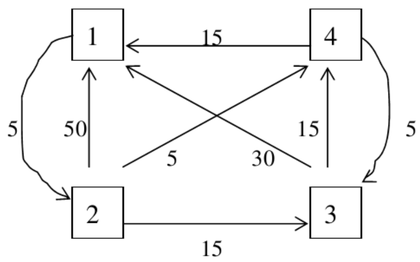
  d[i,j] = a[i, j]

  p[i,j] = 0


```
void floyd()
{
        int i, j, k;
        // Khoi dong ma tran d va p
        for (i = 1; i<= n; i++)
                for (j = 1; j<= n; j++)
                {
                        d[i][j] = a[i][j];
                        p[i][j] = 0;
                }
        for (k = 1; k <= n; k++)  // Tính ma trận d và p ở bước lặp k
                for (i = 1; i <= n; i++)
                        if ( d[i][k] > 0 && d[i][k] < vc )
                                for (j = 1; j<= n; j++)
                                        if ( d[k][j] > 0 && d[k][j] < vc )
                                                if (d[i][k] + d[k][j] < d[i][j] )
                                                {
                                                        d[i][j] = d[i][k] + d[k][j];
                                                        p[i][j] = k;
                                                }
}
```

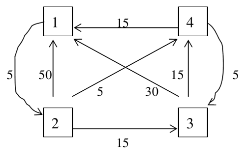

Minh họa

Tìm đường đi ngắn nhất giữa các cặp đỉnh của đồ thị :



Khởi tạo

Tìm đường đi ngắn nhất giữa các cặp đỉnh của đồ thị :



d^0

	1	2	3	4
1	0	5	∞	∞
2	50	0	15	5
3	30	∞	0	15
4	15	∞	5	0

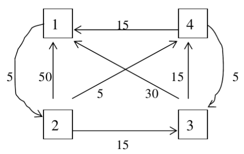
p^0

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

2/2/2017 28

Với k = 1

Tìm đường đi ngắn nhất giữa các cặp đỉnh của đồ thị :



d^1

	1	2	3	4
1	0	5	∞	∞
2	50	0	15	5
3	30	35	0	15
4	15	20	5	0

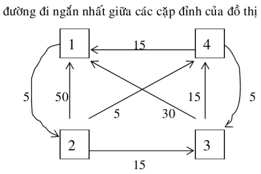
p^1

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

2/2/2017 29

Với K = 2

Tìm đường đi ngắn nhất giữa các cặp đỉnh của đồ thị :



d^2

	1	2	3	4
1	0	5	20	10
2	50	0	15	5
3	30	35	0	15
4	15	20	5	0

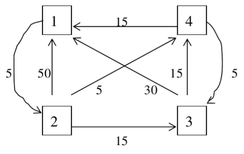
p^2

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

2/2/2017 30

Với K = 3

Tìm đường đi ngắn nhất giữa các cặp đỉnh của đồ thị :



d^3

	1	2	3	4
1	0	5	20	10
2	45	0	15	5
3	30	35	0	15
4	15	20	5	0

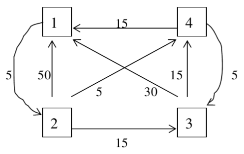
p^3

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

2/2/2017 31

Với K = 4

Tìm đường đi ngắn nhất giữa các cặp đỉnh của đồ thị :



d^4

	1	2	3	4
1	0	5	15	10
2	20	0	10	5
3	30	35	0	15
4	15	20	5	0

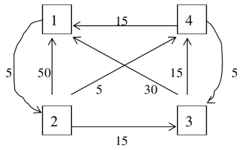
p^4

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

2/2/2017 32

Kết quả

Tìm đường đi ngắn nhất giữa các cặp đỉnh của đồ thị :



d^4

	1	2	3	4
1	0	5	15	10
2	20	0	10	5
3	30	35	0	15
4	15	20	5	0

p^4

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

Đường đi từ 1->3 ?
p[1,3] = 4
Đường đi từ 1->4 ?
p[1,4] = 2

➡ **Đường đi từ 1->3: 1 -> 2 -> 4 -> 3 (15)**

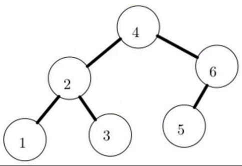
2/2/2017 33

Nội dung

1. Lược đồ chung
2. Bài toán tính số Fibonaci
3. Bài toán cái túi
4. Bài toán dãy con có tổng lớn nhất
5. Bài toán tìm xâu con chung dài nhất
6. Đường đi ngắn nhất - TT Floyd
7. **Cây nhị phân tìm kiếm tối ưu**

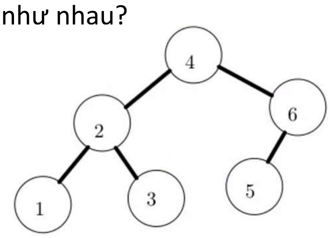
Cây nhị phân tìm kiếm

- Cây nhị phân tìm kiếm (binary search tree) là một cây nhị phân có tính chất sau:
  – Mỗi nút là một khóa tìm kiếm
  – Với mỗi cây con, khóa của nút gốc lớn hơn khóa của mọi nút của cây con trái và nhỏ hơn khóa của mọi nút của cây con phải
- Ví dụ



Cây nhị phân tìm kiếm …

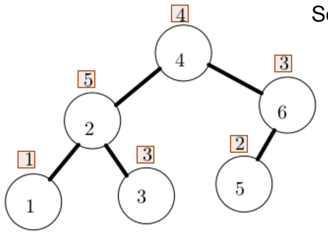
- Nếu số lần tìm kiếm (tần xuất) các khóa trên cây là như nhau?



Cấu trúc của cây không quan trọng

Cây nhị phân tìm kiếm …

- Số lần tìm kiếm các khóa khác nhau:



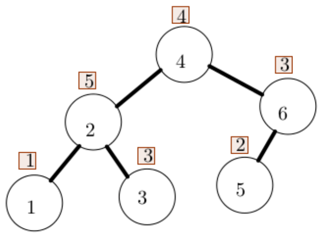
Số lần duyệt qua nút có khóa là:

(4) : 1+5+3 +4 + 2+3 = 18
(2) : 1+5+3 = 9
(6) : 2+3 = 5
(1) : 1 = 1
(3) : 3 = 3
(5) : 2 = 2
Tổng **= 38**

Cấu trúc cây quan trọng

Cây nhị phân tìm kiếm tối ưu

- Vậy cấu trúc nào để cây nhị phân tìm kiếm có số lần duyệt nhỏ nhất (tối ưu)?

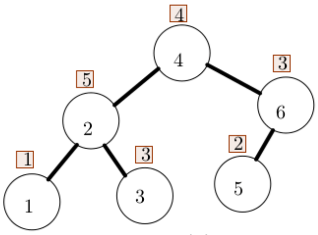


Bài toán

- Cho mảng **A[1,2,…,n]** đã sắp xếp theo chiều tăng dần trong đó các phần tử đôi một khác nhau. Mỗi phần tử **A[i]** có tần số tìm kiếm **f[i]** (i=1..n).

Tìm cây nhị phân với khóa là các phần tử của mảng A sao cho tổng số lượng các phép so sánh là nhỏ nhất

Tiếp cận bằng QHD



(4) : 1+5+3 +4 + 2+3 = 18

- **Nhận xét:** Số lần duyệt ở gốc không phụ thuộc vào cấu trúc cây và SumF(n)= f[1]+f[2]+..+f[n]

Phân rã

- Gọi Op(1..n) là số phép so sánh của cây nhị phân tìm kiếm tối ưu của mảng A[1..n]. Nếu A[r] là khóa của nút gốc, ta có:

  Op(1..n) = Op(1..r-1) + Op(r+1..n) + SumF(1..n)

  (SumF(1..n)= f[1]+f[2]+..+f[n])

  Vì Op(1..n) là tối ưu nên ta có

  Op(1..n) = min {Op(1..r-1) + Op(r+1..n): r=1..n} + SumF(1..n)

Phân rã ...

- Gọi C[i,j] là số phép so sánh của cây nhị phân tìm kiếm tối ưu cho mảng con A[i..j]
- Đặt F[i,j] = f[i]+f[i+1]+..+f[j])
- Ta có

  C[i,j] = min{C[i,r-1] + C[r+1,j]: r=i..j} + F[i,j]

Tiếp cận bằng QHD ...

- Bài toán con
  C[i,i] = F[i,i]

- Tổng hợp:
  C[i,j] = min{C[i,r-1] + C[r+1,j]} + F[i,j]

Tính F[i,j]

- Hàm $\text{PreCompute}(f[1,2,\ldots,n])$
  Tính F[i,j]

$\underline{\text{PreCompute}}(f[1,2,\ldots,n])$:
for $i \leftarrow 1$ to n
  $F[i,i-1] \leftarrow 0$
  for $j \leftarrow i$ to n
    $F[i,j] \leftarrow F[i,j-1] + f[j]$

Tính C[i,j]

- Hàm $\text{ComputeCost}(i, i+d)$
  Tính C[i,j] = min{C[i,r-1] + C[r+1,j]} + F[i,j]

$\underline{\text{ComputeCost}}(i,j)$:
$C[i,j] \leftarrow +\infty$
for $r \leftarrow i$ to j
  tmp $\leftarrow C[i,r-1] + C[r+1,j]$
  if tmp $\leq C[i,j]$
    $C[i,j] \leftarrow$ tmp
    $R[i,j] \leftarrow r$
$C[i,j] \leftarrow C[i,j] + F[i,j]$

Thuật toán

$\underline{\text{OptBinSearchTree}}(A[1, 2, \ldots, n])$:
  $\text{PreCompute}(f[1, 2, \ldots, n])$
  for $i \leftarrow 1$ to n
    $C[i, i] \leftarrow F[i][i]$
    $R[i, i] \leftarrow i$
  for $d \leftarrow 1$ to $n - 1$
    for $i \leftarrow 1$ to $n - d$
      $\text{ComputeCost}(i, i + d)$
  return $C[1, n]$

2/2/2017 46

Độ phức tạp tính toán

- Hàm $\text{PreCompute}(f[1, 2, \ldots, n])$
  Là $O(n^2)$
- Hàm $\text{ComputeCost}(i, i + d)$
  Là O(n)
- Hàm $\underline{\text{OptBinSearchTree}}(A[1, 2, \ldots, n])$
  Là $O(n^3)$

2/2/2017 47

Mảng R[i,j]

- Mảng R[i,j] trong thuật toán trên lưu lại gốc của cây nhị phân tìm kiếm tối ưu của mảng con A[i…j].
- Mảng R[i,j] có thể được sử dụng để truy vết để tìm ra cây nhị phân tìm kiếm tối ưu (bài tập)

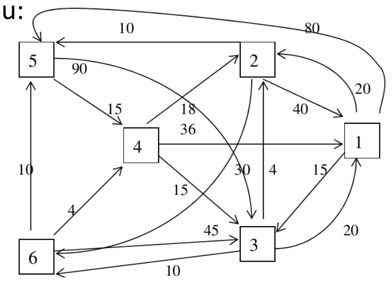
2/2/2017 48

Bài tập

1. Thực hiện và ghi kết quả từng bước thuật toán tìm xâu con dài nhất của 2 xâu: **TOANHOC** và **KHONHOC**
2. Thực hiện và ghi kết quả từng bước thuật toán tìm xâu con dài nhất của 2 xâu: **TINHYEU** va **HOAHONG**

Bài tập

3. Thực hiện và ghi kết quả tường bước thuật toán Floyd tìm đường đi ngắn nhất trên đồ thị sau:



Bài tập

4. Cài đặt thuật toán tìm xâu con dài nhất của 2 xâu ký tự. Đánh giá độ phức tạp bằng thực nghiệm và so sánh với lý thuyết.
5. Cài đặt thuật toán Floyd tìm đường đi ngắn nhất trên đồ thị. Đánh giá độ phức tạp bằng thực nghiệm và so sánh với lý thuyết.
6. Cài đặt thuật toán xây dựng cây tìm kiếm nhị phân tối ưu. Đánh giá độ phức tạp bằng thực nghiệm và so sánh với lý thuyết.

Nội dung đã học

1. Lược đồ chung
2. Bài toán tính số Fibonaci
3. Bài toán cái túi
4. Bài toán dãy con có tổng lớn nhất
5. Bài toán tìm xâu con chung dài nhất
6. Đường đi ngắn nhất - TT Floyd
7. Cây nhị phân tìm kiếm tối ưu