Analysis and Design of Algorithms

Lecture 9,10
Dynamic Programming

Lecturer: Tống Minh Đức
ductm@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

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

Chia để trị

- Khi chia bài toán thành các bài toán con, trong nhiều trường hợp, các bài toán con khác nhau lại chứa các bài toán con hoàn toàn giống nhau.
- Ví dụ: Tính số **Fibonaci** thứ **n, F(n)**:
  - F(0)=0, F(1)=1
  - F(n)=F(n-2)+F(n-1) với n>1
  - F(2)=1, F(3)= 2, F(4) = 3 , F(5)=5, F(6)=8 ...

Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Chia để trị ...

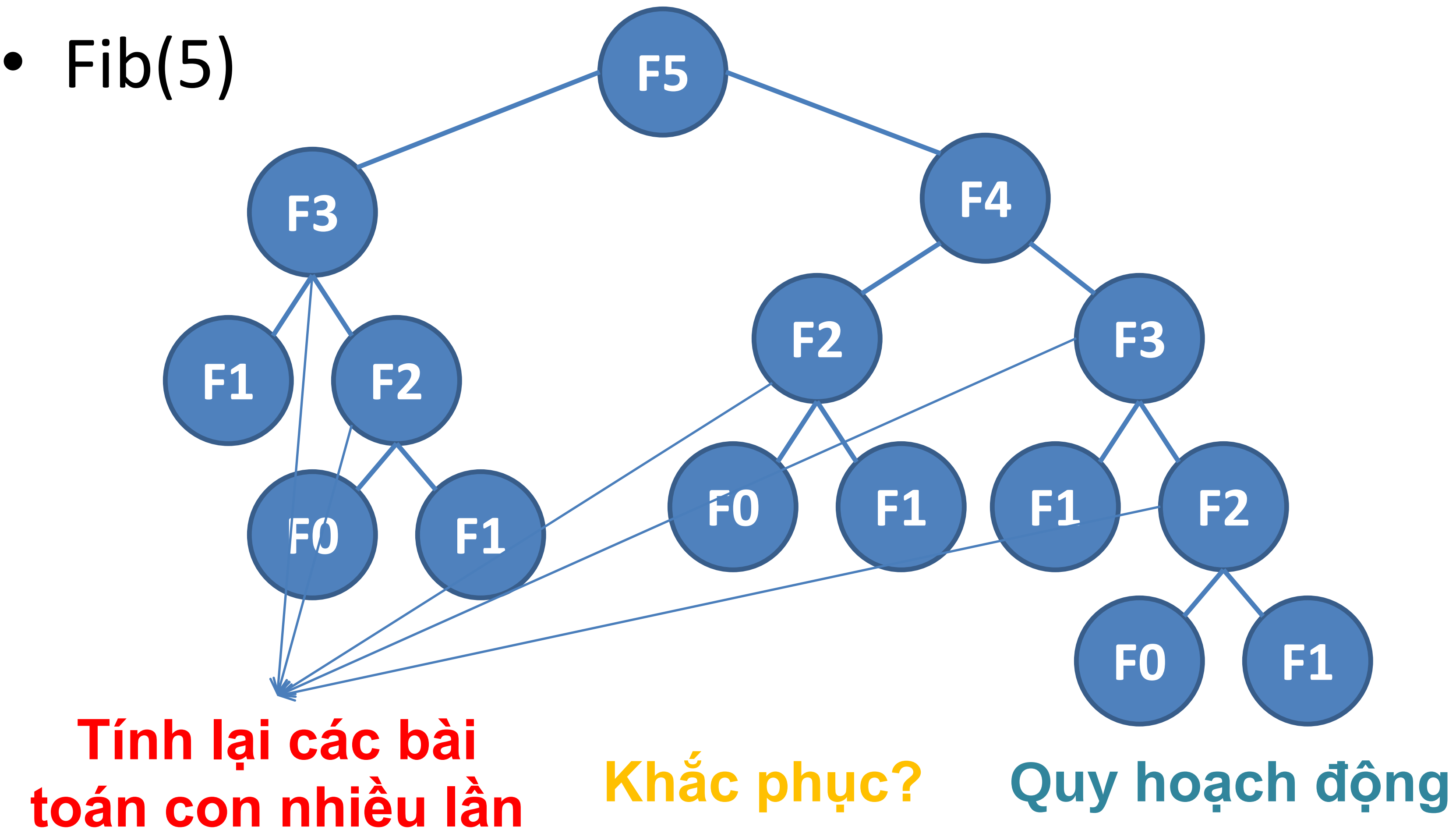
- Fib(n): Tiếp cận theo hướng chia để trị

$$F(n) = F(n-1) + F(n-2)$$

```
Function Fib(n)
{
    If n<2 then
        return n;
    else
        return Fib(n-1) + Fib(n-2);
}
```

Chia để trị ...

- Fib(5)

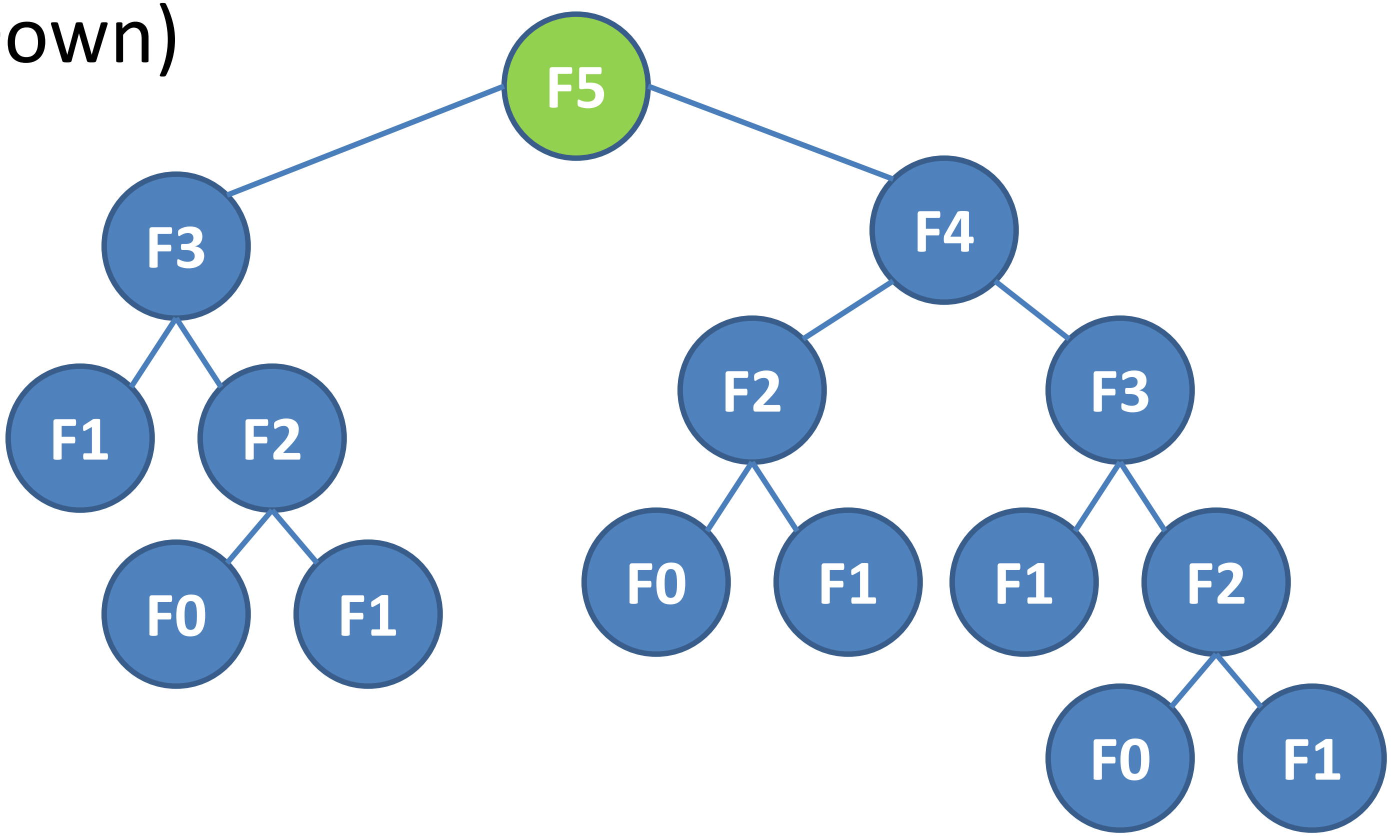


Qui hoạch động

- Là một kĩ thuật thiết kế thuật toán theo kiểu chia bài toán lớn thành các bài toán con, sử dụng lời giải của các bài toán con để tìm lời giải cho bài toán ban đầu.
- Khác với chia để trị, quy hoạch động, thay vì gọi đệ quy, sẽ **tính trước** lời giải của các bài toán con và lưu vào bộ nhớ (thường là một mảng), và sau đó lấy lời giải của bài toán con ở trong mảng đã tính trước để giải bài toán lớn

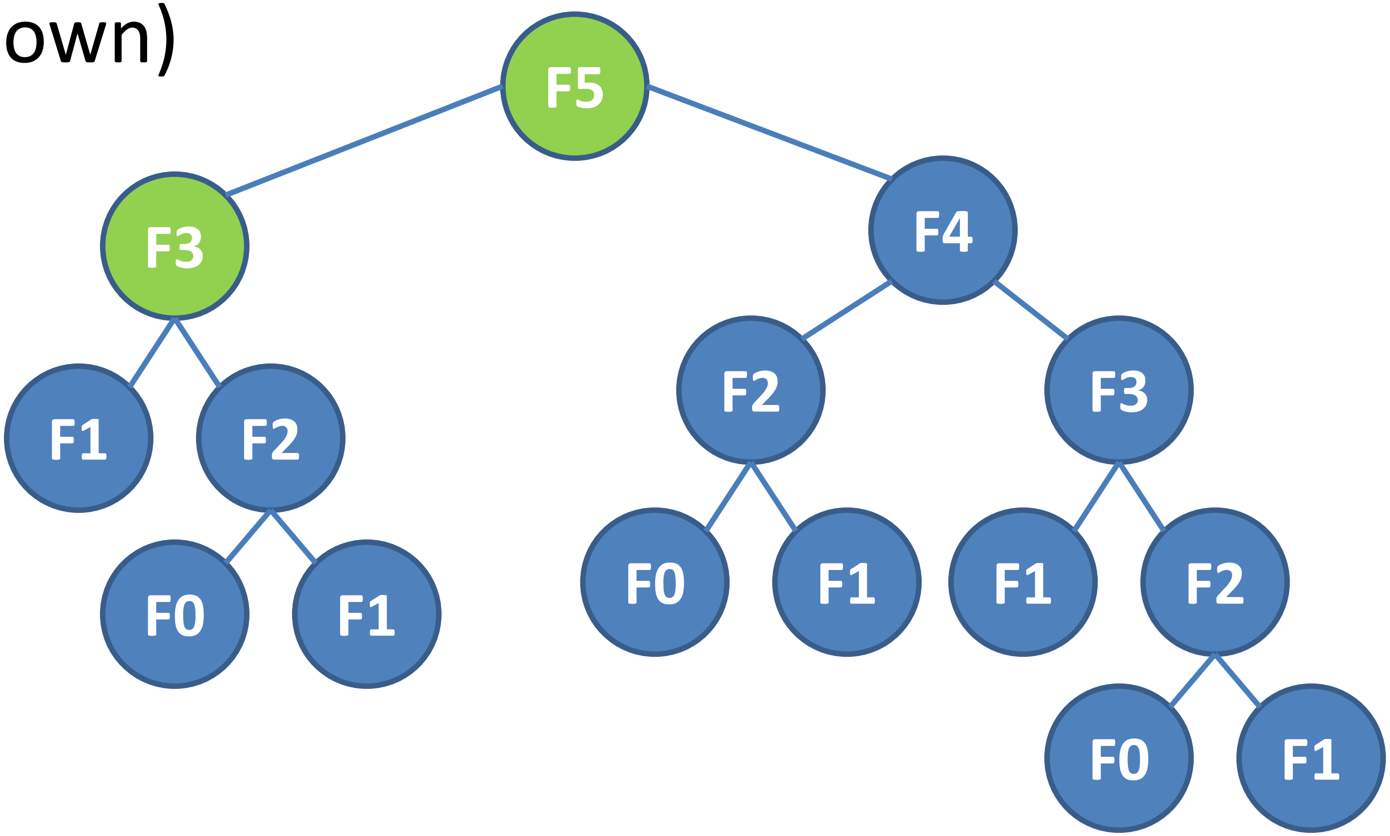
Qui hoặc động vs Chia để trị

- Chia để trị: Tiếp cận từ trên xuống (Top-Down)



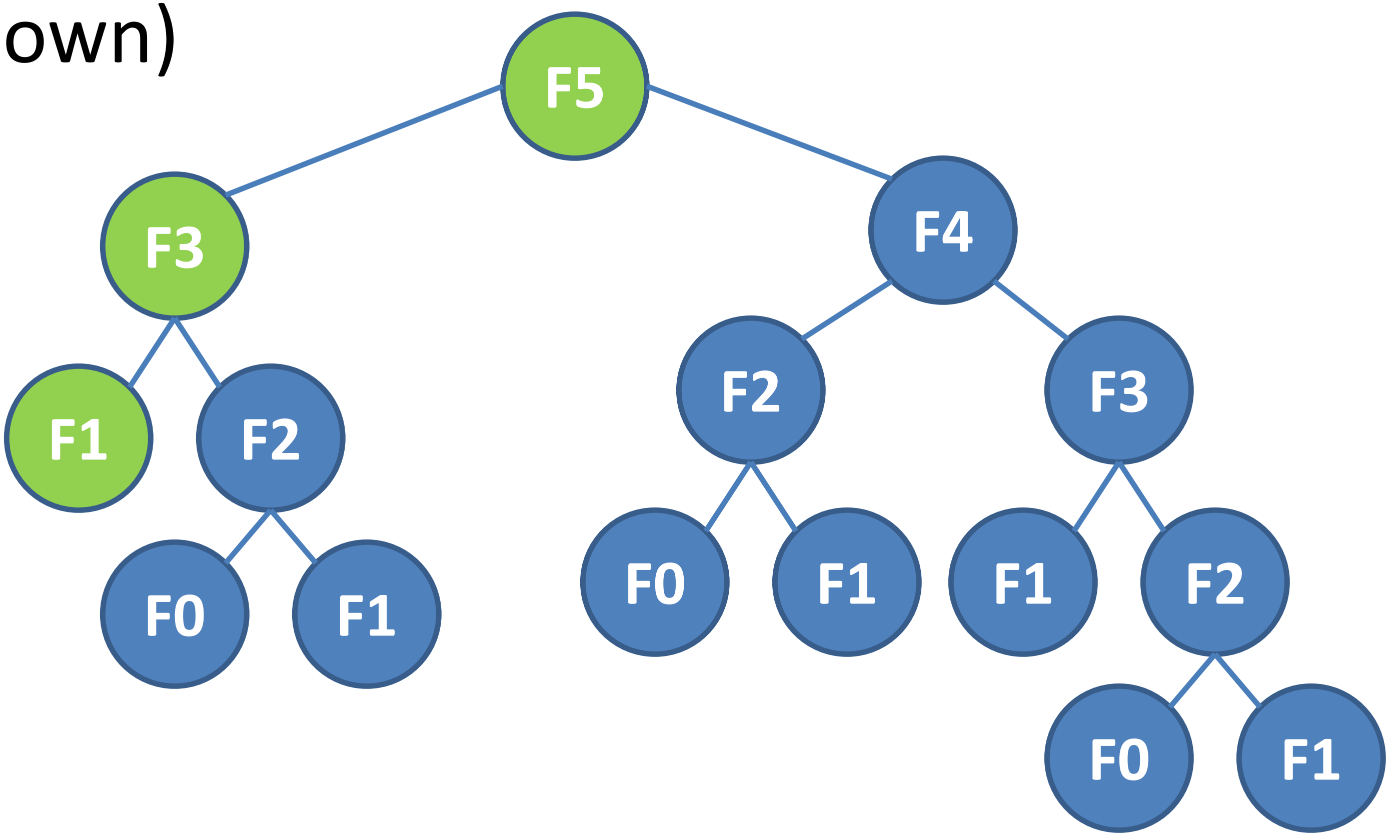
Qui hoặc động vs Chia để trị

- Chia để trị: Tiếp cận từ trên xuống (Top-Down)



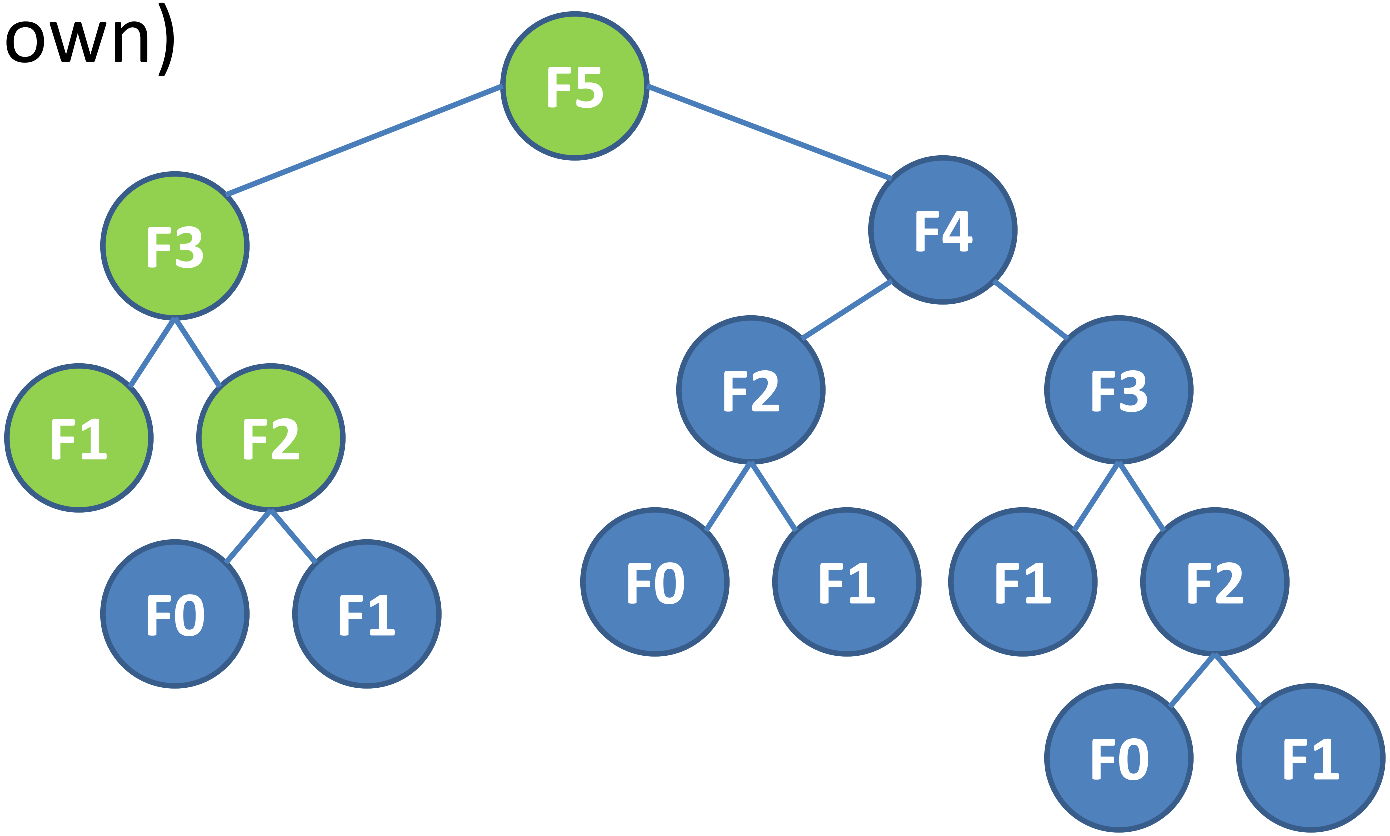
Qui hoặc động vs Chia để trị

- Chia để trị: Tiếp cận từ trên xuống (Top-Down)



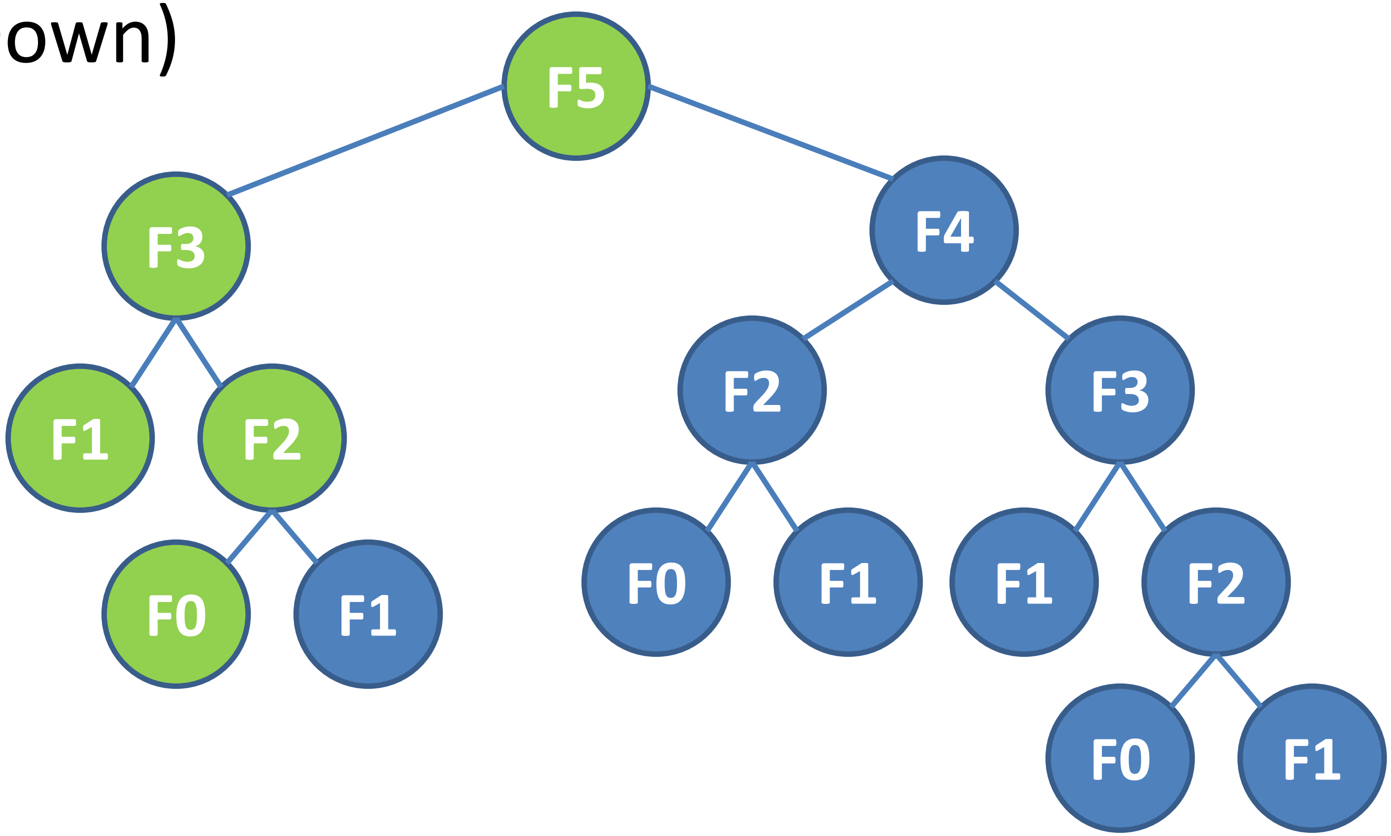
Qui hoặc động vs Chia để trị

- Chia để trị: Tiếp cận từ trên xuống (Top-Down)



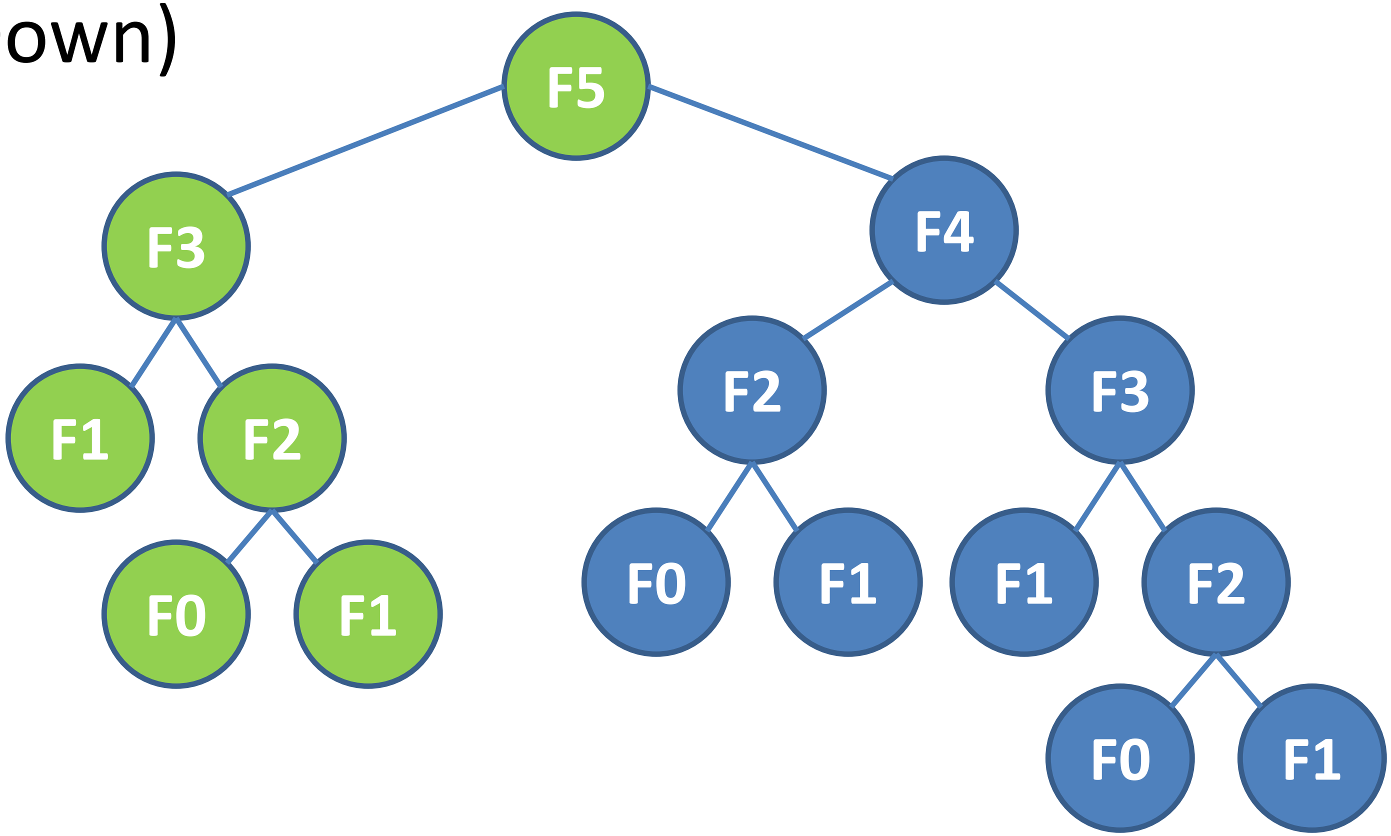
Qui hoặc động vs Chia để trị

- Chia để trị: Tiếp cận từ trên xuống (Top-Down)



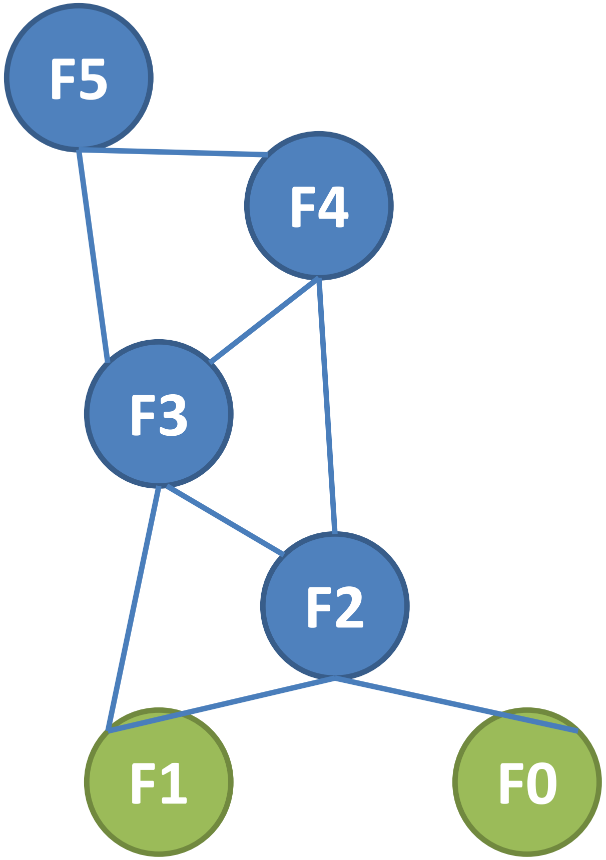
Qui hoạch động vs Chia để trị

- Chia để trị: Tiếp cận từ trên xuống (Top-Down)



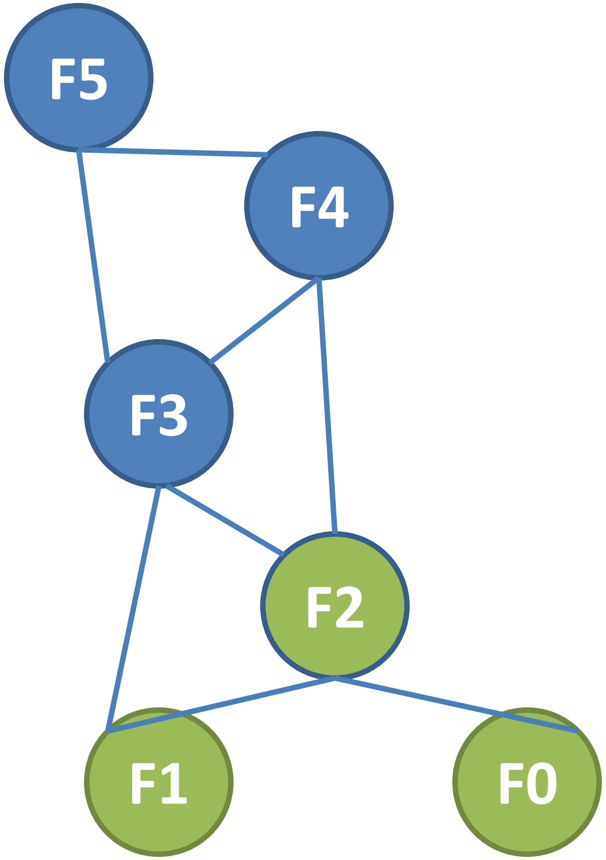
Qui hoặc động vs Chia để trị

- Quy hoạch động: Tiếp cận từ dưới lên (Bottom-up)



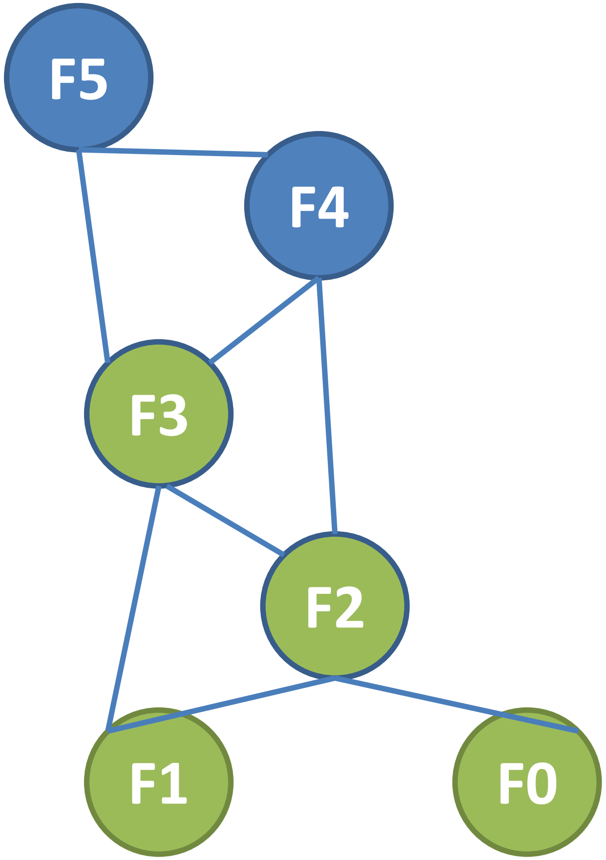
Qui hoặc động vs Chia để trị

- Quy hoạch động: Tiếp cận từ dưới lên (Bottom-up)



Qui hoặc động vs Chia để trị

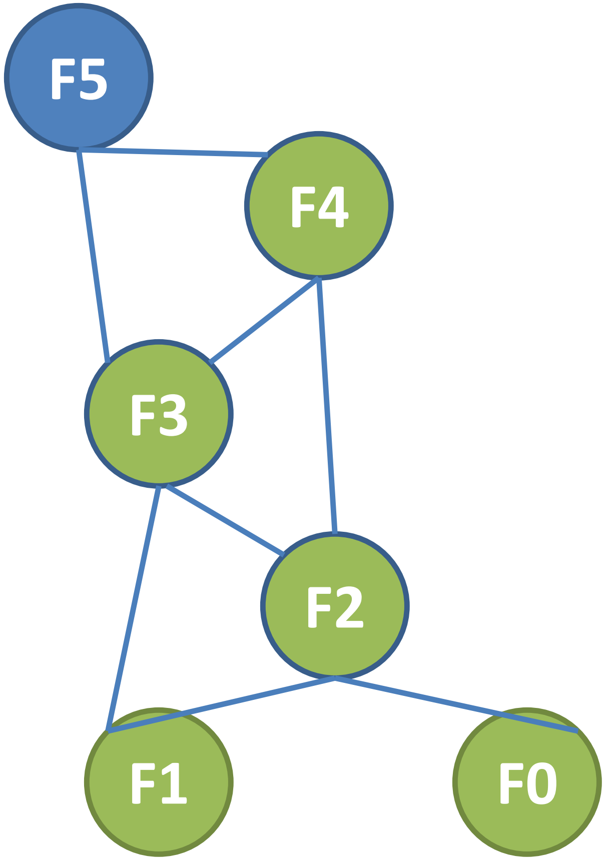
- Quy hoạch động: Tiếp cận từ dưới lên (Bottom-up)



Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

Qui hoặc động vs Chia để trị

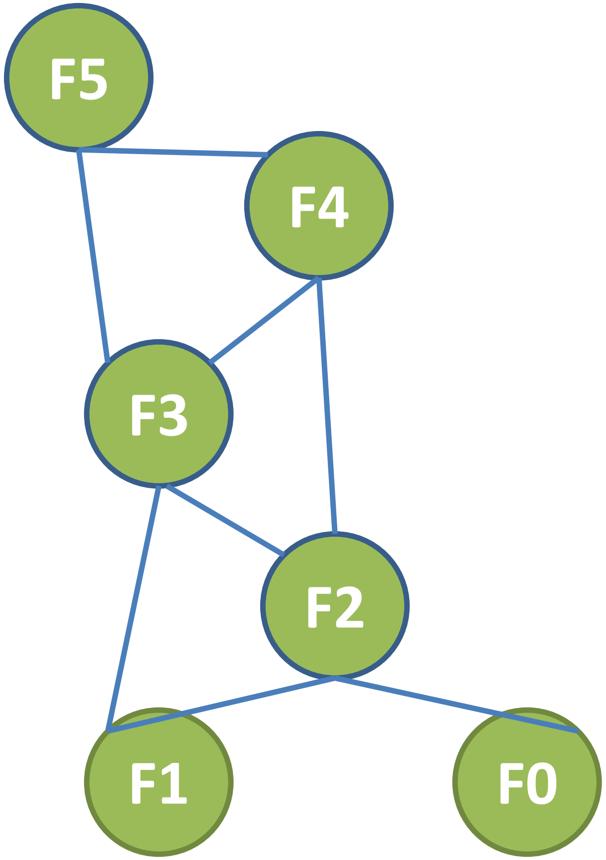
- Quy hoạch động: Tiếp cận từ dưới lên (Bottom-up)



Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Qui hoặc động vs Chia để trị

- Quy hoạch động: Tiếp cận từ dưới lên (Bottom-up)



Lược đồ chung

- **Phân rã**: Chia bài toán cần giải thành những bài toán con nhỏ hơn đến mức có thể giải trực tiếp được hay không?? -> Nếu được
- **Giải các bài toán con và ghi nhận lời giải**: Lưu trữ lời giải của các bài toán con vào một bảng để sử dụng về sau.
- **Tổng hợp lời giải**:
  – Tổng hợp lời giải các bài toán con kích thước nhỏ hơn thành lời giải bài toán lớn hơn.
  – Tiếp tục cho đến khi thu được lời giải của bài toán xuất phát (là bài toán con có kích thước lớn nhất)

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

Tính số Fibonaci bằng QHD

- Phân rã:

  F(n) = **F(n-1) + F(n-2)**

- Giải bài toán con

  F(0) = 0

  F(1) = 1

- Tổng hợp

  F(n) = F(n-1) + F(n-2)

Cài đặt

```
Function DPFib(n)
{
    F[0] = 0;  F[1] = 1;
    If (n>1)
    {
      For k = 2 to n { F[k] = F[k-1] + F[k-2];}
    }
    return F[n];
}
```

Minh họa

- Tính DPFib(5) **= 5**

```
Function DPFib(n)
{
    F[0] = 0;  F[1]=1;
    If (n>1)
    {
      For k = 2 to n
        {
            F[k] = F[k-1] + F[k-2];
         }
    }
    return F[n];
}
```

k=2: F(2)=F(1)+F(0)=1+0=1

K=3: F(3)=F(2)+F(1)=1+1=2

K=4: F(4)=F(3)+F(2)=2+1=3

K=5: F(5)=F(4)+F(3)=3+2=5

Cài đặt khác

```
Function DPFib2(n)
{
    Fk2 = 0;  Fk1 = 1; k=2
    While (k<=n)
    {       tg    =  Fk1;
            Fk1 =  Fk1 + Fk2;
            Fk2 = tg;     k =  k+1;
    }
    return Fk1;
}
```

Đánh giá

- Thuật toán 1 DPFib(n)
  - Bộ nhớ ??
  - Thời gian ??
- Thuật toán 2 DPFib2(n)
  - Bộ nhớ ??
  - Thời gian ??

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

(**Knapsack Problem**)

- Có **n** đồ vật, đồ vật i có trọng lượng $\mathbf{w_i}$ và giá trị $\mathbf{c_i}$, i = 1, 2, ..., n.
- Tìm cách chất các đồ vật này vào cái túi có dung lượng là **b** sao cho tổng trọng lượng của các đồ vật được chất vào túi là không quá b, đồng thời tổng giá trị của chúng là lớn nhất.

Bài toán

(**Knapsack Problem**)

- Có **n** đồ vật, đồ vật i có trọng lượng w_i và giá trị c_i, i = 1, 2, ..., n.
- Tìm cách chất các đồ vật này vào cái túi có dung lượng là **b**

PP Tham lam → **Kết quả nhận được thường là không tối ưu**

Giải bằng QHD ???

- Có: **n** - Số đồ vật, **b** - trọng lượng túi (nguyên)
- Phân rã: Với các giá trị **i** (**1..n**) và **L** (**0..b**) Gọi **MaxV(i,L)** là tổng giá trị lớn nhất có thể chọn trong **i** đồ vật (từ 1 đến i) với trọng lượng tối đa của túi là **L**. Khi đó **MaxV(n,b)** là giá trị lớn nhất mang đi được.
- Giải bài toán con: MaxV(0,L) = 0 với $\forall$L, và MaxV(i,0) = 0 với $\forall$i.

Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Giải bằng QHD ???

- Tổng hợp:
  - Đã có MaxV(i-1,L): Giá trị lớn nhất mang đi được với i-1 đồ vật khi trọng lượng túi là L.
  - Xét đồ vật thứ i khi trọng lượng túi vẫn là L:
    - Chỉ mang thêm đồ vật thứ i khi giá trị của túi lúc mang i-1 đồ vật ở trọng lượng túi là L-w[i] (như thế mới đảm bảo mang thêm được đồ vật i có trọng lượng w[i] khi trọng lượng túi là L) cộng với giá trị của đồ vật thứ i, c[i], lớn hơn khi không mang đồ vật thứ i, MaxV(i-1,L).
    - Nghĩa là

MaxV(i, L) = Max{MaxV(i-1,L-w[i])+c[i], MaxV(i-1,L)}

Cài đặt

```
Procedure Bag_best
{
    For  L= 0 to  b do  MaxV[0,L] =0 ;
    For  i= 0 to  n do  MaxV[i,0] =0 ;
    For i = 1 to n  do
      For L = 1 to b  do
        MaxV[i,L] = MaxV[ i-1,L];
        If  [(L ≥ w[i]) &&  (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]
             MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i] ;
     return MaxV(n, b) ;
}
```

Minh họa

- Cho 6 đồ vật (n = 6), và túi có trọng lượng b = 19. Các đồ vật có trọng lượng và giá trị như sau:

i	c	w
1	7	3
2	10	4
3	20	5
4	19	7
5	13	6
6	40	9

Khởi tạo

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0																			
10	4	2	0																			
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

Lặp ...

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0																			
10	4	2	0																			
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Lặp …

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0																		
10	4	2	0																			
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Lặp …

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0																	
10	4	2	0																			
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Lặp …

- n = 6, b = 19

		i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
c	**w**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7																
10	4	2	0																			
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Lặp …

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7															
10	4	2	0																			
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Lặp …

- n = 6, b = 19

		i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
c	**w**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0																			
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Lặp …

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0	0	0	7																
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Lặp …

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0	0	0	7	10															
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

Lặp ...

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0	0	0	7	10	10														
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

Lặp ...

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0	0	0	7	10	10	10													
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

Lặp …

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0	0	0	7	10	10	10	?												
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L $\geq$ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Lặp ...

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0	0	0	7	10	10	10	17												
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Lặp ...

- n = 6, b = 19

		i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
c	**w**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0	0	0	7	10	10	10	17	17	17	17	17	17	17	17	17	17	17	17	17
20	5	3	0																			
19	7	4	0																			
13	6	5	0																			
40	9	6	0																			

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Lặp ...

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0	0	0	7	10	10	10	17	17	17	17	17	17	17	17	17	17	17	17	17
20	5	3	0	0	0	7	10	20	20	20	27	30	30	30	37	37	37	37	37	37	37	37
19	7	4	0	0	0	7	10	20	20	20	27	30	30	30	39	39	39	46	49	49	49	56
13	6	5	0	0	0	7	10	20	20	20	27	30	30	33	39	39	40	46	49	49	52	56
40	9	6	0	0	0	7	10	20	20	20	27	40	40	40	40	50	60	60	60	67	70	70

MaxV[i,L] = MaxV[i-1,L];

If [(L ≥ w[i]) && (MaxV[i-1,L-w[i]]+c[i] > MaxV[i-1, L])]

MaxV[i, L] = MaxV[i-1,L-w[i]]+c[i];

Kết thúc

- n = 6, b = 19

c	w	i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0	0	0	7	10	**10**	10	17	17	17	17	17	17	17	17	17	17	17	17	17
20	5	3	0	0	0	7	10	20	20	20	27	30	**30**	30	37	37	37	37	37	37	37	37
19	7	4	0	0	0	7	10	20	20	20	27	30	30	30	39	39	39	46	49	49	49	56
13	6	5	0	0	0	7	10	20	20	20	27	30	30	33	39	39	40	46	49	49	52	56
40	9	6	0	0	0	7	10	20	20	20	27	40	40	40	40	50	60	60	60	67	70	**70**

Những vật được mang đi:

Tổng trọng lượng vật:

Tổng giá trị:

Kết thúc

- n = 6, b = 19

		i/L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
c	**w**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3	1	0	0	0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
10	4	2	0	0	0	7	10	**10**	10	17	17	17	17	17	17	17	17	17	17	17	17	17
20	5	3	0	0	0	7	10	20	20	20	27	30	**30**	30	37	37	37	37	37	37	37	37
19	7	4	0	0	0	7	10	20	20	20	27	30	30	30	39	39	39	46	49	49	49	56
13	6	5	0	0	0	7	10	20	20	20	27	30	30	33	39	39	40	46	49	49	52	56
40	9	6	0	0	0	7	10	20	20	20	27	40	40	40	40	50	60	60	60	67	70	**70**

Những vật được mang đi: {2, 3, 6}

Tổng trọng lượng vật: 18

Tổng giá trị: 70

Biên soạn: Hà Đại Dương,
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 mảng N số: A[1..N]
- Hãy tìm dãy con các phần tử liên tiếp của A có tổng lớn nhất.
- Ví dụ: 13, -15, 2, 18, 4, 8, 0, -5, -8

  Thì dãy con cần tìm là A(3)-A(6)

  13, -15, **2, 18, 4, 8**, 0, -5, -8

  (Đã giải quyết theo phương pháp chia để trị)

Tiếp cận qui hoặc động

- Phân rã:
  - Gọi MaxS[i] là tổng lớn nhất của dãy con liên tiếp có i phần tử a[1]..a[i].
  - Khi đó MaxS[N] là giá trị lớn nhất của dã con liên tiếp cần tìm
- Bài toán cơ sở:
  - Với i = 1 ta có MaxS[i] = a[i]

Tổng hợp

- Giả sử i > 1 và MaxS[k] là đã biết với k = 1,.., i-1. Ta cần tính MaxS[i] là tổng của dãy con liên tiếp lớn nhất của dãy a[1]..., a[i-1], a[i].
- Các dãy con liên tiếp của dãy này có thể là một trong hai trường hợp:
  - Các dãy con liên tiếp có chứa a[i]
  - Các dãy con liên tiếp không chứa a[i]

Tổng hợp …

- Gọi **MaxE[i]** là tổng lớn nhất của các dãy con liên tiếp của dãy a[1]..a[i] chứa chính a[i].
- Tổng lớn nhất của các dãy con liên tiếp của dãy a[1]..a[i] không chứa a[i] chính là tổng lớn nhất của các dãy con của dãy a[1]..a[i-1], nghĩa là MaxS[i-1].

⇒ **MaxS[i] = max{MaxS[i-1], MaxE[i]}**

Tính MaxE[i]

- Để tính MaxE[i], i = 1, 2, ..., n, ta cũng có thể sử dụng công thức đệ quy như sau:
  - Với i=1 thì MaxE[i] = a[1];
  - Với i >1, Gọi C là dãy con kế tiếp lớn nhất của dãy a[1]..a[i] có chứa a[i]. Có hai khả năng:
    - Nếu C chứa a[i-1] thì tổng lớn nhất là MaxE[i-1]+a[i];
    - Nếu C không chứa a[i-1] thì C chỉ gồm a[i] và tổng lớn nhất là a[i]

⇒ **MaxE[i] = max {a[i], MaxE[i-1]+a[i]}, i>1**

Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

Cài đặt

- s - chỉ số đầu
- e - chỉ số cuối
- s1 - chỉ số đầu tạm

Procedu subMax

```
{
   MaxS=a[1]; MaxE= a[1];
   s=1; e=1; s1=1;
   For i = 2 to n do
   {
      if MaxE>0 then MaxE=MaxE+a[i]
      else {MaxE = a[i];  s1=i; }
      if (MaxE > MaxS) then {
           MaxS = MaxE;
           e=i;  s=s1; }
   }
}
```

Minh họa

- Dãy a[1..9] = 13, -15, 2, 18, 4, 8, 0, -5, -8

i	a[i]	MaxE	s1	MaxS	s	e	Dãy con tổng lớn nhất
1	13	13	1	13	1	1	13,
2	-15	-2	1	13	1	1	13,
3	2	2	3	13	1	1	13,
4	18	20	3	20	3	4	2, 18,
5	4	24	3	24	3	5	2, 18, 4,
6	8	32	3	32	3	6	2, 18, 4, 8,
7	0	32	3	32	3	6	2, 18, 4, 8,
8	-5	27	3	32	3	6	2, 18, 4, 8,
9	-8	19	3	32	3	6	2, 18, 4, 8,

Bài tập

1. Thực hiện từng bước bài toán cái túi với dữ liệu:
   - Trọng lượng túi b=10
   - Số lượng đồ vật n=6
   - Các vật w{6 ,3 ,3 ,7 ,4 ,3}
     giá trị v{12,1,8 ,1 ,10 ,3}
2. Cho dãy A={-98,54,67, 65,-879,78,65,21,-6,67}, tìm dãy con dài nhất theo phương pháp qui hoạch động.

Bài tập

3. Cài đặt thuật toán giải bài toán cái túi theo phương pháp qui hoạch động. Đánh giá độ phức tạp bằng thực nghiệm và so sánh với lý thuyết.
4. Cài đặt thuật toán tìm dãy con lớn nhất theo phương pháp qui hoạch động. Đánh giá độ phức tạp bằng thực nghiệm và so sánh với lý thuyết.