Dynamic Programming

Soontharee Koompairojn

Optimization Problem

- *การเรียนรู้เพื่อกำหนดวิธีการที่ดีที่สุดให้กับปัญหา (การหาค่าที่เหมาะสมที่สุด)*
- *การหาค่าสูงสุดหรือค่าต่ำสุดของปัญหา*
- *แสดงปัญหาอยู่ในรูปของฟังก์ชันทางคณิตศาสตร์*

Algorithmic Paradigms

- Greedy. (Not optimal)
  - ❑ Build up a solution incrementally, คำตอบเพิ่มขึ้นเรื่อยๆ
  - ❑ myopically optimizing some local criterion.

- Divide-and-conquer. (Optimal) แบ่งครึ่งซ้ายขวา
  - ❑ Break up a problem into sub-problems,
  - ❑ solve each sub-problem independently, and
  - ❑ combine solution to sub-problems to form solution to original problem.

- Dynamic programming. (Optimal) เอาคำตอบก่อนหน้ามาใช้ด้วย
  - ❑ Break up a problem into a series of overlapping sub-problems,
  - ❑ And build up solutions to larger and larger sub-problems.

Ref: Algorithm design, Jon Kleinberg and Eva Tardos, slide by Kevin Wayne

Dynamic Programming Applications

- Areas.
  - Bioinformatics.
  - Control theory.
  - Information theory.
  - Operations research.
  - Computer science: theory, graphics, AI, compilers, systems, ….

- Some famous dynamic programming algorithms.
  - Unix diff for comparing two files.
  - Viterbi for hidden Markov models.
  - Smith-Waterman for genetic sequence alignment.
  - Bellman-Ford for shortest path routing in networks.
  - Cocke-Kasami-Younger for parsing context free grammars.

Compiler

Ref: Algorithm design, Jon Kleinberg and Eva Tardos, slide by Kevin Wayne

Fibonacci Numbers

- ให้เขียน algorithm คำนวณ n^{th} Fibonacci number F_n
- $F_0 = 0, F_1 = 1$
- For all $i \geq 2$, $F_i = F_{i-1} + F_{i-2}$

recursive Algo

```
public int fibo (int n){
    if(n=0)
        return 0;
    else if (n=1)
        return 1;
    else
        return fibo(n-1) + fibo(n-2);
}
```

Fibonacci Numbers

- Finding the nth Fibonacci number F_n, where,
- $F_0 = 0$, $F_1 = 1$
- For all $i \geq 2$, $F_i = F_{i-1} + F_{i-2}$

- <u>Recursive Algorithm</u>

$T(n) = T(n-1) + T(n-2)$ for $n \geq 2$

with $T(0) = c_1$

$T(1) = c_2$

- Run time

$T(n) \geq \phi^{n-2}$ for $n \geq 3$, $\phi = (1+\sqrt{5})/2$ ~

Fibonacci Numbers

Finding the nth Fibonacci number F_n, where,

$F_0 = 0$, $F_1 = 1$

For all $i \geq 2$, $F_i = F_{i-1} + F_{i-2}$

<u>Dynamic Programming Algorithm</u> จะใช้ array เข้ามาช่วยในการเก็บค่า

```
int[] arr = new int[n]
arr[0] = 0
arr[1] = 1
for(i=2; i≤n ; i++) {
   arr[i] = arr[i-1] + arr[i-2]
}
return arr[n]
```

Run time

Dynamic Programming

แนวทางการแก้ปัญหา

- กำหนดโครงสร้างการแก้ปัญหา optimal solution (ปัญหาที่ หาค่ามากที่สุด หรือ หาค่าที่น้อยที่สุด)
- คิดแก้ปัญหา optimal solution แบบ recursive
- คิดแก้ปัญหา optimal solution แบบ bottom-up
- คำนวณค่า optimal solution จากคำตอบที่ได้คำนวณมาก่อนหน้านี้

แบบ Fibo array

ปัญหา Maximum Subsequence Sum

array

กำหนดให้ S เป็นลำดับ (sequence) ตัวเลข n ตัว $<x_1, x_2, \ldots, x_n>$,

โดยอาจจะมีจำนวนลบอยู่ด้วย

ปัญหา คำนวณหา**ผลบวก**ของลำดับย่อย subsequences ของ S ที่มีค่ามากที่สุด

กำหนดให้ s' = subsequence ที่ต่อเนื่องของ S

$= x_i, x_{i+1}, \ldots, x_{i+k}$, โดยที่ $1 \le i \le n$ และ $k \ge 0$

สมมติ $S = \langle 1, -5, 2, -1, 3 \rangle$,

จับกลุ่มยังไงก็ได้ไม่เรียงลำดับกัน

Subsequence ≠ Subset

ตัวอย่างของ S' ได้แก่

$\langle 1 \rangle$, $\langle 2, -1, 3 \rangle$, และ $\langle -5, 2 \rangle$

Boost Force

1, -5, 2, -1, 3

ทำไปเรื่อยๆ แล้ว[illegible]ได้ค่ามากที่สุด

MSS ของ sequence S เป็นเท่าไร?

Maximum subsequence sum problem(2)

<1, -5, 2, -1, 3>

Ending at

1

-5 : -5
1, -5 : -4

2 : 2
-5, 2 : -3
1, -5, 2 : -2

Developing a dynamic programming algorithm

- กำหนดโครงสร้างของ optimal solution

  แบ่งปัญหาออกเป็นปัญหาย่อย

แนวคิด : เก็บผลรวม(Sum)ที่มากที่สุดของsubsequenceใดๆ สิ้นสุดที่ตัวเลข x_i, โดยที่ $1 \leq i \leq n$

กำหนดให้ B เป็น array 1 มิติขนาด n ตัว

B[i] เก็บค่า ผลบวกที่มากที่สุดของ subsequence ที่สิ้นสุดที่ตัวเลข x_i

เมื่อคำนวณค่าในตาราง B จะได้ค่าผลบวกที่มากที่สุดเอง

x_1

$S = \langle 1, -5, 2, -1, 3 \rangle$

B[i]	1	2	3	4	5
	1	-4	2	1	4

1 -5 2 -1 1

1 -4 2 1 2

Pseudocode

$B[1] = x_1$ 1

For i = 1 to n do

$$B[i+1] = \begin{cases} B[i] + x_{i+1} & \text{if } B[i] > 0 \\ x_{i+1} & \text{if } B[i] \le 0 \end{cases}$$

Subset sum problem

กำหนดให้ เซต $S = \{a_1, a_2, ..., a_n\}$ เป็นเซตของเลขจำนวนเต็มบวก
และค่าผลรวมเป็นเลขจำนวนเต็ม B

คำถาม: มีเซตย่อย (subset) S' ของ S ที่ผลบวกของสมาชิกใน S'
มีค่าเท่ากับ B หรือไม่

ตัวอย่างเช่น กำหนดให้ เซต $S = \{3, 34, 4, 12, 5, 2\}$,
และค่าผลรวมเท่ากับ 11

คำถาม: มีเซตย่อย (subset) S' ของ S ที่ผลบวกของสมาชิกใน S'
มีค่าเท่ากับ 11 หรือไม่

มี หรือ ไม่มี , True or False ?

ตัวอย่าง

เซต S = <2,3>, n = 2

ผลรวม B = 5

Is there a subset (มี Subsetหรือไม่)?

True or False?

	ผลรวม j					
	0	1	2	3	4	5
a_1=2			T			
a_2=3				T		

If a_i มีค่าผลรวมเท่ากับ j then ให้เติม T ในช่อง

Else เติม F ในช่อง

ตัวอย่าง

เซต S = <2,3>, n = 2

ผลรวม B = 5

Is there a subset (มี Subsetหรือไม่)?

True or False?

ถ้าตัวก่อนเป็น T ตัวต่อมาจะเป็น T ด้วย

	ผลรวม j					
	0	1	2	3	4	5
a_1=2	T	F	T	F	F	F
a_2=3	T	F	T	T	F	T

i

If a_i มีค่าผลรวมเท่ากับ j then ให้เติม T ในช่อง

Else เติม F ในช่อง

ตัวอย่าง

เซต S = <2,3>, n = 2

ผลรวม B = 5

Is there a subset (มี Subsetหรือไม่)?

True or False?

T(1,2) T(1,5)

	ผลรวม j					
	0	1	2	3	4	5
a_1=2	T		T			
a_2=3	T		T	T		T

a_i

T(2,5)

ตัวอย่าง

เซต S = <2,3>, n = 2

ผลรวม B = 5

Is there a subset (มี Subsetหรือไม่)?

True or False?

$T(1,2)=T(n-1,j-a_n)$

$T(1,5)=T(n-1,j)$

	ผลรวม j					
	0	1	2	3	4	5
$a_1=2$						
$a_2=3$						

$T(2,5)=T(n,j)$

ตัวอย่าง

เซต S = <2,3,5>, n = 3

ผลรวม B = 5

Is there a subset (มี Subsetหรือไม่)?

True or False?

	ผลรวม j					
	0	1	2	3	4	5
a_1=2						
a_2=3						
a_3=5						

if ตัวเลขตัวใดตัวหนึ่งในเซต S มีค่าเท่ากับ B → T(n,B) เป็นจริง

แนวคิดในการแก้ปัญหา

- ใช้แนวทางพิสูจน์ **induction** ช่วยแก้ปัญหา
- ให้ T(n,B) เป็นประพจน์ที่ใช้ในการพิสูจน์ว่า
  "มีเซตย่อย S' จำนวน n ตัวที่มีผลบวก= B หรือไม่"
- Base case:

  if ตัวเลขตัวใดตัวหนึ่งในเซต S มีค่าเท่ากับ B ➔ T(n,B) เป็นจริง ---①
- Induction step:
  - if T(n-1, B) เป็น**จริง**, ➔ ไม่ต้องใช้ a_n ---②
  - if T(n-1, B) เป็น**เท็จ**, ➔ อาจจะมี a_n

    ➔ ต้องดูว่า T(n-1, B-a_n) เป็นจริงหรือไม่---③

ดังนั้นปัญหา T(n,B) เราพิจารณาปัญหาย่อย T(n-1,B) และ T(n-1, B- a_n)

แนวคิดในการแก้ปัญหา

การจะแก้ปัญหา subset sum ที่มีผลรวม = B

เราต้องแก้ปัญหา subset sum ที่มี $\leq$ B ด้วย

กำหนดให้ เซต $S = \{a_1, a_2, \ldots, a_i \ldots, a_n\}$ และค่าผลรวม = B

ให้ เซต S' เป็นเซตตัวเลข i ตัว = $\{a_1, a_2, \ldots, a_i\}$

และค่าผลรวม = j โดยที่ $0 \leq j \leq B$

- ถ้า $j = 0$ นั่นคือ ไม่มีตัวเลขตัวไหนอยู่ในเซตคำตอบ
- ถ้า $a_i = j$ นั่นคือ a_i อยู่ในเซตคำตอบ ---① (Base case)
- ถ้า เซตตัวเลข i-1 ตัว = $\{a_1, a_2, \ldots, a_{i-1}\}$ มีค่าผลรวม = j

  ➔ เซตตัวเลข i ตัว = $\{a_1, a_2, \ldots, a_{i-1}, a_i\}$ มีค่าผลรวม = j ด้วย ---②
- ถ้า เซตตัวเลข i-1 ตัว = $\{a_1, a_2, \ldots, a_{i-1}\}$ มีค่าผลรวม = $j - a_i$

  ➔ เซตตัวเลข i ตัว = $\{a_1, a_2, \ldots, a_{i-1}, a_i\}$ จะมีค่าผลรวม = j ---③

ดังนั้นปัญหา T(i,j) แบ่งออกเป็น

ปัญหาย่อย T(i-1, j) และ T(i-1, j- a_i)

<u>แนวคิด</u> : เก็บคำตอบของปัญหาย่อยไว้ใน 2D Boolean array T,

ที่มีขนาด n แถว และ B+1 คอลัมน์.

ถ้าเซตย่อย (subset) $\{a_1, a_2, ..., a_i\}$ มีผลรวมเท่ากับ j

ให้ค่า T[i,j] = true

มิฉะนั้น ให้ค่า T[i,j] = false

Algorithm สำหรับ subset sum problem

สำหรับแถวแรก, (i=1)

```
        for  0 ≤ j ≤ B
           if   (j=0) or (a_1 = j)
                    T[1,j]  = true
           else  T[1,j]  = false
```

สำหรับแถวที่ i, โดยที่ $2 \le i \le n$

```
        for  0 ≤ j ≤ B
           if ((T[i-1,j]==true) or (T[i-1,j- a_i ] ==true))
                        T[i,j]    = true
           else         T[i,j]    = false
```

ตัวอย่าง

เซต S = <2,3,5,6>

ผลรวม B = 16

Is there a subset (มี Subsetหรือไม่)?

True or False?

T(2,

ดูบรรทัดข้างบน ↑

```
for  0 ≤ j ≤ B
   if ((T[i-1,j]==true) or (T[i-1,j- a_i ] ==true))
                T[i,j]    = true
   else         T[i,j]    = false
```

start

start here

	ผลรวม B																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$a_1=2$	T	F	T	F	F	F	F	F									
$a_2=3$	T		T	T		T											
$a_3=5$	T		T	T		T		T	T		T						
$a_4=6$	T		T	T		T	T	T	T	T	T		T	T		T	