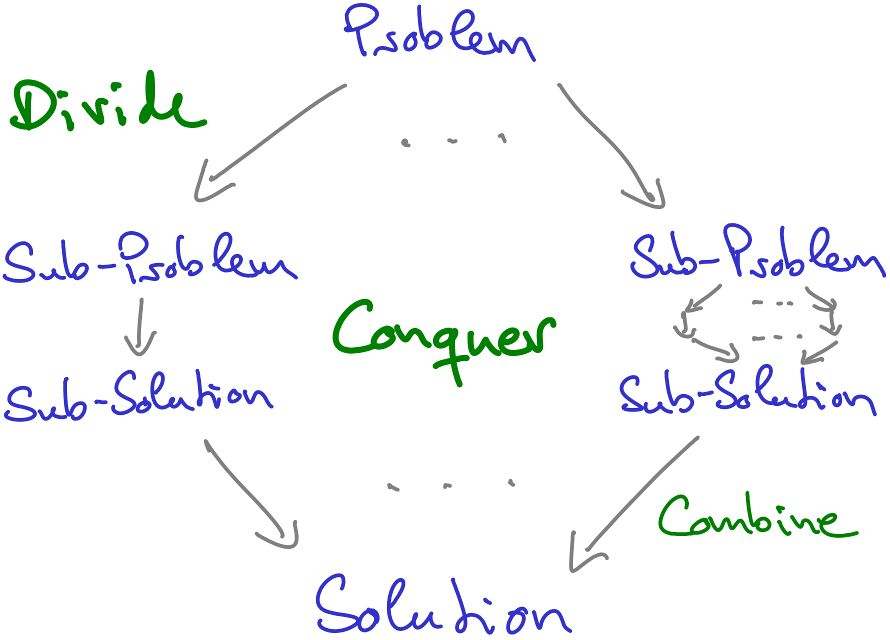
GTA*

An Algebraic Method for Developing Divide and Conquer Algorithms

Sebastian Fischer

(with Kento Emoto and Zhenjiang Hu)

*Generate, Test, and Aggregate


Problem
Divide
Sub-Problem
Sub-Problem
Conquer
Sub-Solution
Sub-Solution
Combine
Solution

Sorting

[3, 1, 5, 2, 6, 4, 7]

Split

[3, 1, 5]

[2, 6, 4, 7]

Sort

[1, 3, 5]

[2, 4, 6, 7]

Merge

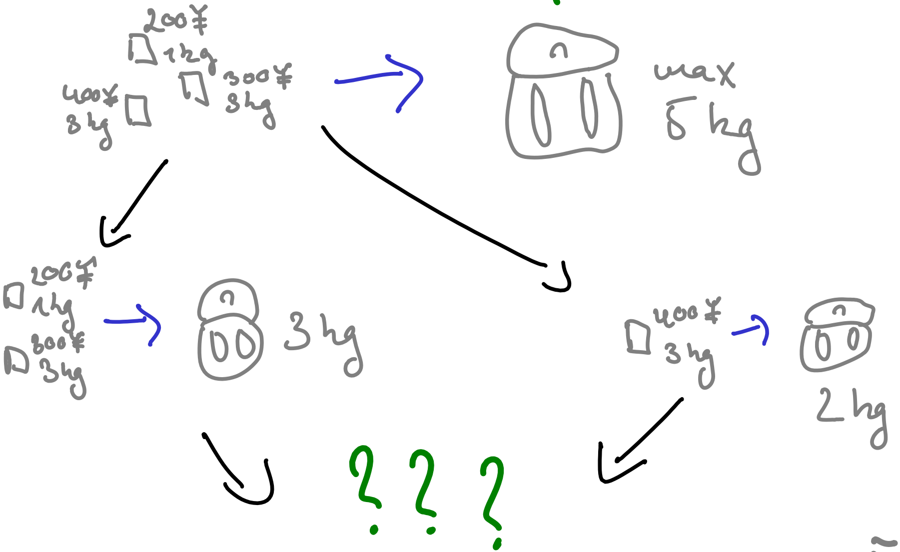
[1, 2, 3, 4, 5, 6, 7]

Knapsack Problem

- fill knapsack with items
- each has value and weight
- Problem:

  maximize total value
  without exceeding
  weight restriction

Divide and Conquer ?



Divide and Conquer

- important design pattern
- useful for parallel programming (→ MapReduce)
- can be (and often is!) tricky in practice

GTA

- hides "trickyness"
- general method applicable to wide class of problems
- automatic parallelization

G: simple divide and conquer algorithm to generate intermediate results

T: simple divide and conquer algorithm to test each intermediate result

A: "kind of" divide and conquer algorithm to aggregate remaining results

GTA

- makes complex divide and conquer algorithm from simpler ones
- specification as a search problem
- possibly many intermediate results
- turns inefficient search into efficient parallel program

Knapsack Problem

G: generate (multi set) of all possible combinations of items easy ✓

T: discard all combinations with a too big total weight easy ✓

A: among remaining combinations select one with maximum value easy ✓

Knapsack Problem

Intuition:

too many intermediate results
(2^n, if n is number of items)

Nevertheless:

efficient algorithm can be obtained automatically

How ???

Using

Math !!!

Some Algebra

- Monoids, Lists
- Semirings, Multisets of Lists
- Homomorphisms

Monoids

set M with associative binary operation $\otimes$ and identity element $id_\otimes$

example: lists with concatenation

$$(x ++ y) ++ z = x ++ (y ++ z)$$

$$[\,] ++ x = x = x ++ [\,]$$

List Homomorphism

$$h\ [\,] = id_{\otimes}$$

some function into monoid M

$$h\ [x] = f\ x$$

$$h\ (x +\!\!+ y) = h\ x \otimes h\ y$$

$\longrightarrow$ divide and conquer on lists

Semirings

same M

two monoids $(M, \otimes)$, $(M, \oplus)$

commutative

with distributivity:

$$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$$

$$(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$$

and zero laws:

$$id_{\oplus} \otimes x = id_{\oplus}$$

$$x \otimes id_{\oplus} = id_{\oplus}$$

Example

$\otimes$ $\oplus$

$$(\mathbb{Z} \cup \{-\infty\}, +, \max)$$

$$id_+ = 0$$

$$id_{\max} = -\infty$$

$$x + \max\ y\ z = \max\ (x+y)\ (x+z)$$

$$-\infty + x = -\infty$$

$\vdots$

Another Example

$$(\{[a]\}, X_{++}, \uplus)$$

multisets (bags) of lists

bag union

$$a \; X_{++} \; b = \{ x{+\!\!+}y \mid x \in a, y \in b \}$$

Bags of Lists

$$(\{[1]\} \uplus \{[2]\}) \times_{++} \{[3]\} =$$

$$\{[1], [2]\} \times_{++} \{[3]\} =$$

$$\{[1] ++ [3], [2] ++ [3]\} =$$

$$\{[1, 3], [2, 3]\}$$

Semiring Homomorphisms

$$h\,\{\!\{\}\!\} = id_{\oplus}$$

$$h\,\{\!\{[\,]\}\!\} = id_{\otimes}$$

$$h\,\{\!\{[x]\}\!\} = f\ x$$

some function into semiring

$$h\,(a \uplus b) = h\ a \oplus h\ b$$

$$h\,(a \times_{++} b) = h\ a \otimes h\ b$$

if

G : list homomorphism generate

T : (almost) list homomorphism test

A : semiring homomorphism aggregate

then

aggregate . filter test . generate

↓

(efficient) divide & conquer alg.

Ok, but how???

Secret!

(but see paper)

Knapsack Generator

$id_{\times_{++}}$

$$\text{sublists } [\,] = \{[\,]\}$$
$$\text{sublists } [item] = \{[\,]\} \uplus \{[item]\}$$
$$\text{sublists } (a ++ b) = \text{sublists } a \times_{++} \text{sublists } b$$

→ list homomorphism

Knapsack Test

weight [] = 0
weight [(v, w)] = w
weight (a ++ b) = weight a + weight b

test items = weight items ≤ W

maximum weight

almost

→ list homomorphism

Knapsack Aggregator

(not essential) simplification :

compute maximum possible value
rather than corresponding items

$$\begin{aligned}
\mathit{maxval}\ \{\} &= -\infty \\
\mathit{maxval}\ \{[\,]\} &= 0 \\
\mathit{maxval}\ \{[(v,w)]\} &= v \\
\mathit{maxval}\ (a \uplus b) &= \max\ (\mathit{maxval}\ a)\ (\mathit{maxval}\ b) \\
\mathit{maxval}\ (a \times_{+\!+} b) &= \mathit{maxval}\ a + \mathit{maxval}\ b
\end{aligned}$$

→ semiring homomorphism

Efficient Knapsack Algorithm

[(200¥, 1kg), (300¥, 3kg), (400¥, 3kg)]

Divide

3kg : 300¥

3kg : 400¥

1kg : 200¥

Combine

Combine

3kg : 400¥

1kg : 200¥
3kg : 400¥
4kg : 600¥ → max value : 600¥

Associativity

[(200¥, 1hg), (300¥, 3hg), (400¥, 3hg)]

Divide

1hg : 200¥

3hg : 300¥

3hg : 400¥

Combine

1hg : 200¥
3hg : 300¥
4hg : 500¥

Combine

1hg : 200¥
3hg : 400¥
4hg : 600¥

Knapsack Complexity

- linear in number of items
- quadratic in maximum weight
  (→ pseudo polynomial)
- one processor : $O(n\,w^2)$
- p processors : $O\left(\left(\log p + \frac{n}{p}\right) w^2\right)$

GTA

efficient divide and conquer algorithm
from intuitive specification if

G : list homomorphism
T : (almost) list homomorphism
A : semiring homomorphism

knapsack example generalizes
to other applications, complexity too

GTA

many predefined generators:

sublists, prefixes, suffixes, segments, ...

practical applications:

- inferring states of hidden Markov model
- incremental refinement via filters

generalizes to:

- other input types easy
- other intermediate types possible too

Secrets Revealed

Emoto, F., Hu

Generate, Test, and Aggregate — A Calculation-based Framework for Systematic Parallel Programming with MapReduce

ESOP 2012

ありがとう!