Lecture 1 Introduction and Peak Finding 6.006 Fall 2011 Lecture 1: Introduction and Peak Finding

Lecture Overview

*•* Administrivia

*•* Course Overview

*•* “Peak finding” problem — 1D and 2D versions

Course Overview

This course covers:

*•* Efficient procedures for solving problems on large inputs (Ex: U.S. Highway Map, Human Genome)

*•* Scalability

*•* Classic data structures and elementary algorithms (CLRS text)

*•* Real implementations in Python

*•* Fun problem sets!

The course is divided into 8 modules — each of which has a motivating problem and problem set(s) (except for the last module). Tentative module topics and motivating problems are as described below:

1. Algorithmic Thinking: Peak Finding

2. Sorting & Trees: Event Simulation

3. Hashing: Genome Comparison

4. Numerics: RSA Encryption

5. Graphs: Rubik’s Cube

6. Shortest Paths: Caltech *→* MIT

7. Dynamic Programming: Image Compression

8. Advanced Topics

1

Lecture 1 Introduction and Peak Finding 6.006 Fall 2011

Peak Finder

One-dimensional Version

Position 2 is a peak if and only if *b ≥ a* and *b ≥ c*. Position 9 is a peak if *i ≥ h*.

1 2 3 4 5 6 7 8 9

a c b d e f g h i

Figure 1: a-i are numbers

Problem: Find a peak if it exists (Does it always exist?)

Straightforward Algorithm

6 7 4 3 2 1 4 5

Start from left

. . .

1 2 . . . n/2 n-1 n

might be peak

θ(n) complexity worst case

Figure 2: Look at *n/*2 elements on average, could look at *n* elements in the worst case

What if we start in the middle? For the configuration below, we would look at *n/*2 elements. Would we have to ever look at more than *n/*2 elements if we start in the middle, and choose a direction based on which neighboring element is larger that the middle element?

n/2

2

Lecture 1 Introduction and Peak Finding 6.006 Fall 2011

Can we do better?

1 2 . . . n/2 n-1 n

n/2 -1 n/2 +1. . .

look at n/2 position

Figure 3: Divide & Conquer

*•* If *a*[*n/*2] *< a*[*n/*2 *−* 1] then only look at left half 1 *. . . n/*2 *− − −* 1 to look for peak *•* Else if *a*[*n/*2] *< a*[*n/*2 + 1] then only look at right half *n/*2 + 1 *. . . n* to look for peak *•* Else *n/*2 position is a peak: WHY?

*a*[*n/*2] *≥ a*[*n/*2 *−* 1]

*a*[*n/*2] *≥ a*[*n/*2 + 1]

What is the complexity?

*T*(*n*) = *T*(*n/*2) + Θ(1) = Θ(1) + *. . .* + Θ(1) (log2(*n*) *times*) = Θ(log2(*n*)) ' -v "

to compare a[n*/*2] to neighbors

In order to sum up the Θ(*i*)‘s as we do here, we need to find a constant that works for all. If *n* = 1000000*,* Θ(*n*) algo needs 13 sec in python. If algo is Θ(log *n*) we only need 0*.*001 sec. Argue that the algorithm is correct.

Two-dimensional Version

c

n rows

b a e

d

m columns

Figure 4: Greedy Ascent Algorithm: Θ(*nm*) complexity, Θ(*n*2) algorithm if *m* = *n a* is a 2D-peak iff *a ≥ b, a ≥ d, a ≥ c, a ≥ e*

3

Lecture 1 Introduction and Peak Finding 6.006 Fall 2011 14 13 12

15 16

9 11 17 17 19 20

Figure 5: Circled value is peak.

Attempt # 1: Extend 1D Divide and Conquer to 2D

i

j = m/2

*•* Pick middle column *j* = *m/*2.

*•* Find a 1D-peak at *i, j*.

*•* Use (*i, j*) as a start point on row *i* to find 1D-peak on row *i*.

Attempt #1 fails

Problem: 2D-peak may not exist on row *i*

10

14 13 12

15

16

End up with 14 which is not a 2D-peak.

9 11

17 19 20 4

Lecture 1 Introduction and Peak Finding 6.006 Fall 2011

Attempt # 2

*•* Pick middle column *j* = *m/*2

*•* Find global maximum on column *j* at (*i, j*)

*•* Compare (*i, j −* 1)*,*(*i, j*)*,*(*i, j* + 1)

*•* Pick left columns of (*i, j −* 1) *>* (*i, j*)

*•* Similarly for right

*•* (*i, j*) is a 2D-peak if neither condition holds *←* WHY?

*•* Solve the new problem with half the number of columns.

*•* When you have a single column, find global maximum and you‘re done. Example of Attempt #2

go with

10 8 10 10

10 10

10

14 13 12

11

12

11

11

15 16

9 11 21 17 19 20

11 21 19 20

21  nd 21 20

pick this column

17 global max

for this column

Complexity of Attempt #2

pick this column 19 global max for this column

If *T*(*n, m*) denotes work required to solve problem with *n* rows and *m* columns

*T*(*n, m*) = *T*(*n, m/*2) + Θ(*n*) (to find global maximum on a column — (n rows)) *T*(*n, m*) = Θ(*n*) + *. . .* + Θ(*n*) ' -v "

log *m*

= Θ(*n* log *m*) = Θ(*n* log *n*) if m = n

Question: What if we replaced global maximum with 1D-peak in Attempt #2? Would that work?

5

MIT OpenCourseWare

http://ocw.mit.edu

6.006 Introduction to Algorithms

Fall 2011

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.