CS 420: Advanced Algorithm Design and Analysis
Spring 2015 – Lecture 2

Department of Computer Science
University of British Columbia

January 8, 2015

Announcements

General

- Office and TA consultation hours in effect as of today

Assignments...

- Asst1 out ... due January 15
- read over assignments *early*...well before you plan to work on them

Announcements

Readings...

- check the CS 420 homepage:
  http://people.cs.ubc.ca/~kirk/cs420/
- go carefully through the General Information Handout
  - policy on assignments: late policy, 25% rules, collaboration, acknowledgements
- review CS320 notes, particularly material on binary search and basic data structures: know where you can find what you may need to revisit
- read the essay:
  The Algorithm: Idiom of Modern Science, by Bernard Chazelle
  http://www.cs.princeton.edu/~chazelle/pubs/algorithm.html
  for thoughtful (and amusing) motivation for studying algorithms.

Last class...

Administration

- quick overview of course
- highlights of General Information Handout

Started a case study (reviewing basic issues & previewing others)

- finding extrema of a set of n elements (and related problems)
  - find the maximum
    - (several) algorithms using $n - 1$ comparisons; iterative, recursive, tournament
    - a *lower bound* of $n - 1$ comparisons: need to identify $n - 1$ non-maximums
  - find the minimum (by reduction to maximum)
  - other problems reducible to max-finding
    - find both the maximum *and* the minimum $\lceil 3n/2 \rceil - 2$
    - find the largest *and* second largest $n + \lceil \lg n \rceil - 2$
    - find the first, second and third largest ???
    - find the median worst case is between $2n$ and $3n$; expected case (using a randomized algorithm) is at most $1.5n$

Important issues/ideas...

Started a case study (reviewing basic issues & previewing others)

- algorithm design
  - same algorithm has different expressions: iterative, recursive...
  - exploit real-world solutions for algorithmic ideas: tournaments
  - exploit non-trivial data structures: heaps
  - reductions...algorithm re-use
  - randomization
- algorithm analysis
  - worst-case, average-case, expected-case analysis
  - lower bounds...intrinsic cost of underlying problem; optimality

Today...

Continue case study on finding extrema (reviewing basic issues & previewing others)

- taking the cost of other operations/resources into account
  - *auxiliary space* in finding the max and second largest; *streaming algorithms*; *time-space tradeoffs*
  - *update costs* in finding the maximum (the iterative and on-line hiring problems); *randomized algorithms*
- finding extrema in other computation models
  - parallel algorithms
  - distributed algorithms; communication complexity
- finding extrema in more restricted or more general input domains
  - inputs are drawn from $\mathcal{U} = \{0, 1, \ldots m - 1\}$
  - inputs are specified *implicitly*; *linear programming*
  - inputs are points in two (or higher) dimensions; *computational geometry*

Next time...

building and searching *dictionaries*

Continue case study...

taking the cost of other operations/resources into account

- *auxiliary space* in finding the max and second largest; *streaming algorithms*; *time-space tradeoffs*
- *update costs* in finding the maximum (the iterative hiring problems); *randomized algorithms*

Hiring Problem

- *update costs* in finding the maximum (the hiring problem)

  Claim. The *average* (over all input permutations) of the number of max-updates in the incremental max-finding algorithm is $\Theta(\log n)$.
  Proof. For a *random* input permutation, the probability that the the i-th input leads to a max-update (new hire) is $1/i$. So the expected number of updates is $\sum_{i=1}^{n} 1/i$ which is $\ln n + O(1)$.

  Corollary. The *expected* number of updates in the *randomized* incremental max-finding algorithm is $\Theta(\log n)$.

Continue case study...

finding extrema in other computation models

- parallel algorithms
- distributed algorithms; communication complexity

Maximum finding / Leader Election on a ring of processors...

How many messages are needed?

- $O(n^2)$ using naive algorithm
- $\Theta(n \lg n)$ are sufficient (and necessary)
  - Idea: candidate elimination
  - each round eliminates all but local maxima
  - each round eliminates half of the remaining candidates

Continue case study...

finding extrema in more general input domains

- inputs are specified *implicitly*; *linear programming*
- inputs are points in two (or higher) dimensions; *computational geometry*

Convex hull computation..

reduction to sorting...Graham's algorithm

- sort points by x-coordinate
- build upper and lower hulls incrementally ($O(n)$)

Continue case study...

finding extrema in more restricted input domains

- inputs are drawn from the restricted universe $\mathcal{U} = \{0, 1, \ldots m - 1\}$

Finding the maximum with inputs drawn from $\mathcal{U} = \{0, 1, \ldots m - 1\}$

Claim. The maximum input can be found using only *unary predicate* evaluations (eg. $x_7 \geq 13$?).
Proof. n-fold binary search suffices to determine the exact value of all n inputs with $O(n \lg m)$ unary predicate evaluations.

Remark 1. It is easy to argue that $\Theta(n + \lg m)$ unary predicate evaluations are *required*, and in fact it can be shown that $O(n + \lg m)$ also suffice.

Remark 2. This has an interesting interpretation as a problem in distributed computing (communication complexity)