Nature Inspired Search and Optimisation
Advanced Aspects of Nature Inspired Search and Optimisation
Randomised algorithms

Shan He

School for Computational Science
University of Birmingham

January 16, 2020

Outline of Topics

1. Motivating problems
2. Randomised algorithms
3. Let's solve the problems
4. Conclusion
5. Reading list

Problem 1: Matching one Bolt to n Nuts

- The everyday problem: given one bolt and a collection of n nuts of different sizes, find a nut match the bolt
- In mathematical form: given an array of n elements, find the first element of which the value equals to x
- Q: How to solve it using an algorithm? How to solve it efficiently?

Search $O(N)$

Problem 2: the real Nuts and Bolts Problem

$n \leftrightarrow n$

- A disorganized carpenter has a collection of n nuts of distinct sizes and n bolts and would like to find the corresponding pairs of bolts and nuts. Each nut matches exactly one bolt (and vice versa), and he can only compare nuts to bolts, i.e., he can neither compare nuts to nuts nor bolts to bolts.
- Can you help the carpenter match the nuts and bolts quickly?
- This is called Nuts and Bolts Problem or Lock and Key problem.
- Q: How to formulate the problem? How to solve it using an algorithm? How to solve it efficiently?

n^2

Problem 3: How can you help Dan Brown?

- Dan Brown is having holiday on Mars. He discovers a digital manuscript (a long string) of The Da Vinci Code in his laptop. He wants to compare this manuscript with his final version in his PC (also a long string) at home on Earth to see whether they are the same. He can access his manuscript in his PC at home, but the communication is very expensive.
- Q: How to compare these two versions via this expensive communication channel?

The one solution to the above problems

Randomised algorithms

- "*For many problems, a randomised algorithm is the simplest, the fastest or both.*" - Prabhakar Raghavan, Vice President of Engineering at Google.

Categories of algorithms by design paradigm

- Divide and conquer algorithms, e.g., quicksort algorithm
- Dynamic programming algorithms
- Mathematical programming algorithms, e.g., linear programming
- Search and enumeration algorithms
  - Brute force algorithms, enumerating all possible candidate solutions and check
  - Improved brute force algorithms, e.g., branch and bound algorithms
  - **Heuristic algorithms**
    - Local search, e.g., greedy search
    - **Randomised algorithms**, which include Evolutionary Computation, etc.

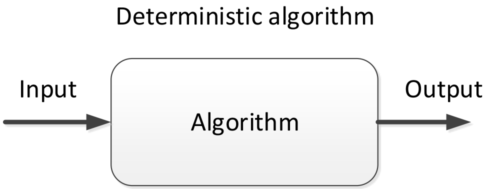
Heuristic algorithms

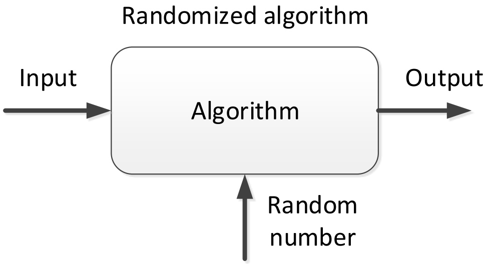
- CS Definition of **heuristic**: a (usually simple) algorithm that produces a **good enough** solution for a problem in a **reasonable time frame**
  near optimal
  - Heuristic: “”’find” or ‘discover” trade - off
  - Solutions are (Usually) non-optimal but satisfactory:
    - **Faster**: alternative to brute force (exhaustive) search
    - Trade off optimality, completeness, accuracy, or precision for **speed**.
  - Usually to solve problems that are difficult for other algorithms, e.g., brute force algorithms
  - Includes deterministic, e.g., local search and **randomised algorithms**

Randomised algorithms

- Randomised algorithm: An heuristic algorithm that **makes random choices** during execution to produce a result
  - Takes a source of random numbers to make random choices
  - Behaviour, e.g., output or running time can vary even on a fixed input
- The goal: design an algorithm and analyse it to show that its behaviour is likely to be good, on every input

Randomised algorithms vs deterministic algorithms





Randomised algorithms

- Two categories of randomised algorithms:
  - **Using random numbers to find a solution to a problem**
  - Using random numbers to improve a solution to a problem
- For the first category, two representative simple algorithms:
  - Las Vegas algorithm
  - Monte Carlo algorithm
- Let's see how they can solve problem 1.

random numbers $\Rightarrow$ result

Solving Problem 1 using Randomise algorithms

- The problem: given an array of n elements, find the first element of which the value equals to x
- Solution 1:

Las Vegas algorithm:

```
begin
    repeat
        Randomly select one element $a$ out of $n$ elements.
    until $a == x$
end
```

Solution to Problem 1

- The problem: given a array of n elements, find the first element of which the value equals to x
- Solution 2:

Monte Carlo algorithm:

```
begin
    $i := 0$
    repeat
        Randomly select one element $a$ out of $n$ elements.
        $i := i + 1$
    until $(a == x) \,||\, (i == k)$
end
```

Monte Carlo algorithm vs Las Vegas algorithm

- Las Vegas algorithm: a randomised algorithm that always gives correct results, the only variation from one run to another is the running time
- Monte Carlo algorithm: a randomised algorithm whose running time is deterministic, but whose results may be incorrect with a certain (typically small) probability.
- Differences:
    - Monte Carlo algorithm runs for a fixed number of steps
    - Las Vegas algorithm runs in an infinite loop until the correct results are found
    - Las Vegas algorithm can be converted into Monte Carlo algorithm using early termination

Why use randomised algorithms?

- The element search problem is very simple, but
  - For deterministic sequential (linear) search algorithm, e.g., search the array one by one from the beginning:
    - Average run time complexity is $O(\frac{n+1}{2}) \approxeq O(n)$
    - The worst run time complexity is $O(n)$
  - For Las Vegas algorithm:
    - The average run time complexity depends on the input, e.g., the array and the element you want to find x (Question: what is the average time complexity of finding the first element of which the value equals to 1 if half the array contains 0s, the other half contains 1s?)
    - The worst run time complexity is unbound, i.e., could get super unlucky
  - For Monte Carlo algorithm:
    - The run time (including worst) complexity is fixed: $O(1)$
    - But with some probability of error: not finding the element

Solution to Problem 2: Nuts and Bolts Problem

- A disorganized carpenter has a collection of n nuts of distinct sizes and n bolts and would like to find the corresponding pairs of bolts and nuts. Each nut matches exactly one bolt (and vice versa), and he can only compare nuts to bolts, i.e., he can neither compare nuts to nuts nor bolts to bolts.
- The brute-force solution: to compare each nut with all bolts to find the matching bolt
  - Time complexity: $O(n^2)$ $\frac{n(n-1)}{2}$
- Can you help the carpenter match the nuts and bolts quickly?
- Hint:
  - **Think out of the box**: What if the carpenter does **NOT** have the constraints, i.e., he can compare nuts to nuts or bolts to bolts. Sort both
  - Now, with the constraint, what is the essence of this Nuts and Bolts Problem (Lock and Key problem)?

Hint: quicksort algorithm

- The sorting problem: given a array A of n numbers, sort the numbers in increasing order

Quicksort algorithm:

```
less, equal, greater := three empty arrays
if length(array) $> 1$
        pivot := select an element of array
        for each x in array
                if $x <$ pivot then add x to less
                if $x =$ pivot then add x to equal
                if $x >$ pivot then add x to greater
quicksort(less)
quicksort(greater)
array := concatenate(less, equal, greater)
```

Example 2: randomised quicksort algorithm

- Deterministic quicksort algorithm time complexity: $O(n \log n)$ on average for a random permutation array (n is the size of the array)
- **The worst case:** $O(n^2)$, e.g., for an sorted array
- Randomised quicksort algorithm: selecting a pivot randomly
  - Average run time complexity: $O(n \log n)$
  - The worst case: $O(2n \log n) \approx O(n \log n)$
  - Proof: Probabilistic Analysis and Randomized Quicksort
- An example of the second category of randomised algorithms: using random numbers to improve a solution to a problem

Solution to Problem 2

- Question: can you design an algorithm based on the idea of randomised quick sort to help the disorganised carpenter (Problem 2)? (The complexity should be $O(n \log n)$)
- This is not a trivial problem for deterministic algorithms: see the paper.

Problem 3: How can you help Dan Brown?

- Dan Brown is having holiday on Mars. He discovers a digital manuscript (a long string) of The Da Vinci Code in his laptop. He wants to compare this manuscript with his final version in his PC (also a long string) at home on Earth to see whether they are the same. He can access his manuscript in his PC at home, but the communication is very expensive. prime fingerprinting
- How should he do to compare these two versions via this expensive communication channel?
- Hint: Richard M. Karp (1991), An introduction to randomized algorithms, 34, Discrete Mathematics, 175-176, Algorithm 5.2

Applications of randomised algorithms

- Mathematics:
  - Number theory, e.g., primality test
  - Computational Geometry: graph algorithms, e.g., minimum cut
  - Linear algebra: matrix computations
- Computer Science:
  - Data analysis: PageRank
  - Parallel computing: Deadlock avoidance
  - Optimisation: Nature inspired Optimisation and Search algorithms
- Computational biology: DNA read alignment

Advantages/disavantages of randomised algorithms

- Advantages:
  - Simplicity: usually very easy to implement
  - Performance: usually produce (near-) optimum solutions with high probability
- Disadvantages:
  - Getting a wrong answer with a finite probability.
    - Solution: Repeat the algorithm
  - Difficult to analyse the running time and probability of getting an incorrect solution
  - Impossible to get truly random numbers

Conclusion

- Randomness is a resource to improve simplicity and efficiency of algorithms.
- In fact, randomness is also the important ‘resource’ in other fields, such as biology and physics.
- “*For many problems, a randomised algorithm is the simplest, the fastest or both*” - Prabhakar Raghavan, Vice President of Engineering at Google.

Further readings

- Motwani, Rajeev; Raghavan, Prabhakar (1995). Randomized Algorithms. New York: Cambridge University Press. ISBN 0-521-47465-5.
- Richard M. Karp (1991), An introduction to randomized algorithms, 34, Discrete Mathematics.
- Randomized algorithms for matrices and data