MERRIMACK COLLEGE

CSC6023 - Advanced Algorithms

Randomized Algorithms

Patrick Neff

Randomized Algorithms

This is the last full topic week of the course. So, let's see Randomized Algorithms, which is a popular approach to solve problems, sometimes with optimal solution.

MERRIMACK COLLEGE

Randomized Algorithms Presentation

Randomness

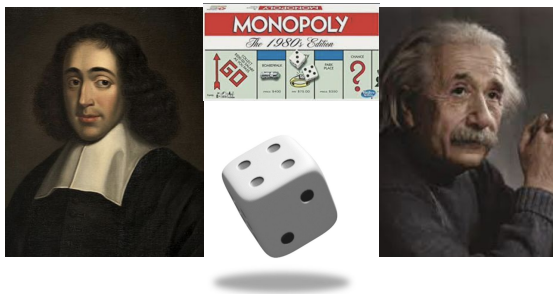
- Is there randomness?
  a. Pseudorandom generators
  b. Determinism and Randomness
- Solving with randomness
  a. Las Vegas and Monte Carlo

Random Algorithms Applications

- Quicksort
- Machine Learning Algorithms
  a. Bagging
  b. Random Forest
  c. Genetic Algorithms
- Hill Climbing

MERRIMACK COLLEGE

A Philosophical Approach



Is there such thing as randomness?

- Baruch Spinoza 1632-1677 - The Netherlands
  - "men are conscious of their desire and unaware of the causes by which [their desires] are determined."
- If you flip a coin, is the outcome random?
  - If you know everything of the initial state and the laws of physics, the outcome is deterministic
- What about free will?
- What about quantum physics?

- Regardless of the extent of the randomness and determinism, it is safe to say that the more you know, the less random the outcome seems.
  - Sometimes it is useful to be random (no bias)

Determinism and Randomness

MERRIMACK COLLEGE

What surprises you?

- If you could shoot a rock towards the moon it might reach the moon or not
  - That is very random
- If NASA shoots a space module towards the moon it might reach the moon more likely than if you shoot a rock
  - That is less random
- If the computational tools are perfectly adjusted, and both the physical knowledge and all factors are known, you can plan a mission to the moon that will reach the moon
  - That is as close as it gets to determinism
- The difference between randomness and determinism is what you know about the phenomenon

1, 2, 4, 8, 16

What’s next?

A Practical Approach

Linear Congruence Generator

MERRIMACK COLLEGE

Randomness inside a Computer

- Everything inside a computer is (hopefully) unavoidably deterministic
  - How to generate randomness?
- Pseudorandom Number Generators
  - A mathematical formula that is too complicated to be perceived, for example:

$$X_{n+1} = (aX_n + c) \bmod m$$

  - m, 0<m - the modulus
  - a, 0<a<m - the multiplier
  - c, 0<=c<m - the increment
  - X_0, 0<=X_0 < m - the seed/start value

A Practical Approach

Linear Congruence Generator

Randomness inside a Computer

- Everything inside a computer is (hopefully) unavoidably deterministic
  - How to generate randomness?
- Pseudorandom Number Generators
  - A mathematical formula that is too complicated to be perceived, for example:

$$X_{n+1} = (aX_n + c) \bmod m$$

  - For example, $m = 2^{31}$ $a = 1103515245$ $c = 12345$
    - $X_0 = 7$
    - $X_1 = (1103515245 (7) + 12345) \bmod 2^{31}$
      - $X_1 = 1282168116$

What would be X_2?

642666333

Let’s run some code!

Random Package in Python

Randomness inside a Computer

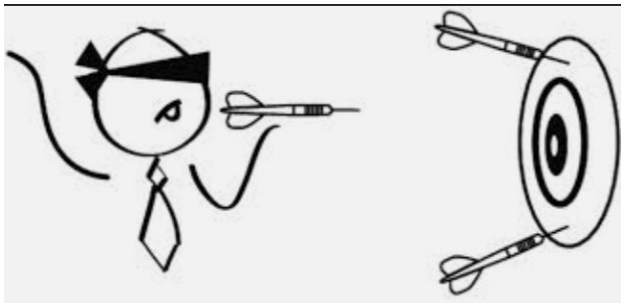
- Uses the Mersenne Twister algorithm to generate pseudorandom numbers
- Mersenne Twister is still a deterministic algorithm like the approach we looked at before
- It is possible to set the seed to see this clearly
- If no seed is set, the system time is used as the seed

Random Package Documentation:
https://docs.python.org/3/library/random.html

Mersenne Twister:
https://en.wikipedia.org/wiki/Mersenne_Twister

Let's run some code!

Solving with Randomness



Randomized Algorithms

- A randomized algorithm is an approach that is based in taking choices that are based on (pseudo) random decisions in some level
- Randomized algorithms can be exact, i.e., they can deliver exact (optimal) solution, or deliver approximated solution (may be suboptimal)
- The goal of randomized algorithms can be reduction of complexity, as for example, avoiding a brute force (exhaustive) solution
    - It may end up with time gains, or not
    - It may end with an approximated solution, or even an exact solution

Solving with Randomness

Find a zero among zeros and ones

- Given a (large) vector with an equal number of elements equal to 0 (zero) and equal to 1 (one) find an element that is equal to 1
- Las Vegas algorithm - *O(n) *kind of, sort of** and *average* constant time- *exact*
  - Repeat:
    - Pick randomly an element of the array
  - Until an one is found
- Monte Carlo algorithm - *O(1) - approx.*
  - For i in range(k):
    - Pick randomly an element of the array
    - If an one is found break out

Find the one in an array, Vegas way

Task #1 for this week's In-class exercises

- Create a program that generates a random array of 10,000 elements with equal number of 0's and 1's and then implement the randomized Las Vegas algorithm that searches for an element holding a 1
  - Your program should output the position of the first 1 found, plus the number of tries before founding it

Go to IDLE and program your solution
Save your program in a .py file and submit it in the appropriate delivery room

Deadline: Friday 11:59 PM EST

Find the one in an array, the Monte Carlo way

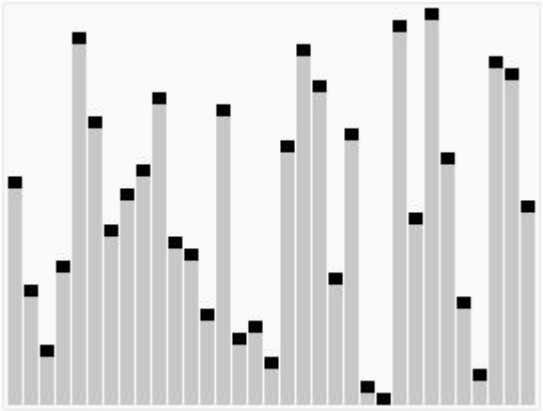
Task #2 for this week's In-class exercises

- Create a program that generates a random array of 10,000 elements with equal number of 0's and 1's and then implement the randomized Monte Carlo algorithm that searches for an element holding a 1
  - You can set *k* equal to 10
  - Your program should output the position of the first 1 found, plus the number of tries before founding it (or giving up)

Go to IDLE and program your solution
Save your program in a .py file and submit it in the appropriate delivery room

Deadline: Friday 11:59 PM EST

Quicksort



MERRIMACK COLLEGE

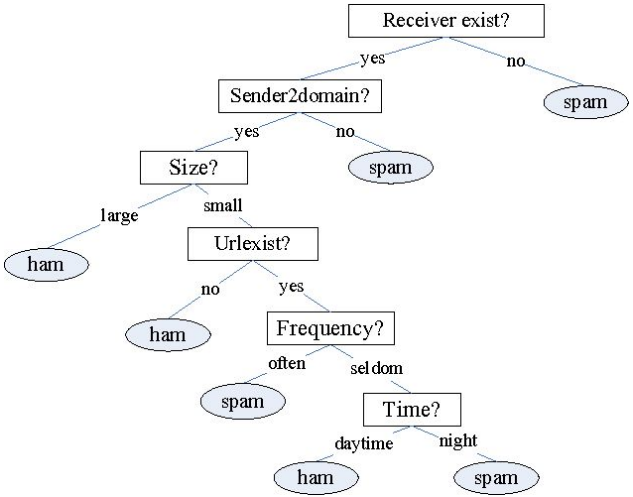
The choice of the pivot

- Quicksort algorithm picks one element (a pivot) then sort elements smaller than the pivot at its left, and greater elements at its right.
- While some input arrays may lead to an $O(n^2)$ time complexity, if the pivot is uniformly chosen randomly it has a high probability of $O(n \log n)$ time regardless of the initial array.
    - For example if the input is an already sorted array, only picking the first or last element the worst case time will occur.
        - For a large array (for example, n=100000) the probability to randomly choose the first or last element is very small (for this example, 0.00002).

5 minute break

Random Algorithms

Machine Learning

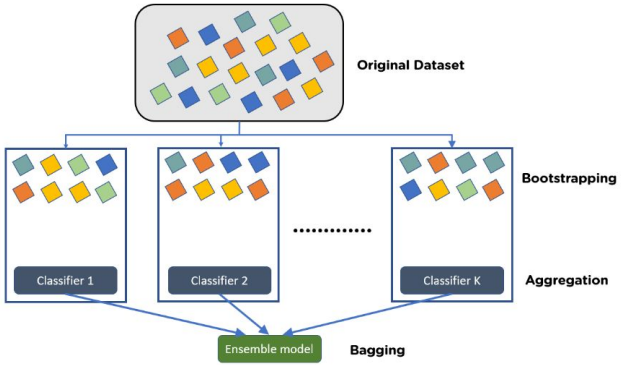


MERRIMACK COLLEGE

The basic idea

- To predict what is going to happen in a given situation characterized by a certain number of attributes, one option is to start with a certain number of situations that you know both the attributes and the outcome and "learn" what leads to the outcome.
  - For example, to decide if an email is spam or not you can define several attributes for emails (e.g. existence of specific words, number of recipients) and several examples (emails) classified if they are spam or not.
- An old traditional approach is to create a decision tree that sorts out the outcome.
  - This approach is deterministic and yet sometimes it fails.

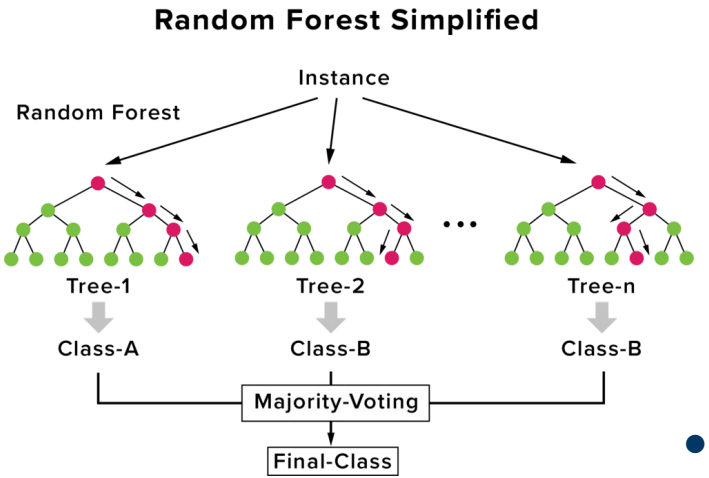
Machine Learning



The Bagging algorithm

- Instead of creating a single decision tree, the training dataset (cases with classified outcome) can be sampled several times and a different decision tree is created for each sample.
- To classify an email each decision tree is applied and their outcome is considered as a vote and the final decision is the one with the most votes.
- While the traditional approach (single decision tree) is effective in well behaved scenarios (good training dataset, good choice of attributes, logical decisions), for misbehaving ones Bagging is more flexible and, thus, usually more effective.

Machine Learning



MERRIMACK COLLEGE

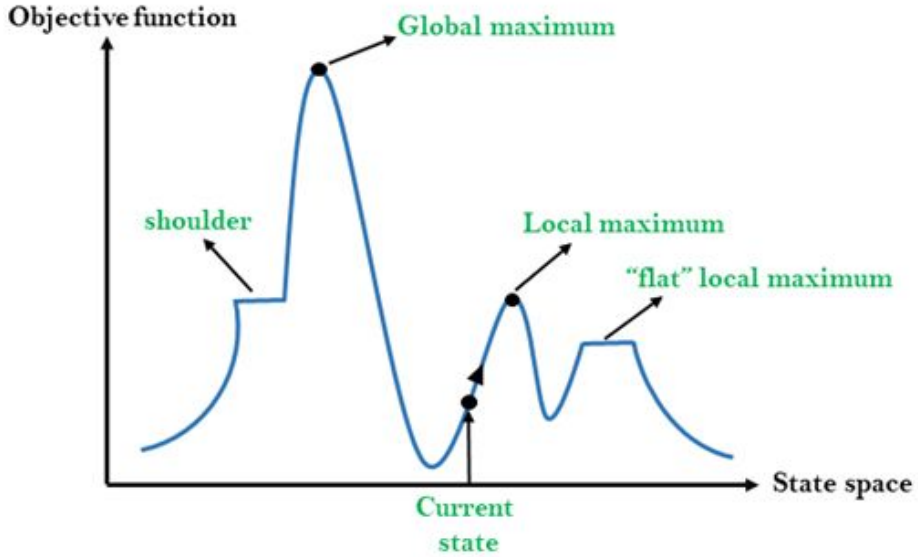
The Random Forest algorithm

- While Bagging sample examples come from the training set (to chose the rows of the training set), the Random Forest randomly chooses the examples and also some attributes (to chose the columns of the training set).
  - Random Forest is more stable than Bagging
  - Curiously, Bagging and Random Forest were proposed by the same researcher (Leo Breiman), who provided ample statistical justification for the algorithms.
- Machine Learning is by nature an approximation algorithm, thus the inexact solution is not an issue.

Looking for the maximum

Hill Climbing

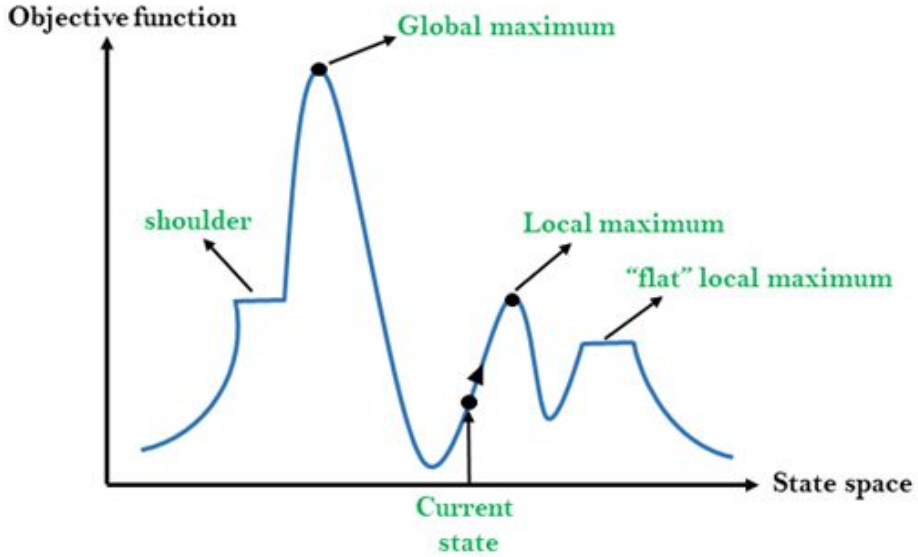
- If a problem has a function which is very costly to compute, and we need to find the optimal parameter for it, hill climbing can be an interesting solution.
  - Hill climbing looks for local maximum near a specific search point (current state).



Looking for the maximum

Hill Climbing

- The choice of the initial search point has great impact on the optimal or suboptimal solution.
- The random choice of the initial search point(s) may tackle the deterministic stiffness of the algorithm.



Project - this week's Assignment

- Create a program that calls the function ***myFunction(x)*** from 0 to 9999 and applies the Hill Climbing algorithm to find the value of ***x*** that delivers the largest value for the objective function.
- Implement a version where the initial search value ***x*** is chosen randomly between the values 1 to 9998.
- You can use this function to test your program, but your program should work with any version of ***myFunction(x)***.

```
def myFunction(x):
    if (x == 0):
        return 0
    elif ((log2(x) * 7) % 17) < (x % 13):
        return (x + log2(x))**3
    elif ((log2(x) * 5) % 23) < (x % 19):
        return (log2(x)*2)**3
    else:
        return (log2(x)**2) - x
```

Seventh (and last) Assignment

Project - this week's Assignment

- This program must be your own, do not use someone else's code
- Any specific questions about it, please bring to the last Office hours meeting this Friday or contact me by email
- This is a challenging program to make sure you are mastering your Python programming skills
- Don't be shy with your questions

Go to IDLE and try to program it
Save your program in a .py file and submit it in the appropriate delivery room

Deadline: Next Monday 11:59 PM EST

That’s all for today folks!

This week’s tasks

- Tasks #1 and #2 for the In-class exercises
  - Deadline: This Friday 11:59 PM EST
- Quiz to be available this Friday
  - Deadline: Next Monday 11:59 PM EST
- Project assignment
  - Deadline: Next Monday 11:59 PM EST

- Try all exercises seen in class and consult the reference sources, as the more you practice, the easier it gets

Next week

- Advanced data structures

- Don’t be shy about your questions

MERRIMACK COLLEGE

Have a Great Week!