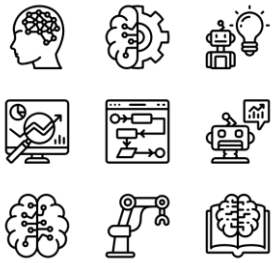
Computer Science for Practicing Engineers

Thuật toán heurictic
Thuật toán xác suất



TS. Huỳnh Bá Diệu
Email: dieuhb@gmail.com
Phone: 0914146868

1

Nội dung

1. Thuật toán heuristic là gì?
2. Thuật toán xác suất là gì?
3. Phân lớp thuật toán xác suất

2

Thuật toán heuristic

A heuristic algorithm is one that is designed to solve a problem in a faster and more efficient fashion than traditional methods by sacrificing optimality, accuracy, precision, or completeness for speed.

Heuristic algorithms often times used to solve NP-complete problems, a class of decision problems. In these problems, there is no known efficient way to find a solution quickly and accurately although solutions can be verified when given. Heuristics can produce a solution individually or be used to provide a good baseline and are supplemented with optimization algorithms.

Heuristic algorithms are most often employed when approximate solutions are sufficient and exact solutions are necessarily computationally expensive

3

Thuật toán heuristic

Thuật toán: là dãy xác định các câu lệnh đảm bảo việc tìm ra lời giải cho bài toán.

Thuật toán heuristic: tập các qui luật, kinh nghiệm khi giải quyết bài toán. Thuật toán heuristic không đảm bảo việc cho ra lời giải đúng.

Thuật toán HA được suy ra từ kinh nghiệm, cho phép tìm ra lời giải có thể chấp nhận được nhưng không chứng minh tính đúng đắn hay tối ưu.

4

Thuật toán heuristic

Thuật toán heuristic bỏ qua tiêu chí tối ưu hay biết rõ thời gian, tài nguyên khi thực hiện thuật toán.

Thuật toán AH chỉ tìm ra giải pháp chấp nhận được, không chứng minh được nó tối ưu hơn giải pháp khác.

Ví dụ: Bài toán mua hay thuê nhà.

Rent-A-House®

5

Thuật toán heuristic

Có những yếu tố ảnh hưởng đến bài toán mà ta không thể định giá được khi xây dựng thuật toán.

Thuật toán H dù không cho ra lời giải tối ưu nhưng vẫn sử dụng vì nó làm đơn giản quyết định và cho ra lời giải đủ tốt.

6

Thuật toán heuristic

Ví dụ bài toán người du lịch qua n thành phố.

Dùng vét cạn: không thể thử hết n! các trường hợp O(n!)

Nếu n= 21 thì thử (n-1)! trường hợp

= 2,432,902,008,176,640,000

Dùng H thì độ phức tạp là $O(n^2)$

Vì sao là $O(n^2)$???

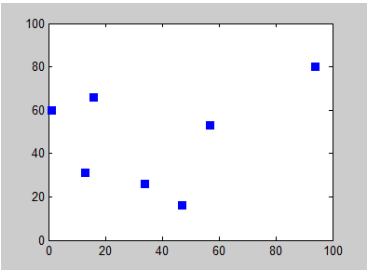
7

Thuật toán heuristic: **Traveling Salesmen Problem**

Nearest neighbor (NN) algorithm (also known as the Greedy Algorithm).

Starting from a randomly chosen city, the algorithm finds the closest city. The remaining cities are analyzed again, and the closest city is found.

1. Start at a random vertex
2. Determine the shortest distance connecting the current vertex and an unvisited vertex *V*
3. Make the current vertex the unvisited vertex *V*
4. Make *V* visited
5. Record the distance traveled
6. Terminate if no other unvisited vertices remain
7. Repeat step 2



8

Thuật toán heuristic: **Traveling Salesmen Problem**

There are 4 points of interest located in a 10x10 plot of space: (3,4.5), (9,6.25), (1,8), and (5.5,0). The table below lists the distance required to touch all 4 points with the first and last point known using the nearest neighbor algorithm:

		Ending point			
		(5.5,0)	(1,8)	(3,4.5)	(9,6.25)
Starting Point	(5.5,0)	0	9.18	5.15	7.16
	(1,8)	9.18	0	4.03	8.19
	(3,4.5)	5.15	4.03	0	6.25
	(9,6.25)	7.16	8.19	6.25	0

9

Thuật toán heuristic: **Traveling Salesmen Problem**

Starting at point (1,8): The shortest distance to an unvisited point is 4.03 units to point (3,4.5). The shortest distance to an unvisited point is 5.15 units to point (5.5,0). The shortest distance to an unvisited point is 7.16 units to point (9,6.25). The total distance traveled is 16.34 units.

Starting at point (9,6.25): The shortest distance to an unvisited point is 6.25 units to point (3,4.5). The shortest distance to an unvisited point is 4.03 units to point (1,8). The shortest distance to an unvisited point is 9.18 units to point (5.5,0). The total distance traveled is 19.46 units. but not the best solution.

		Ending point			
		(5.5,0)	(1,8)	(3,4.5)	(9,6.25)
Starting Point	(5.5,0)	0	9.18	5.15	7.16
	(1,8)	9.18	0	4.03	8.19
	(3,4.5)	5.15	4.03	0	6.25
	(9,6.25)	7.16	8.19	6.25	0

Both situations followed the NN algorithm to solve the problem, however the total distance traveled changed based on the started location. This shows how a heuristic algorithm can give a good solution.

10

Thuật toán xác suất (probabilistic algorithms)

Thuật toán đơn định: với 1 dữ liệu vào chỉ có duy nhất một dữ liệu ra

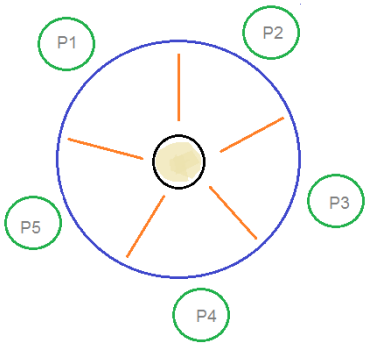
Thuật toán xác suất là thuật toán mà kết quả của nó phụ thuộc vào một số ngẫu nhiên.

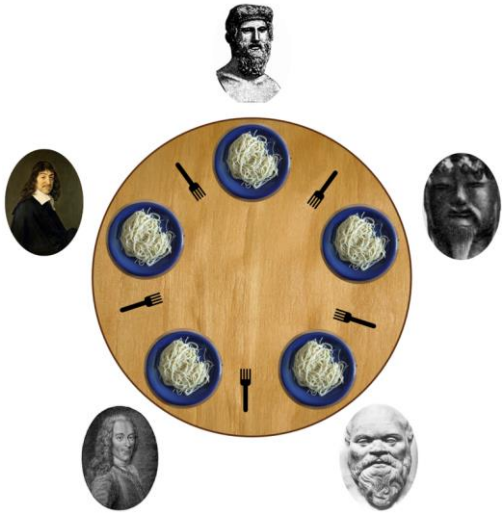
Thuật toán xác suất còn gọi là thuật toán ngẫu nhiên.

Hoạt động của thuật toán có thể khác nhau trên cùng một dữ liệu vào.

11

Thuật toán xác suất: bữa ăn của 5 nhà triết học

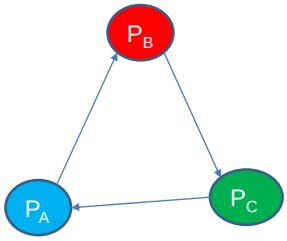


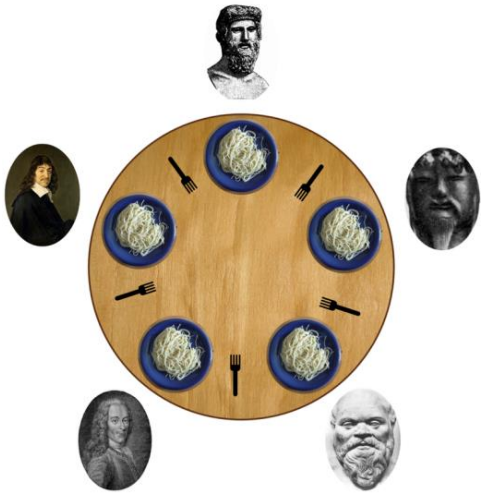


12

Thuật toán xác xuất: bữa ăn của 5 nhà triết học

Hệ điều hành: Quản lý tài nguyên găng

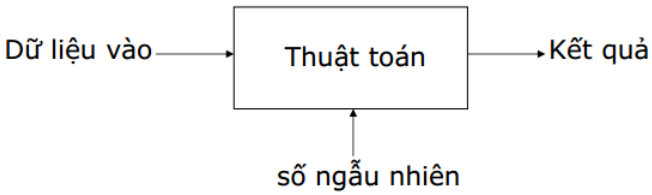




13

Thuật toán xác suất: probabilistic algorithms

- Ưu điểm:
- Đơn giản
- Hiệu quả
- Nhược điểm:
- Có thể cho kết quả sai
- Hoặc không dừng



14

Thuật toán xác suất: probabilistic algorithms

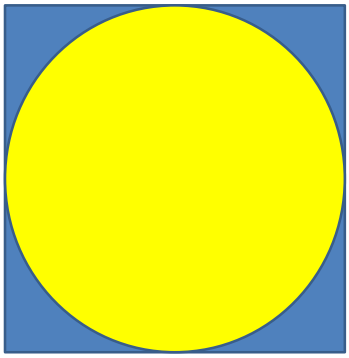
Ví dụ: Tính gần đúng số pi

Cho hình vuông có độ dài cạnh a

Hình tròn nội tiếp hình vuông có bán kính a/2

Diện tích hình vuông là a^2

Diện tích hình tròn là (a^2 * pi) /4



15

Thuật toán xác suất: probabilistic algorithms

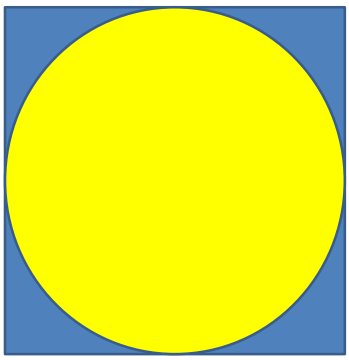
Chấm n điểm trên hình vuông

Số điểm nằm trong đường tròn là k

k/n = (diện tích hình tròn / diện tích hình vuông)

$$\frac{k}{n} = \frac{\frac{\pi a^2}{4}}{a^2} = \frac{\pi}{4}$$

$$\Rightarrow \pi = 4k/n$$



16

Thuật toán xác suất: probabilistic algorithms

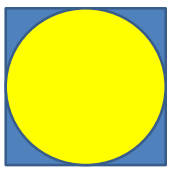
Các bước xây dựng chương trình:

Giả sử đường tròn có đường kính= 1

```
double tinh_pi(int n)
{
        double k=0;
        Lặp từ 1 đến n
         {
             Sinh điểm  ngẫu nhiên (x, y) (thuộc đoạn 0 --1)
             Nếu điểm (x, y) thuộc đường tròn thì tăng biến k lên 1;
         }
         return 4*k/n;
}
```

17

Thuật toán xác suất: probabilistic algorithms



Kiểm tra điểm thuộc đường tròn:

$x^2 + y^2 <= 1$

```
x= Math.random();
y= Math.random();
```

Gọi hàm main:

N=1000

N=1000000

18

Thuật toán xác suất: probabilistic algorithms

```
public static void main(String[] args) {
    CSPE_MyP11_THUAT_TOAN_XAC_XUAT_TINH_PI m= new
      CSPE_MyP11_THUAT_TOAN_XAC_XUAT_TINH_PI();
    System.out.println("\n  PI= " + m.tinhPi(1000));
    System.out.println("\n  PI= " + m.tinhPi(1000000));
}
```

```
run:

  PI= 3.208

  PI= 3.143336
BUILD SUCCESSFUL (total time: 0 seconds)
```

19

Thuật toán xác suất: probabilistic algorithms

```
public static void main(String[] args) {
    CSPE_MyP11_THUAT_TOAN_XAC_XUAT_TINH_PI m= new
      CSPE_MyP11_THUAT_TOAN_XAC_XUAT_TINH_PI();
    System.out.println("\n  PI= " + m.tinhPi(1000));
    System.out.println("\n  PI= " + m.tinhPi(1000000));
}
```

```
run:

  PI= 3.208

  PI= 3.143336
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
double tinhPi(int n)
  {
     double k=0;
     for (int i=1; i<=n; i++)
     {
        double x= Math.random();
        double y= Math.random();
        if(x*x + y*y <=1) k++;
     }
     return 4*k/n;
  }
```

20

Thuật toán xác suất: probabilistic algorithms

Cho dãy A gồm n phần tử. Dãy A được gọi là dãy chứa phần tử đa số nếu như một phần tử nào đó trong mảng chiếm hơn nữa.

Hãy kiểm tra mảng a có phải đông đúc không.

Dữ liệu vào FILE DD.inp chứa các số nguyên

Dữ liệu ra FILE DD.out chứa số 0 hoặc 1.

-- Cách tiếp cận bình thường: SOL1()

-- Cách tiếp cận theo thuật toán xác suất: SOL2()

[4 2 5 4 7 5 5 8 3 4 5 5 1 5 5 5 2 5 5 5 5 5 1 5 5 8 4 6 5 5 5 5 5 5]

21

Thuật toán xác suất: probabilistic algorithms

Cho dãy A gồm n phần tử. Dãy A được gọi là dãy chứa phần tử đa số nếu như một phần tử nào đó trong mảng chiếm hơn nữa.

Hãy kiểm tra mảng a có phải đông đúc không.

Dữ liệu vào FILE DD.inp chứa các số nguyên

Dữ liệu ra FILE DD.out chứa số 0 hoặc 1.

-- Cách tiếp cận bình thường: SOL1()

-- Cách tiếp cận theo thuật toán xác suất: SOL2()

[4 2 5 4 7 5 5 8 3 4 5 5 1 5 5 5 2 5 5 5 5 5 1 5 5 8 4 6 5 5 5 5 5 5]

```
1. Sắp xếp
2. Cho i từ 1 đến n/2 -1
        nếu a[i]= a[i+n/2] return 1
3. return 0
```

22

Phân lớp các Thuật toán xác suất (Classification of PA)

There is a variety of behaviours associated with probabilistic algorithms:

1/ Algorithms which ***always return a result, but the result may not always be correct***. We attempt to minimise the probability of an incorrect result, and using the random element, multiple runs of the algorithm will reduce the probability of incorrect results. These are sometimes called **Monte Carlo algorithms**. Terminology has not been fixed, and varies with author.

2/ Algorithms that ***never return an incorrect result, but may not produce results at all on some runs***. Again, we wish to minimise the probability of no result, and, because of the random element, multiple runs will reduce the probability of no result. These are sometimes called **Las Vegas algorithms**.

23

Phân lớp các Thuật toán xác suất (Classification of PA)

Classification of probabilistic algorithms

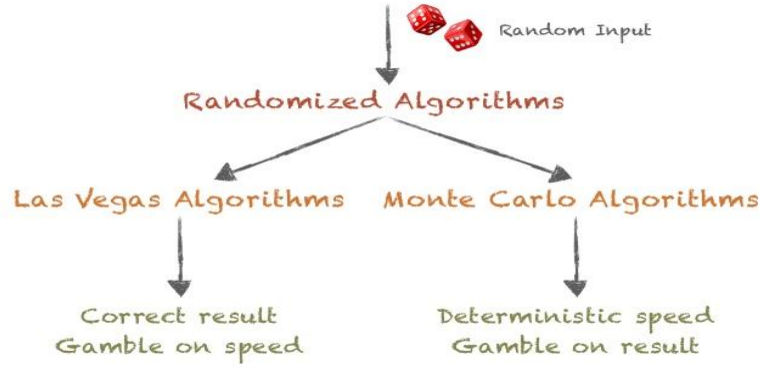
There is a variety of behaviours associated with probabilistic algorithms:

1/ Algorithms which ***always return a result, but the result may not always be correct***. We attempt to minimise the probability of an incorrect result, and using the random element, multiple runs of the algorithm will reduce the probability of incorrect results. These are sometimes called **Monte Carlo algorithms**. Terminology has not been fixed, and varies with author.

2/ Algorithms that ***never return an incorrect result, but may not produce results at all on some runs***. Again, we wish to minimise the probability of no result, and, because of the random element, multiple runs will reduce the probability of no result. These are sometimes called **Las Vegas algorithms**.

24

Phân lớp các Thuật toán xác suất (Classification of PA)



25

Thuật toán xác suất: probabilistic algorithms

Example: finding an '*a*' in an array of *n* elements.

- **Input**: An array of $n \geq 2$ elements, in which half are '*a*'s and the other half are '*b*'s.
- **Output**: Find an '*a*' in the array.

We give two versions of the algorithm, one Las Vegas algorithm and one Monte Carlo algorithm.

26

Thuật toán xác suất: Las Vegas algorithm

```
findingA_LV(array A, n)
begin
  repeat
    Randomly select one element out of n elements.
  until 'a' is found
end
```

This algorithm succeeds with probability 1. The number of iterations varies and can be arbitrarily large, but the expected number of iterations is:

$$\lim_{n\to\infty} \sum_{i=1}^{n} \frac{i}{2^i} = 2$$

Since it is constant the expected run time over many calls is Theta(1).

https://en.wikipedia.org/wiki/Randomized_algorithm

27

Thuật toán xác suất: Monte Carlo

```
findingA_MC(array A, n, k)
begin
  i=0
  repeat
    Randomly select one element out of n elements.
    i = i + 1
  until i=k or 'a' is found
end
```

If an '*a*' is found, the algorithm succeeds, else the algorithm fails. After *k* iterations, the probability of finding an '*a*' is:

$$\Pr[\text{find a}] = 1 - (1/2)^k$$

This algorithm does not guarantee success, but the run time is bounded. The number of iterations is always less than or equal to k. Taking k to be constant the run time (expected and absolute) is Theta(1).

28

Bài tập về nhà

Cài đặt thuật toán kiểm tra số nguyên n có phải số nguyên tố không theo thuật toán **Miller-Rabin.**

Tham khảo thêm tại:
http://www.giaithuatlaptrinh.com/?p=278

29

Tài liệu đọc thêm

https://www.geeksforgeeks.org/dining-philosopher-problem-using-semaphores/

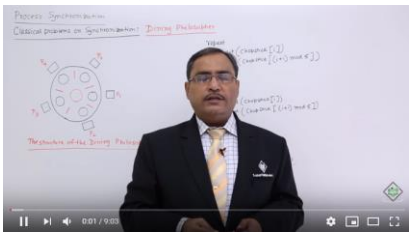
https://www.geeksforgeeks.org/randomized-algorithms/

30

Link YouTube

https://www.youtube.com/watch?v=9f1oOMX3mP4

https://www.khanacademy.org/computing/computer-science/cryptography/random-algorithms-probability/v/randomized-algorithms-prime-adventure-part-8



31