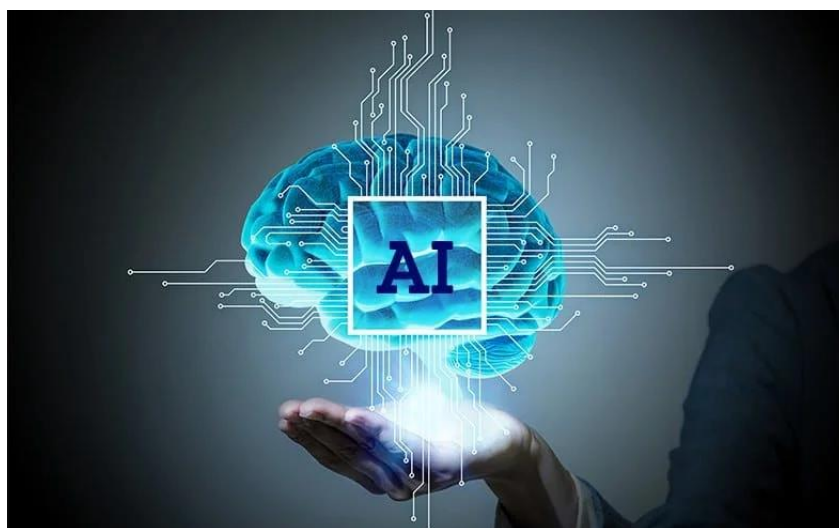


**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
KHOA TOÁN - TIN HỌC**

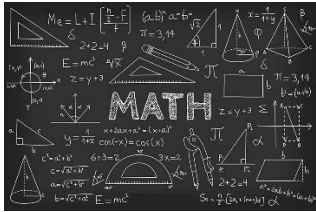


BÀI BÁO CÁO THỰC HÀNH TUẦN 2



MÔN HỌC: Phân Tích Thuật Toán
Sinh Viên: Trần Công Hiếu - 21110294
Lớp: 21TTH

TP.HCM, ngày 14 tháng 04 năm 2024



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA TOÁN – TIN HỌC



BÀI BÁO CÁO THỰC HÀNH TUẦN 2

HK1 - NĂM HỌC: 2024-2025

MÔN: PHÂN TÍCH THUẬT TOÁN

SINH VIÊN: TRẦN CÔNG HIẾU

MSSV: 21110294

LỚP: 21TTH

[illegible]

Giảng viên Bộ môn

Mục Lục

Bài 1.	5
1.Trình bày đoạn mã.....	5
1.1. Chương trình với N cố định.....	5
1.2. Chương trình với N tăng dần theo thứ tự 10, 20, ..., 1000.	8
2. Chứng minh thuật toán.	9
3. So sánh kết quả.....	10
3.1. Kết quả thực nghiệm.	10
3.2. Kết quả lý thuyết.....	13
3.3. Nhận xét chung.....	14

Bài 1.

1.Trình bày đoạn mã

1.1. Chương trình với N cố định.

```
1 import numpy as np
2
3 N = 1000
4 A = sorted(np.random.choice(np.arange(1, 10001), size=N, replace=False))
5 print(A)
6
7 x = int(50)
8 count = 0
9
```

(File: bai1_1.py).

“import numpy as np”: Gọi thư viện numpy để thao tác với mảng trong đoạn mã của chương trình và gọi tắt là “np”, điều này cho phép gọi các hàm và đối tượng từ thư viện numpy bằng cách sử dụng tiền tố “np”.

“N = 1000”: Khởi gán biến n bằng 1000, đây là số phần tử của mảng, ta xét trường hợp là N = 1000. Ở đoạn mã sau, ta sẽ tạo vòng lặp for để xét N với từng giá trị 10, 20, ..., 1000 như yêu cầu bài toán.

“A = sorted(np.random.choice())”: Trong phương thức choice() có đối số np.arange(1, 10001). Nó tạo một mảng numpy chứa các số từ 1 đến 10000. Dùng phương thức choice() để chọn ngẫu nhiên size=N phần tử từ mảng chứa đoạn [1,10000] trên. Và với việc replace=False sẽ đảm bảo không cho phép lặp lại việc chọn một phần tử đã được chọn trước đó, đúng với yêu cầu các phần tử đôi một khác nhau. Và cuối cùng sắp xếp các phần tử trong mảng để mảng có thứ tự thông qua hàm sorted. Lúc này, A là mảng chứa N phần tử đôi một khác nhau và các phần tử được sắp xếp từ bé đến lớn, thuận tiện cho việc sử dụng thuật toán tìm kiếm nhị phân.

“`x = int(50)`”: Khởi gán biến `x` với giá trị 50 thuộc kiểu dữ liệu số nguyên như yêu cầu bài toán.

“`count = 0`”: Khởi tạo biến `count` và gán bằng 0 để nếu không tồn tại cặp (i, j) nào thỏa thì sẽ in ra thông báo.

```
10 for i in range(N):
11     target = A[i]
12     temp = i
13     while (i <= N):
14         mid = int(i + (N-i)/2)
15         if ((target == x - A[mid]) and (temp != mid)):
16             print("- Cap (i,j):", temp, mid)
17             print("  Voi gia tri la:", target, A[mid])
18             count+=1
19             break
20         elif (x-A[mid] < target):
21             N = mid-1
22         else:
23             i = mid+1
24     if count == 0: print("Khong ton tai (i,j)")
25
```

“`for i in range(N):`”: Tạo vòng lặp để duyệt từng phần tử trong mảng `A`, ứng với từng phần tử có index là `i`, ta dùng vòng `while` để tìm kiếm phần tử có index `j` còn lại. Vào vòng lặp:

“`target = A[i]`”: Khởi tạo biến `target` để lưu giá trị `A[i]`, bởi trong đoạn mã `i` sẽ thay đổi nên phải lưu tạm vào một biến khác là `target`.

“`temp = i`”: Lưu tạm chỉ số index `i` đang xét vào biến `temp` để in thông báo.

“`while(i <= N)`”: Tạo vòng lặp `while` với điều kiện `i <= N`. Bắt đầu thuật toán tìm kiếm nhị phân.

“`mid = int(i + (N-i)/2)`”: Khởi gán biến `mid` bằng biểu thức, biểu thức này luôn cập nhật được `mid` của mảng sau khi “bị cắt”.

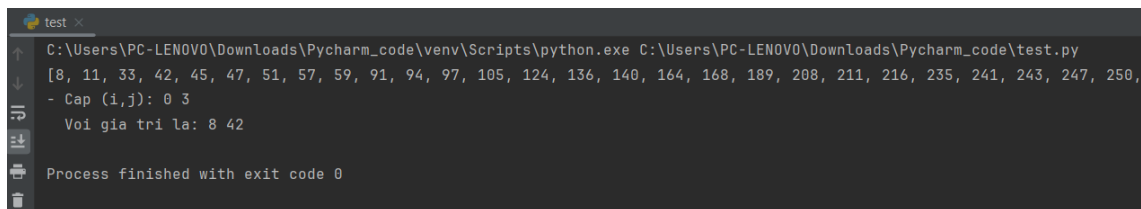
“if ((target == x - A[mid]) and (temp != mid)):” Xét điều kiện nếu phần tử thứ i (i ban đầu với giá trị tại i đã lưu là target) có bằng x - A[mid] hay không. Và để tránh trường hợp đặc biệt tồn tại phần tử trong mảng mang giá trị 25, khi i và mid cùng tiến về index mà giá trị tại index đó bằng 25, kết quả trả về cặp (i, j) có i = j thì ta cho thêm điều kiện kiểm tra là 2 index đó phải khác nhau. Và ta in thông báo kết quả vừa tìm được, biến count cộng thêm 1 đơn vị chỉ để tí kiểm tra nếu không khác 0 thì in thông báo không tồn tại cặp (i, j) thỏa. Cuối cùng là dùng lệnh break để thoát khỏi vòng lặp while hiện tại, bởi nếu không thoát thì vòng lặp sẽ chạy vô hạn.

“elif (x-A[mid] < target):”: Nếu không thỏa điều kiện if trên, kiểm tra tiếp nếu x-A[mid] < target thì ta phải điều chỉnh khoảng sao cho giá trị của A[mid] giảm, tức N = mid-1. Ngược lại với các trường hợp trên thì i = mid-1.

Cuối cùng là in ra thông báo nếu không tìm thấy cặp giá trị (i, j) thỏa.

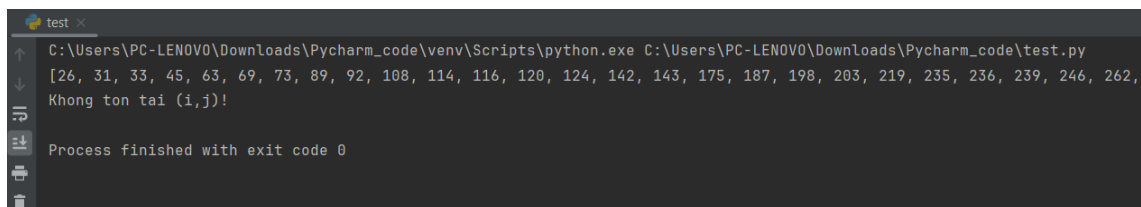
Kết quả.

- Khi tồn tại cặp (i, j) thỏa $A[i] + A[j] = x$.



```
test x
C:\Users\PC-LENOVO\Downloads\Pycharm_code\venv\Scripts\python.exe C:\Users\PC-LENOVO\Downloads\Pycharm_code\test.py
[0, 11, 33, 42, 45, 47, 51, 57, 59, 91, 94, 97, 105, 124, 136, 140, 164, 168, 189, 208, 211, 216, 235, 241, 243, 247, 250,
- Cap (i,j): 0 3
Voi gia tri la: 8 42
Process finished with exit code 0
```

- Khi không tồn tại cặp (i, j) thỏa $A[i] + A[j] = x$.



```
test x
C:\Users\PC-LENOVO\Downloads\Pycharm_code\venv\Scripts\python.exe C:\Users\PC-LENOVO\Downloads\Pycharm_code\test.py
[26, 31, 33, 45, 63, 69, 73, 89, 92, 108, 114, 116, 120, 124, 142, 143, 175, 187, 198, 203, 219, 235, 236, 239, 246, 262,
Khong ton tai (i,j)!
Process finished with exit code 0
```

1.2. Chương trình với N tăng dần theo thứ tự 10, 20, ..., 1000.

Đưa bài toán trên vào function với biến đầu vào là N. Sau đó dùng vòng lặp for, ta sẽ được chương trình với N tăng dần theo thứ tự 10, 20, ..., 1000 như yêu cầu bài toán. (File: bai1_2.py)

```
1 import numpy as np
2 usage
3 def Check(N):
4     A = sorted(np.random.choice(np.arange(1, 10001), size=N, replace=False))
5     #print(A)
6     x = int(50)
7     count = 0
8     for i in range(N):
9         target = A[i]
10        temp = i
11        while (i <= N):
12            mid = int(i + (N-i)/2)
13            if ((target == x - A[mid]) and (temp != mid)):
14                print("- Cap (i,j):", temp, mid)
15                print(" Voi gia tri la:", target, A[mid])
16                count+=1
17                break
18            elif (x-A[mid] < target):
19                N = mid-1
20            else:
21                i = mid+1
22        if count == 0: print("Khong ton tai (i,j) thoa yeu cau bai toan")
23
24    for i in range(10, 1001, 10):
25        print("Voi N = ", i)
26        Check(i)
27
```

Kết quả thu được:


```

test1 x
Khong ton tai (i,j) thoa yeu cau bai toan
Voi N = 930
Khong ton tai (i,j) thoa yeu cau bai toan
Voi N = 940
Khong ton tai (i,j) thoa yeu cau bai toan
Voi N = 950
Khong ton tai (i,j) thoa yeu cau bai toan
Voi N = 960
Khong ton tai (i,j) thoa yeu cau bai toan
Voi N = 970
- Cap (i,j): 0 8
  Voi gia tri la: 3 47
- Cap (i,j): 2 6
  Voi gia tri la: 19 31
- Cap (i,j): 4 5
  Voi gia tri la: 22 28
Voi N = 980
Khong ton tai (i,j) thoa yeu cau bai toan
Voi N = 990
Khong ton tai (i,j) thoa yeu cau bai toan
Voi N = 1000
Khong ton tai (i,j) thoa yeu cau bai toan

Process finished with exit code 0

```

2. Chứng minh thuật toán.

- So sánh:

$$T_1(N) = \sum_{i=0}^{N-1} \left(\sum_{j=0}^{\lfloor \log_2 N \rfloor} (1) \right) + 1 = N \cdot \log_2(N) + 1$$

- Gán:

$$T_2(N) = 4 + \sum_{i=0}^{N-1} \left(\sum_{j=0}^{\lfloor \log_2 N \rfloor} (2) \right) = 2N \cdot \log_2(N) + 4$$

Ta được:

$$T(N) = T_1(N) + T_2(N) = 3N \cdot \log_2(N) + 5$$

Ta đi chứng minh: $T(N) = O(N \cdot \log_2 N)$.

Tức ta cần chỉ ra tồn tại c và N_0 là các hằng số dương sao cho $T(N) \leq c \cdot (N \log_2 N)$ với mọi $N \geq N_0$.

Xét:

$$3N \cdot \log_2(N) + 5 \leq c \cdot N \log_2(N) \quad (*)$$

Ta sẽ tạm thời bỏ 5 ở vế trái để xét riêng tìm c để:

$$3N \cdot \log_2(N) \leq c \cdot N \log_2(N)$$

Sau đó ta xét với c thêm 1 đơn vị để tách ra phần $N \log(N)$ rồi tìm N sao cho $N \log_2(N) \geq 5$, khi đó: N vừa tìm được sẽ chính là N_0 và thỏa (*).

Lúc này:

$$\begin{aligned} 3N \cdot \log_2(N) &\leq c \cdot N \log_2(N) \\ \Leftrightarrow \quad 3 &\leq c \end{aligned}$$

Chọn $c = 4$. Lúc này, ta cần tìm N thỏa:

$$\begin{aligned} 3N \cdot \log_2(N) + 5 &\leq 3 \cdot N \log_2(N) + N \log_2(N) \\ \Leftrightarrow 5 &\leq N \log_2(N) \end{aligned}$$

Suy ra: $N = 4$.

Chọn $c = 11$, $N_0 = 4$ thì với mọi $n \geq 4$, ta có:

$$T(N) = 3N \cdot \log_2(N) + 5 \leq 4 \cdot (N \log_2 N)$$

Suy ra: $T(N) = O(N \log(N))$.

3. So sánh kết quả.

3.1. Kết quả thực nghiệm.

Ta sử dụng hàm time đặt trong function Check(N), để lấy time hiện tại sau khi kết thúc hàm, tức kết thúc việc kiểm tra với từng giá trị i , ta lấy time hiện tại bằng cú pháp time.time() rồi trừ cho giá trị time ban đầu lưu vào biến start. Hơn nữa, để đảm bảo

thời gian chạy đoạn mã chính xác, ta sẽ hạn chế (bỏ) các thao tác gán, in thông báo chỉ nhằm mục đích hiển thị ra bằng cách comment những đoạn mã đó, để chương trình chỉ tập trung vào việc tính toán và trả về thời gian thực. (File: bai1_3.py)

```
3 def Check(N):
4     start = time.time()
5     A = sorted(np.random.choice(np.arange(1, 10001), size=N, replace=False))
6     #print(A)
7
8     x = int(50)
9     count = 0
10
11    for i in range(N):
12        target = A[i]
13        #temp = i
14        while (i <= N):
15            mid = int(i + (N-i)/2)
16            if ((target == x - A[mid]) and (i != mid) ):
17                #print("- Cap (i,j):",temp,mid)
18                #print(" Voi gia tri la:",target, A[mid])
19                #count+=1
20                break
21            elif (x-A[mid] < target):
22                N = mid-1
23            else:
24                i = mid+1
25        #if count == 0: print("Khong ton tai (i,j) thoa yeu cau bai toan")
26    return float(time.time() - start)
```

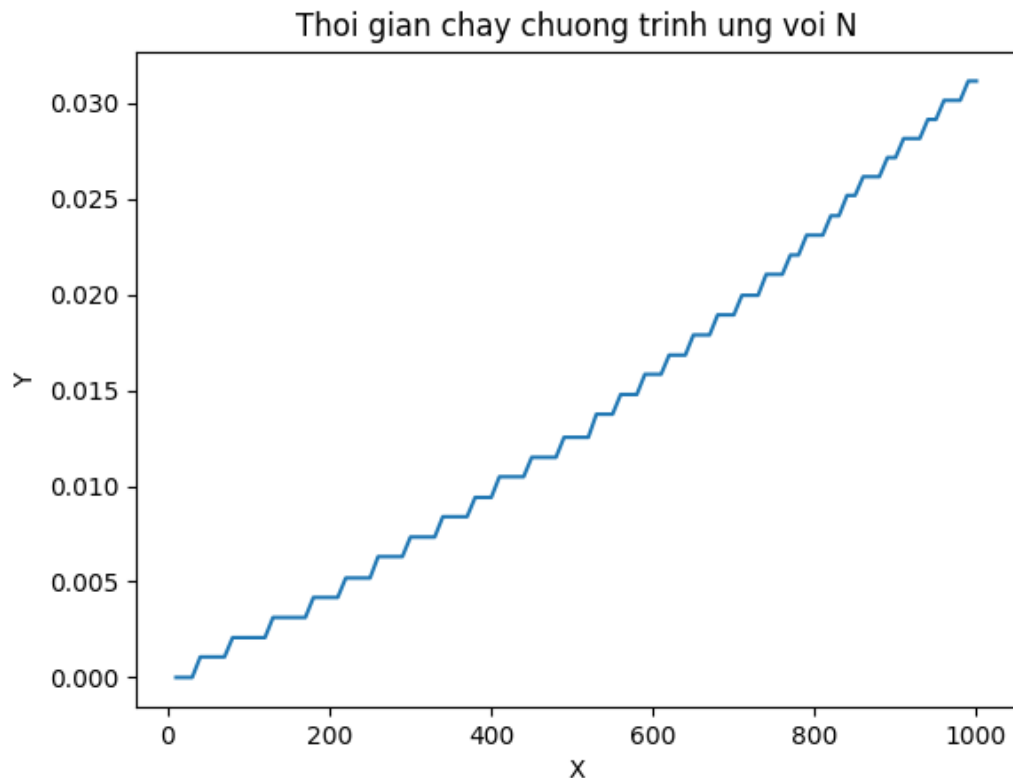
Hàm return về giá trị là khoảng thời gian chạy đoạn mã ứng với từng $N = 10, 20, \dots, 1000$. Sau đó, ta tạo 2 danh sách x và y để x lưu các giá trị của i, y lưu giá trị khoảng thời gian chạy. Thoát vòng lặp, ta đưa chúng về mảng thông qua phương thức array(), rồi in thời gian chạy của toàn bộ đoạn mã với từng giá trị N. Vì y là mảng chứa tất cả khoảng giá trị tương ứng với i nên ta in ra sum của y. Và vẽ hình.

```

27     total_time = 0.0
28
29     x=[]
30     y=[]
31     total = 0.0
32     for i in range(10, 1001, 10):
33         #print("Với N = ",i)
34         total += Check(i)
35         x.append(i)
36         y.append(total)
37
38     import matplotlib.pyplot as plt
39
40
41     x = np.array(x)
42     y = np.array(y)
43
44     plt.plot(*args: x,y)
45
46     plt.xlabel('X')
47     plt.ylabel('Y')
48
49     plt.title("Thoi gian chạy chương trình ứng với N")
50
51     plt.show()

```

Kết quả thu được:

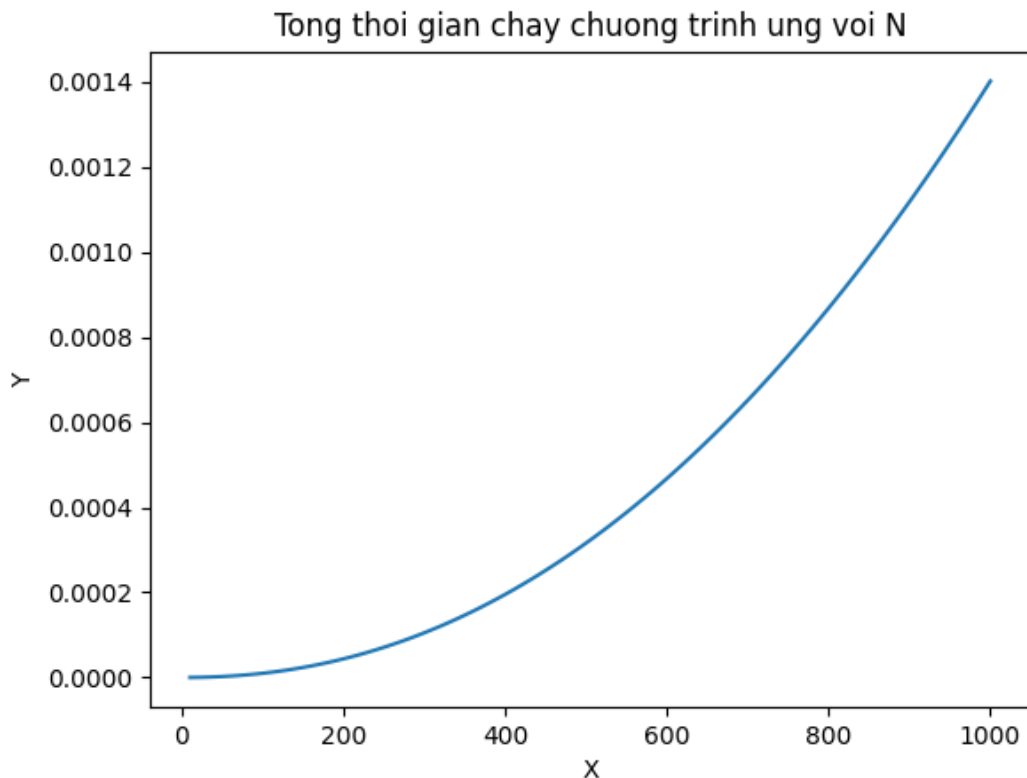


Nhận xét: Tùy vào mảng được tạo ra mà kết quả chạy thực nghiệm có sự tăng giảm hoặc giữ nguyên về thời gian tính toán ứng với mỗi N. Dù vậy, tổng thời gian chạy chương trình với toàn bộ N chỉ dao động trong khoảng từ 0.027 đến 0.031.

3.2. Kết quả lý thuyết.

Từ chứng minh trên ở mục 2, ta biết được rằng $T(N) = 3N \log_2(N) + 5$. Giờ ta sẽ thiết lập vòng lặp với từng giá trị N vào công thức trên. Lưu các giá trị N vào mảng x và các khoảng thời gian tương ứng với N là $T(N)$ vào mảng y. Ta vẽ đồ thị và đánh giá kết quả thu được từ lý thuyết (**File: bai3_2.py**)

Kết quả thu được:



```
C:\Users\PC-LENOVO\Downloads\Pycharm_code\venv\Scripts\python.exe C:\Users\PC-LENOVO\Downloads\Pycharm_code\alo.py
Tổng thời gian chạy là: 0.014021415670116319
Process finished with exit code 0
```

Nhận xét: Tổng thời gian chạy biến thiên gần tuyến tính với giá trị N đầu vào. Tổng thời gian thu được xấp xỉ 0.014.

3.3. Nhận xét chung.

Khoảng thời gian giữa thực nghiệm và lý thuyết có sự khác nhau rất rất nhỏ. Chúng chủ yếu đến từ việc thay đổi cách đếm phép gán và so sánh của thuật toán. Hơn nữa, kết quả lý thuyết là tính toán và chia cho 10^8 , mặc dù ta có thể đọc đâu đó máy tính xử lý được khoảng 10 mũ 8 lệnh trong 1 giây. Nhưng vì mỗi máy có tốc độ, phần xử lý bên trong khác nhau nên có thể dẫn đến sự sai khác nhỏ về tổng thời gian. Nhìn chung, nếu bỏ qua các tác nhân khách quan thì ta có thể thấy khoảng thời gian là như nhau và biến thiên tuyến tính.