**MỤC LỤC**

[A Hàm sinh xác suất 1](#_Toc376072308)

[I Tổng quan hàm sinh 1](#_Toc376072309)

[II Định nghĩa và thuộc tính 2](#_Toc376072310)

[III Tóm lược 4](#_Toc376072311)

[IV Tổng của các biến ngẫu nhiên độc lập 6](#_Toc376072312)

[V Tổng của số ngẫu nhiên độc lập 8](#_Toc376072313)

[VI Dùng hàm sinh để giải quyết quan hệ tái phát sinh 9](#_Toc376072314)

[VII Quá trình phân nhánh 11](#_Toc376072315)

[1. Định nghĩa 11](#_Toc376072316)

[2. Sự phát triển của quần thể 11](#_Toc376072317)

[3. Xác suất tuyệt chủng 14](#_Toc376072318)

[B Phân tích một số thuật toán sắp xếp 15](#_Toc376072319)

[I Định nghĩa độ phức tạp 15](#_Toc376072320)

[II Các thuật toán sắp xếp 17](#_Toc376072321)

[1. Thuật toán sắp xếp bằng phương pháp Quick Sort 17](#_Toc376072322)

[2. Thuật toán sắp xếp bằng phương pháp ShellSort 18](#_Toc376072323)

[3. Thuật toán sắp xếp bằng phương pháp nổi bọt 19](#_Toc376072324)

[Đánh giá 21](#_Toc376072325)

[Tài liệu tham khảo 22](#_Toc376072326)

# A Hàm sinh xác suất

## I Tổng quan hàm sinh

Hàm sinh được sử dụng phổ biến trong toán học, và đóng một vai trò quan trọng trong lý thuyết xác suất. Xét chuỗi {ai: i = 0; 1; 2; :::} thuộc tập số thực : là một trong những dãy số đã được phân tích trong một vài hàm sinh. Hàm sinh phổ dụng của chuỗi được định nghĩa như sau :

G(s) =

với những giá trị của các tham số s mà tổng hội tụ. Đối với một chuỗi được xét, tồn tại một bán kính hội tụ R( ≥ 0) như tổng hội tụ tuyệt đối nếu |s| < R và phân kỳ nếu |s| < R.

Với nhiều chuỗi được định nghĩa hoàn chỉnh, G(s) có thể được viết trong hình thức kín, và những số cá nhân trong chuỗi có thể được phục hồi hoặc mở rộng bằng cách dẫn xuất.

## II Định nghĩa và thuộc tính

Xem xét một số X riêng biệt lấy giá trị không âm. Ta viết

pk = P(X=k), k = 0,1,2,…

(nếu X là một số hữu hạn, ta chỉ cần thêm vào những xác suất bằng không ứng với các giá trị không xảy ra). Hàm sinh xác suất (PGF) của X được định nghĩa như sau

GX(s) = = E(sX)

Chú ý rằng GX(1) = 1, vì vậy dãy số hội tụ hoàn toàn cho |s| ≤ 1. Cũng như GX(0) = p0. Đối với một số phân phối phổ biến hơn, hàm sinh xác suất như sau :

*(i) Hằng số – nếu pc  = 1, pk = 0, k ≠ c*, ta có

GX(s) = E(sX) = sc

*(ii) Dãy Bernoulli – nếu p1 = p, p0 = 1 – p = q , pk = 0, k ≠ 0 hoặc 1*, ta có

GX(s) = E(sX) = q + ps

*(iii) Geometric – nếu pk = pqk-1, k = 1,2,…; q = 1-p*, ta có

*GX(s) = nếu |s| < q-1*

*(iv) Binomial – nếu X ̴ Bin(n,p)*, ta có

*GX(s) = (q + ps)n, (q = 1 - p)*

*(v) Poisson – nếu X ̴ Poisson(λ)*, ta có

*GX(s) = = e λ(s-1)*

*(vi) Negative binomial – nếu X ̴ NegBin(n,p)*, ta có

GX(s) = = nếu |s| < q-1 và p + q = 1

***Định lý khẳng định :***

Nếu X và Y có hàm sinh xác suất tương ứng GX và GY, ta có

*GX(s) = GY(s) với mọi s (a)*

*Nếu và chỉ nếu P(X = k) = P(Y = k) với k = 0,1,… (b)*

*Chú ý :* nếu và chỉ nếu X và Y có cùng phân phối xác suất.

**Chứng minh :** Ta cần chứng minh rằng (a) ứng dụng cho (b). Bán kính hội tụ của GX và GY thì ≥ 1, do đó, chúng có thể mở rộng chuỗi về gốc

GX(s) = P(X = k)

GY(s) = P(Y = k)

Nếu GX = GY, có 2 dãy số có hệ số giống nhau.

Ví dụ : Nếu X có hàm sinh xác suất với q = 1 – p, ta có thể kết luận rằng :

X ̴ Geometric(p)

Cho hàm A(s) với một hàm sinh xác suất của một số X, ta có thể được pk = P(X = k)

bằng cách triển khai A(s) trong chuỗi số trong s và đặt

pk = hệ số của sk;

hoặc cách khác A(s) k lần có liên quan đến s và đặt s = 0

Chúng ta có thể mở rộng định nghĩa của hàm sinh xác suất với hàm của X. Hàm sinh xác suất của Y = H(X) là

GY(s) = GH(X)(s) = E(sH(X)) =

Nếu H khá đơn giản, nó thể hiện GY(s) trong điều kiện của GX(s).

Ví dụ : Đặt Y = a + bX thì

GY(s) = E(sY) = E(sa+bX)

= saE[(sb)X] = saGX(sb)

## III Tóm lược

**Lý thuyết**

Lấy X là số đếm và số thứ r phát sinh của chính hàm sinh xác xuất GX(s) khi s = 1, thì

= E[X(X – 1)…(X – r + 1)]

***Chứng minh***

(s) = [GX(s)]

= []

=

(giả sử không mâu thuẫn khi đổi chỗ và ). Chuỗi hội tụ với |s| ≤ 1, vì thế :

(1) = E[X(X - 1)…(X – r + 1)], r ≥ 1

Ngoài ra :

(1) (hoặc (1)) = E(X)

Và

(1) (hoặc (1)) = E[X(X - 1)]

= E(X2) – E(X)

= Var(X) + [E(X)]2 – E(X)

Vì thế

Var(X) = (1) – [(1)2] + (1).

Ví dụ : Nếu X ̴ Poiss(λ), thì

GX(s) = e λ(s-1);

(s) = λeλ(s-1)

* E(X) = (1) = λe0 = λ.

(s) = λ2eλ(s-1)

* Var(X) = λ2 – λ2 + λ = λ

## IV Tổng của các biến ngẫu nhiên độc lập

**Lý thuyết**

Đặt X và Y là biến đếm độc lập, hàm sinh xác suất tương ứng với GX(s) và GY(s) và cho Z = X + Y.

GZ(s) = GX+Y(s) = GX(s)GY(s)

***Chứng minh***

GZ(s) = E(sZ) = E(sX+Y)

= E(sX)E(sY) (độc lập)

= GX(s)GY(s)

**Hệ quả**

Nếu X1, … ,Xn là các biến đếm độc lập, tương ứng với các hàm sinh xác suất GX1(s), … ,GXn(s) (và n là số nguyên cho trước),

GX1+ … + Xn(s) = GX1(s) … GXn(s)

**Ví dụ 1:**

Tìm phân phối của n số độc lập Xi, i = 1, … n, khi Xi ̴ Poisson(λi)

**Giải**

GXi(s) = e λi(s-1).

* GX1 + X2 + … + Xn(s) = λi(s-1)

= e(λ1+ … + λn)(s-1).

Đây là hàm sinh xác suất Poisson

**Ví dụ 2:**

Trong chuỗi số độc lập n Bernoulli

Đặt X = = số lượng đúng trong n lần. Tìm phân phối xác suất của X ?

**Giải:** Từ những lần thử độc lập nhau, I1, … In độc lập.

GX(s) = GI1, GI2, … GIn(s).

Nhưng GIi(s) = q + ps, I = 1, … n.

GX(s) = (q + ps)n =

P(X=x) = hệ số của sx trong GX(s)

= p+qn-x, x = 0, … , n

X ̴ Bin(n, p).

## V Tổng của số ngẫu nhiên độc lập

**Lý thuyết**

Cho N, X1, X2, … là các số đếm độc lập. Nếu tập {Xi} có phân phối giống nhau, với mỗi hàm sinh xác suất GX.

Có hàm sinh xác suất

SN = X1 + … + XN

GSN = GN(GX(s))

***Chứng minh :*** Ta có

(s) = E()

= (điều kiện với N)

=

=

= (hệ quả đã chứng minh trước đó)

= GN(GX(s)) (theo định nghĩa của GN)

**Hệ quả**

1) E(SN) = E(N) . E(X)

Chứng minh

[(s)] = [GN(GX(s))]

= . khi u = GX(s)

Đặt s = 1, (vì vậy u = GX(1) = 1), ta có

E(SN) = [] . [] = E(N) . E(X)

Tương tự ta có thể suy luận rằng

Var(SN) = E(N)Var(X) + Var(N) [E(X)]2 (3.19)

## VI Dùng hàm sinh để giải quyết quan hệ tái phát sinh

Thay vì sử dụng lý thuyết của quan hệ từ khi giải quyết các mối quan hệ tái phát sinh trong quá trình giải quyết bằng các điều kiện, người ta thường chuyển đổi các mối quan hệ trong phương trình cho hàm sinh, để được giải quyết mục tiêu theo các điều kiện ràng buộc thích hợp.

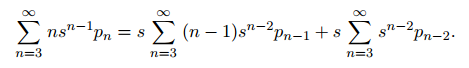
**Ví dụ 1:**

E:\q1.png

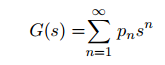
Bắt đầu với xác suất pn. Giải quyết vấn đề này bằng hàm sinh xác suất

**Giải**

Bằng nsn-1 và cộng tất cả các giá trị của n ta có được



Hàm sinh



(đây không là hàm sinh xác suất, khi {pn : n ≥ 1} không tạo ra một xác suất) đó là

E:\q4.png

Bây giờ phương trình này được giải quyết theo các điều kiện ràng buộc

G(0) = 0

Kết quả là

E:\q5.png

Giải G(s) theo chuỗi số trong s, và phân rã của sn

E:\q5.png

## VII Quá trình phân nhánh

### 1. Định nghĩa

Xét giả thuyết một cá thể sinh ra trong một điểm thời gian và chết đi trong quá trình sinh ra các cá thể mới. Ta cho rằng

*i) Kích thước của quần thể là độc lập, mỗi cá thể mang giá trị 0,1,2…*

*ii) Kích thước của quần thể được mô tả, các cá thể con trong quần thể, C, được xây dựng*

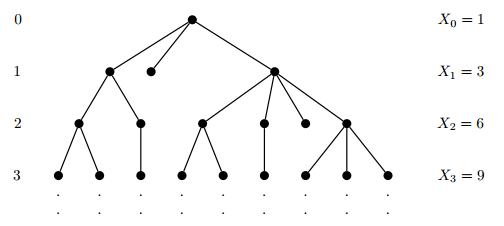
P(C = k) = pk, k = 0,1,2…

Sự phát triển cộng đồng quần thể trong quá trình thời gian, được gọi là quá trình phân nhánh : cung cấp một mô hình phát triển đơn giản, tạo ra mối quan hệ của các quần thể.

Đặt Xn = số cá thể sinh ra trong thời gian n

Sự phát triển của quần thể được mô tả bằng chuỗi X0, X1, X2,… Ta cho X0 = 1, bắt đầu với một cá thể xác định

Cây quần thể được thể hiện như sau :



Ta có thể dùng hàm sinh xác suất để duyệt quá trình này

### 2. Sự phát triển của quần thể

Đặt G(s) là hàm sinh xác suất của C

E:\q5.png

Và Gn(s) là hàm sinh xác suất của Xn

E:\q5.png

G0(s) = s, P(X0 = 1) = 1; P(X0 = x) = 0 với x ≠ 1 và G1(s) = G(s)

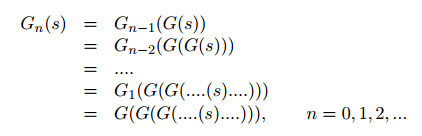
Xn = C1 + C2 + … + Cxn-1

Những điểm Ci là kích thước của quần thể được sản sinh bởi thế hệ thứ i thành viên của thế hệ thứ (n – 1)

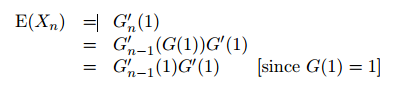
Vì thế, Xn là tổng của số ngẫu nhiên không phụ thuộc và được xây dựng giống nhau, ta có

Gn(s) = Gn-1(G(s)) với n = 2,3…

Và điều này cũng đúng cho n = 1. Lặp lại kết quả này, ta có

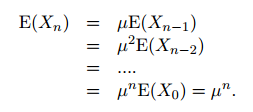


Gn là quần thể thứ n của G

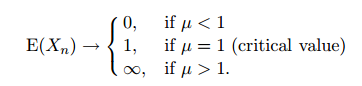


E(Xn) = E(Xn-1)ὴ

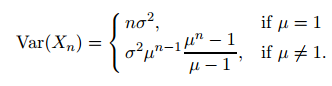
ὴ = E(C ) là ý nghĩa kích thước quần thể. Do đó



Có các suy luận



Tương tự



**Ví dụ 1 :**

Kiểm tra một phân nhánh C được định nghĩa phân phối hình học

pk = pqk, k = 0,1,2,…; 0 < p – 1 < 1, với p ≠ q

**Giải:** hàm sinh xác suất của C là

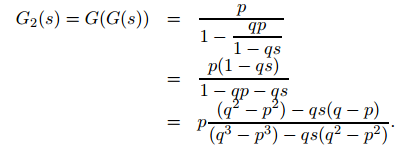
E:\q2.png

Ta cần giải quyết hàm Gn(s) = Gn-1(G(s))

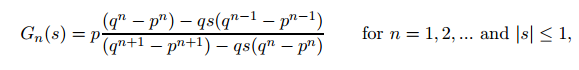
Đầu tiên, nếu |s| ≤ 1,

E:\q2.png

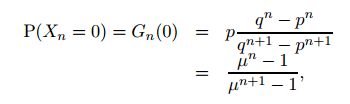
Sau đó



Ta phỏng đoán rằng



Và đây là kết quả phát triển từ n



### 3. Xác suất tuyệt chủng

Xác suất tiến trình tuyệt chủng thứ n được mô tả

En = P(Xn = 0)

en ≤ 1 và en ≤ en+1. {en} là một dãy đơn điệu

E:\q2.png

Gọi là xác suất tuyệt chủng cuối cùng

**Định lý 1**

e là gốc không âm nhỏ nhất của công thức x = G(x)

**Định lý 2**

e = 1 nếu và chỉ nếu ὴ ≤ 1

# B Phân tích một số thuật toán sắp xếp

## I Định nghĩa độ phức tạp

Thời gian mà [máy tính](http://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh) khi thực hiện một [thuật toán](http://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) không chỉ phụ thuộc vào bản thân thuật toán đó, ngoài ra còn tùy thuộc từng máy tính. Để đánh giá hiệu quả của một thuật toán, có thể xét số các phép tính phải thực hiện khi thực hiện thuật toán này. Thông thường số các phép tính được thực hiện phụ thuộc vào cỡ của bài toán, tức là độ lớn của đầu vào. Vì thế **độ phức tạp thuật toán** là một hàm phụ thuộc đầu vào. Tuy nhiên trong những ứng dụng thực tiễn, chúng ta không cần biết chính xác hàm này mà chỉ cần biết một ước lượng đủ tốt của chúng.

Để ước lượng độ phức tạp của một thuật toán ta thường dùng khái niệm bậc O-lớn và bậc Θ (bậc [Theta](http://vi.wikipedia.org/wiki/Theta)).

O(n)

Gọi n là độ lớn đầu vào. Tùy thuộc từng bài toán mà n có thể nhận những giá trị khác nhau. Chẳng hạn, bài toán tính [giai thừa](http://vi.wikipedia.org/wiki/Giai_th%E1%BB%ABa) thì n chính là số cần tính giai thừa. Nhiều bài toán [số trị](http://vi.wikipedia.org/w/index.php?title=Ph%C6%B0%C6%A1ng_ph%C3%A1p_s%E1%BB%91&action=edit&redlink=1), chẳng hạn tính [sai phân](http://vi.wikipedia.org/w/index.php?title=Sai_ph%C3%A2n&action=edit&redlink=1) thì n là số [chữ số có nghĩa](http://vi.wikipedia.org/w/index.php?title=Ch%E1%BB%AF_s%E1%BB%91_c%C3%B3_ngh%C4%A9a&action=edit&redlink=1) cần đạt được. Trong các phép tính đối với [ma trận](http://vi.wikipedia.org/wiki/Ma_tr%E1%BA%ADn) thì n là số hàng hoặc cột của ma trận.

Độ phức tạp của bài toán phụ thuộc vào n. Ở đây ta không chỉ đặc trưng độ phức tạp bởi số lượng phép tính, mà dùng một đại lượng tổng quát là *tài nguyên cần dùng* R(n). Đó có thể là số lượng phép tính (có thể tính cả số lần truy nhập bộ nhớ, hoặc ghi vào bộ nhớ); nhưng cũng có thể là thời gian thực hiện chương trình (*độ phức tạp về thời gian*) hoặc [dung lượng bộ nhớ](http://vi.wikipedia.org/w/index.php?title=Dung_l%C6%B0%E1%BB%A3ng_b%E1%BB%99_nh%E1%BB%9B&action=edit&redlink=1) cần phải cấp để chạy chương trình (*độ phức tạp về không gian*).

Xét quan hệ giữa tài nguyên và độ lớn đầu vào, nếu như tìm được [hằng số](http://vi.wikipedia.org/wiki/H%E1%BA%B1ng_s%E1%BB%91) C>0, C không phụ thuộc vào n, sao cho với n đủ lớn, các hàm  R(n), g(n) đều dương và

R(n)\leq C.g(n)

thì ta nói thuật toán có độ phức tạp cỡ O(g(n)).

Các độ phức tạp thường gặp đối với các thuật toán thông thường gồm có:

* Độ phức tạp hằng số, O(1). Số phép tính/thời gian chạy/dung lượng bộ nhớ không phụ thuộc vào độ lớn đầu vào. Chẳng hạn như các thao tác hệ thống: đóng, mở file.
* Độ phức tạp tuyến tính, O(n). Số phép tính/thời gian chạy/dung lượng bộ nhớ có xu hướng tỉ lệ thuận với độ lớn đầu vào. Chẳng hạn như tính tổng các phần tử của một mảng một chiều.
* Độ phức tạp đa thức, O(P(n)), với P là đa thức bậc cao (từ 2 trở lên). Chẳng hạn như các thao tác tính toán với mảng nhiều chiều (tính định thức ma trận).
* Độ phức tạp [logarit](http://vi.wikipedia.org/wiki/Logarit), O(\log n) (chú ý: bậc của nó thấp hơn so với O(n)). Chẳng hạn thuật toán Euclid để tìm [ước số chung lớn nhất](http://vi.wikipedia.org/wiki/%C6%AF%E1%BB%9Bc_s%E1%BB%91_chung_l%E1%BB%9Bn_nh%E1%BA%A5t).
* Độ phức tạp [hàm mũ](http://vi.wikipedia.org/wiki/H%C3%A0m_m%C5%A9), O(2^n). Trường hợp này bất lợi nhất và sẽ rất phi thực tế nếu thực hiện thuật toán với độ phức tạp này.

\Omega(f(n)),  \Theta(h(n)).

* Tương tự như với bậc O-lớn, nếu như tìm được các hằng số C, k_1, k_2 đều dương và không phụ thuộc vào n, sao cho với n đủ lớn, các hàm R(n), f(n) và h(n) đều dương và
* R(n)\geq C \cdot f(n)
* k_1\cdot h(n) \leq R(n) \leq k_2\cdot h(n)
* thì ta nói thuật toán có độ phức tạp cỡ lớn hơn \Omega(f(n)), và đúng bằng cỡ \Theta(h(n)).
* Như vậy nếu xét một cách chặt chẽ, kí hiệu Θ mới biểu thị độ phức tạp của thuật toán một cách chặt chẽ. Tuy nhiên qua một thời gian dài kí hiệu O(n) cũng đã được dùng phổ biến.

## II Các thuật toán sắp xếp

### 1. Thuật toán sắp xếp bằng phương pháp Quick Sort

***Cài đặt :***

void QuickSort(float a[],int l,int r)

{

int i,j;

int x;

i=l;

j=r;

x=a[(l+r)/2];

do

{

while (a[i]<x)i++;

while(a[j]>x)j--;

if(j<=j)

{

if(i<j)

{

int temp=a[i];

a[i]=a[j];

a[j]=temp;

}

i++;

j--;

}

}

while(i<j);

if(l<j)QuickSort(a,l,j);

if(i<r)QuickSort(a,i,r);

}

***Đánh giá thuật toán***

*Ta nhận thấy hiệu quả của thuật toán phụ thuộc vào việc chọn giá trị mốc (hay phần tử chốt).*

o   Trường hợp tốt nhất: mỗi lần phân hoạch ta đều chọn được phần tử median (phần tử lớn hơn hay bằng nửa số phần tử và nhỏ hơn hay bằng nửa số phần tử còn lại) làm mốc. Khi đó dãy được phân hoạch thành hai phần bằng nhau, và ta cần log2(n) lần phân hoạch thì sắp xếp xong. Ta cũng dễ nhận thấy trong mỗi lần phân hoạch ta cần duyệt qua n phần tử. Vậy độ phức tạp trong trường hợp tốt nhất thuộc O(nlog2(n)).

o   Trường hợp xấu nhất: mỗi lần phần hoạch ta chọn phải phần tử có giá trị cực đại hoặc cực tiểu làm mốc. Khi đó dãy bị phân hoạch thành hai phần không đều: một phần chỉ có một phần tử, phần còn lại có n-1 phần tử. Do đó, ta cần tới n lần phân hoạch mới sắp xếp xong. Vậy độ phức tạp trong trường hợp xấu nhất thuộc O(n2).

*Tổng kết lại, ta có độ phức tạp của Quick Sort như sau:*

·        *Trường hợp tốt nhất: O(nlog2n)*

·        *Trường hợp xấu nhất: O(n2)*

·        *Trường hợp trung bình: O(nlog2n)*

### 2. Thuật toán sắp xếp bằng phương pháp ShellSort

***Cài đặt***

void ShellSort(float a[],int n,int k)

{

int step,i,j;

int x;

for(step=1;step<=k;step++)

{

for(i=1;i<=n;i++)

{

x=a[i];

j=i-1;

while((x<a[j])&&(j>=1))

{

a[j+1]=a[j];

j=j-1;

}

a[j+1]=x;

}

}

}

***Đánh giá thuật toán***

o   Yếu tố quyết định chính của thuật toán chính là cách chọn khoảng cách h trong từng bước sắp xếp và số bước sắp xếp k. Nhưng phải thỏa 2 điều kiện sau: hi > hi + 1 và hk= 1.

o   Các phần tử h không được là bội số của nhau nhằm tránh hiện tượng mỗi bước sắp thứ tự phải tổ hợp 2 nhóm mà bước trước chúng không hề có ảnh hưởng lẫn nhau. Điều mong muốn là ảnh hưởng giữa các nhóm khác nhau càng nhiều càng tốt. **???**

o   Việc đánh giá giải thuật Shell sort hiện nay rất phức tạp, thậm chí 1 số chưa được chứng minh. Nhưng có 1 điều chắc chắn là hiệu quả của thuật toán phụ thuộc vào dãy các độ dài được chọn. Trong trường hợp chọn dãy độ dài theo công thức hi= (hi – 1- 1)/2 và hk= 1, k = log2- 1 thì giải thuật có độ phức tạp tương đương n1,2 << n2

### 3. Thuật toán sắp xếp bằng phương pháp nổi bọt

***Cài đặt :***

void BubbleSort(float a[],int n)

{

int temp;

for(int i=1;i<n;i++)

for(int j=n;j>i;j--)

if(a[j]<a[j-1])

{

temp=a[j];

a[j]=a[j-1];

a[j-1]=temp;

}

}

***Phân tích thuật toán***

o   Thấy ngay số phép so sánh là luôn không đổi, tức không phụ thuộc vào tình trạng ban đầu của dãy. Với i bất kỳ, ta luôn phải so sánh V[j] với V[j-1], mà j chạy từ n đến i+1, tức ta tốn n-i phép so sánh. Thêm nữa, i chạy từ 1 đến n-1. Vậy ta tính được số phép so sánh tổng cộng: ∑(n-i) với i chạy từ 1 đến n-1 = (n-1) + (n-2) + … + 1 = n(n-1)/2.

o   Số phép hoán vị (tương đương 3 phép gán) lại phụ thuộc vào tình trạng ban đầu của dãy. Cụ thể như sau:

§  Trường hợp tốt nhất: Dãy ban đầu đã có thứ tự. Ta thấy ngay ta không tốn một phép hoán vị nào.

§  Trường hợp xấu nhất: Dãy ban đầu có thứ tự ngược. Xét i bất kỳ, ta thấy rằng mỗi lần so sánh a[j] với a[j-1], ta đều phải thực hiện hoán vị. Điều này có nghĩa là số phép hoán vị bằng n(n-1)/2.

o   *Tổng kết lại, ta có độ phức tạp của Bubble Sort thuộc O(n2) trong mọi trường hợp*

# Tài liệu tham khảo

\_ Giáo trình bài giảng Probability Generating Functions (Chương 3)

\_ Generating Functions

\_ Giáo trình lý thuyết hàm sinh và ứng dụng

\_ Các bài phân tích thiết kế thuật toán

\_ The textbook *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne

\_ Introduction to Algorithms - Third Edition - Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein