Phần trắc nghiệm

1. A

2. C

3. H

4. D

5. A

6. D

7. C

Phần tự luận

1.

a) Lấy c = 4, k = 1, với mọi x >= 1 thì 4x2-5x+3 < 4x2

nên f(x) là O(x2)

b) Với mọi c hữu hạn ta tính giới hạn

= =

= =

Vậy ta chỉ cần chọn c > 2 thì khi x đủ lớn (chọn k đủ lớn) thì

f(x) sẽ nhỏ hơn cg(x) nên f(x) là O(x)

c) Tương tự câu b), ta tính giới hạn

= =

Vậy ta chỉ cần chọn c > 0.5 thì với x đủ lớn, giới hạn trên sẽ nhỏ hơn 1 tức

f(x) < cg(x), suy ra f(x) là O(x).

2.

a) Phải. Lấy c = 1, k = 10, với mọi x >= k thì 10 <= x

b) Phải. Lấy c = 4, k = 2, với mọi x >= k thì 3x + 7 < 4x2

c) Không. Vì với mọi c hữu hạn, k hữu hạn, = ∞

nên không tồn tại c để 2x2+2 <= cx

3. Thuật toán tìm giá trị nhỏ nhất trong dãy số tự nhiên N phần tử A0,A1,..,AN-1

min ← A[0]

i ← 1;

while i < N

if min > A[i]

then min ← A[i]

end if

i ← i+1

end while

return min

Có tổng cộng N-1 phép so sánh (so sánh min với A1 tới AN-1).

Có N phép gán cho i.

Trong trường hợp tốt nhất thì chỉ 1 phép gán cho min ( min ← A[0]) nếu A[0] là phần tử nhỏ nhất.

Tổng cộng có N+1 phép gán.

Trong trường hợp xấu nhất thì sẽ có N phép gán khi A[N-1] là phần tử nhỏ nhất.

Tổng cộng có 2N phép gán.

4. Tổng S có thể viết lại như sau

S

Mô tả lại thuật toán dưới dạng mã giải

k ← 2

gt ← 1

S ← 1

while k <= n

gt ← gt\*k

S ← S +1/gt

k ← k + 1

end while

return S

Đặt f(n) là hàm mô tả thời gian chạy thuật toán, đầu thuật toán ta có 3 phép gán.

Trong vòng while ta có 1 phép so sánh, 4 phép tính rồi gán (\*,+,/), vòng while kết thúc khi k = n+1, vậy vòng while đã thực hiện lặp n-1 lần rồi sau đó khi k = n + 1 thì thực hiện phép so sánh cuối cùng trước khi thoát khỏi vòng lặp

* Vòng while thực hiện (n-1)\*5 + 1 phép tính
* F(n) = 3 + 5(n-1) + 1 = 5n – 1

5.

Đặt f(n) là hàm mô tả thời gian chạy thuật toán.

Lúc đầu có 2 phép gán cho sum và i.

Trong vòng for có tổng cộng 4 phép tính và lệnh.

Vòng for kết thúc khi i = n, khi đó 1 phép so sánh cuối cùng được thực hiện i < n rồi thoát khỏi vòng lặp.

Vòng for thực hiện lặp n lần rồi sau đó thực hiện phép so sánh cuối cùng i với n,

* Tổng cộng vòng for thực hiện 4n+1 phép tính
* f(n) = 4n + 3

6.

a) Số phép gán 2n2 + 2n + 1

Số phép so sánh n2 + 2n + 1

* thuật toán có chi phí giới hạn trên big-O là O(n2)

b) Tính trong trường hợp xấu nhất khi trong mảng a không chứa k

Số phép gán n + 1

Số phép so sánh n

* thuật toán có chi phí giới hạn trên big-O là O(n)

c) Số phép gán

Số phép so sánh

* thuật toán có chi phí giới hạn trên big-O là O(n2)

7.

a) g(n) = n\*log2n

b) g(n) = n

c) g(n) = n2

d) g(n) = log2n

8.

a) g(n) = 1

b) g(n) = n

c) g(n) = n

d) g(n) = n2

e) g(n) = logn

f) g(n) = n\*logn

9.

Câu a) đúng vì 2n+1 = 2n.21 nên có big-O là O(2n)

Câu b) sai vì 22n = 4n nên có big-O là O(4n)  
10.

a) A =< 31, 41, 59, 26, 41, 58 >

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 31 | 41 | 59 | 26 | 41 | 58 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 31 | 41 | 59 | 26 | 41 | 58 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 31 | 41 | 59 | 26 | 41 | 58 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 26 | 31 | 41 | 59 | 41 | 58 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 26 | 31 | 41 | 41 | 59 | 58 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 26 | 31 | 41 | 41 | 58 | 59 |

b) B =< 100, 77, 49, 29, 51, 100 >

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 100 | 77 | 49 | 29 | 51 | 100 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | | 2 | | 3 | | 4 | 5 | 6 |
| 77 | | 100 | | 49 | | 29 | 51 | 100 |
| 1 | | 2 | | 3 | | 4 | | 5 | 6 |
| 49 | | 77 | | 100 | | 29 | | 51 | 100 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 29 | 49 | 77 | 100 | 51 | 100 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 29 | 49 | 51 | 77 | 100 | 100 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 29 | 49 | 51 | 77 | 100 | 100 |

c) C =< 14, 23, 10, 9, 9, 10 >

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 14 | 23 | 10 | 9 | 9 | 10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 14 | 23 | 10 | 9 | 9 | 10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 10 | 14 | 23 | 9 | 9 | 10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 9 | 10 | 14 | 23 | 9 | 10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 9 | 9 | 10 | 14 | 23 | 10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 9 | 9 | 10 | 10 | 14 | 23 |

11.

for i from 2 to n do

temp ← A[i]

j ← i-1

for j to 0 do

if A[j] > temp then

A[j+1]=A[j]

else

break

end for

A[j+1]=temp

end for

12.

Selection sort viết bằng C

void SelectionSort(int \*A, int N)

{

for(int i = 0; i < N - 1; i++)

{

int min = i;

for(int j=i+1;j<N;j++)

{

if(A[j]<A[min])

min=j;

}

int temp=A[i];

A[i]=A[min];

A[min]=temp;

}

}

a) Vì khi chọn đến phần tử gần cuối (phần tử lớn thứ nhì mảng), thì phần này sẽ được xếp vào đúng vị trí và phần tử lớn nhất mảng cũng sẽ vào đúng vị trí cuối mảng.

b) Trường hợp xấu nhất là khi mảng được sắp xếp giảm dần, khi đó tất cả các phép gán min = j đều được thực hiện, nhìn chung thuật toán chạy với độ phức tạp O(n2)

Trường hợp tốt nhất là khi mảng đã được sắp xếp tăng dần, không xảy ra phép gán min = j. Nhưng khi đó trong hàm vẫn xảy ra đầy đủ các phép so sánh nên nhìn chung thuật toán vẫn chạy độ phức tạp O(n2) nhưng nhanh hơn trong trường hợp xấu nhất 1+2+…+N-1 = = phép tính.

c) Nhìn chung khi mảng được cho ngẫu nhiên thì Insertion sort chạy nhanh hơn Selection sort.

Bởi dù Insertion sort phải dịch các phần tử thứ lớn hơn phần tử đang xét sang phải rất nhiều lần nhưng khi tìm được vị trí thích hợp thì quá trình này sẽ dừng lại, hạn chế được số phép so sánh lãng phí, còn Selection sort thì số phép so sánh là không đổi, đều phải so sánh phần tử đang xét với tất cả phần tử còn lại trong mảng.

Đặc biệt trong trường hợp mảng có thiên hướng được xếp tăng dần thì insertion sort tỏ ra nhanh hơn rất nhiều do có thời gian chạy là gần như là O(n), với mỗi phần tử chỉ so sánh với số lần rất ít (thậm chí 1 lần).

Trường hợp mà selection sort có vẻ như đã đuổi kịp insertion sort là trường hợp xấu nhất của cả 2 thuật toán, mảng được sắp xếp giảm dần. Khi đó với insertion sort thì cứ mỗi phép so sánh là 1 phép dịch phải của phần tử đang so sánh, cộng thêm số phép so sánh sẽ là tối đa. Tương ứng với đó trong selection sort khi số phép so sánh vốn không đổi của mình, thuật toán này còn phải gánh thêm lệnh min=j sau mỗi lần so sánh nên dường như không có sự khác biệt về thời gian tính toán giữa 2 thuật toán trong trường hợp này.