

- Nghiệm của phương trình đệ quy $T(1) = 1$ và $T(n) = 3T(n/2) + n^2$ là
-> $O(n^2)$ "Ô n bình phương"
- Trong việc vận dụng kỹ thuật nhánh cận để giải bài toán tìm MAX, tại sao chúng ta lại ưu tiên phân nhánh cho nút có cận trên lớn hơn?
-> Vì hy vọng ở hướng đó sẽ có phương án tối ưu.
- Hàm không âm $f(n)$ được gọi là một hàm nhân nếu
-> $f(n.m) = f(n).f(m)$ với mọi n, m

4. Xét hàm PushDown

```
void PushDown(recordtype a[], int first,int last)
{ int r = first;
while (r <= (last-1)/2)
    if (last == 2*r+1) {
        if (a[r].key > a[last].key) Swap(&a[r],&a[last]);
        r = last;
    } else if ((a[r].key>a[2*r+1].key) && (a[2*r+1].key<=a[2*r+2].key)) {
        Swap(&a[r],&a[2*r+1]);
        r = 2*r+1 ;
    } else if ((a[r].key>a[2*r+2].key) && (a[2*r+2].key<a[2*r+1].key)) {
        Swap(&a[r],&a[2*r+2]);
        r = 2*r+2 ;
    } else
        r = last;
}
```

Độ phức tạp của PushDown($a, 0, n-1$) là

-> $O(\log n)$

- Đối với hàm nhân 2 số nguyên lớn. Nếu dùng thuật toán chia để trị, tổng hợp theo công thức

$$XY = AC \cdot 10^n + (AD+BC) \cdot 10^{n/2} + BD$$

thì độ phức tạp thời gian của thuật toán là:

-> $O(n^2)$

6. Trong khi đánh giá giải thuật QuickSort một mảng gồm n phần tử có giá trị khóa đôi một khác nhau, trường hợp tốt nhất là trường hợp nào?

-> Mảng được phân hoạch cân bằng

7. Nếu $T_1(n)$ và $T_2(n)$ là thời gian thực hiện của hai đoạn chương trình P1 và P2; và $T_1(n)$ là $O(f(n))$, $T_2(n)$ là $O(g(n))$ thì độ phức tạp của đoạn hai chương trình đó nối tiếp nhau là

-> $O(\max(f(n), g(n)))$

8. Độ phức tạp của thuật toán HeapSort (sắp xếp vun đống) để sắp xếp danh sách có n phần tử là

-> $O(n \log n)$

9. Độ phức tạp của thuật toán MergeSort (sắp xếp trộn) để sắp xếp danh sách có n phần tử là

-> $O(n \log n)$

10. Xét chương trình hàm phân hoạch được cho ở phía dưới. Hàm này sẽ phân hoạch mảng $a[i],..,a[j]$ thành 2 mảng con có tính chất như sau:

```
int Partition(recordtype a[], int i,int j, keytype pivot) {  
    int L,R;  
    /*1*/ L = i;  
    /*2*/ R = j;  
    /*3*/ while (L <= R) {  
        /*4*/ while (a[L].key <= pivot) L++;  
        /*5*/ while (a[R].key > pivot) R--;  
        /*6*/ if (L < R) Swap(a[L],a[R]);  
    }  
}
```

```
/*7*/ return L; /*Tra ve diem phan hoach*/  
}
```

-> Mảng bên trái có khoá nhỏ hơn hoặc bằng pivot và mảng con bên phải có khoá lớn hơn pivot

11. Để tính độ phức tạp của một chương trình không gọi chương trình con, chúng ta sử dụng:

-> Quy tắc cộng, quy tắc nhân và quy tắc tổng quát.

12. Độ phức tạp của giải thuật là

-> Là tỷ suất tăng của hàm thời gian.

13. Xét hàm phân hoạch

```
int Partition(recordtype a[],int i,int j, keytype pivot)  
{ int L = i,R = j;  
    while (L <= R) {  
        while (a[L].key < pivot) L++;  
        while (a[R].key >= pivot) R--;  
        if (L < R) Swap(&a[L],&a[R]);  
    }  
    return L;  
}
```

Độ phức tạp của Partition(a,0,n-1,pivot) là

-> O(n)

14. Kỹ thuật tham ăn là :

-> Kỹ thuật xây dựng một phương án, trong đó mỗi thành phần của phương án được lựa chọn một cách tối ưu.

15. Nghiệm của phương trình đệ quy với $T(1) = 1$ và $T(n) = 2T(n/2) + 1$ là
-> $O(n)$

16. Để viết hàm tính x^n (x lũy thừa n) theo kỹ thuật chia để trị, dùng công thức sau:

$$x^n = x \text{ nếu } n=1$$

$$x^n = x^{n/2} * x^{n/2} \text{ nếu } n > 1 \text{ và là số chẵn}$$

$$x^n = x^{n/2} * x^{n/2} * x \text{ nếu } n > 1 \text{ và là số lẻ}$$

Độ phức tạp của hàm tính x^n theo thuật toán đệ quy là:

-> $O(n)$

17. Trong kỹ thuật nhánh cận để giải bài toán tìm MAX, một nút sẽ được cắt tỉa nếu

-> Cận trên của nút đó nhỏ hơn hoặc bằng giá lớn nhất tạm thời

18. Đối với hàm nhân 2 số nguyên lớn. Dùng thuật toán chia để trị, tổng hợp theo công thức

$$XY = AC * 10^n + [(A-B)(D-C)+AC+BD] * 10^{n/2} + BD.$$

Nếu không sử dụng các biến $m1, m2$ và $m3$ để lưu kết quả gọi đệ quy tính các tích tương ứng $AC, (A-B)(D-C)$ và BD thì độ phức tạp thời gian của thuật toán là:

-> $O(n^{\log 5})$

19. Trong kỹ thuật nhánh cận để giải bài toán tìm MIN, chúng ta ưu tiên phân nhánh cho nút nào trước?

-> Nút có cận dưới nhỏ nhất trong tất cả các nút cùng cấp trên cây

20. Trong phương trình đệ quy tổng quát, với trường hợp $a < d(b)$, nghiệm của phương trình là $O(n^{\log_b d(b)})$, muốn cải tiến giải thuật chúng ta phải làm gì ?

-> Cải tiến việc phân chia bài toán và tổng hợp lời giải

21. Đối với hàm nhân 2 số nguyên lớn. Nếu dùng thuật toán nhân thông thường (như đã biết trong trường tiểu học) thì độ phức tạp thời gian của thuật toán là:

-> $O(n^2)$

22. Thế nào là một Min-heap?

-> Min-heap là cây nhị phân mà giá trị tại mỗi nút (khác nút lá) đều nhỏ hơn hoặc bằng giá trị của các con của nó.

23. Xét hàm tìm chốt được viết như sau:

```
int FindPivot(recordtype a[], int i,int j){  
    keytype firstkey;  
    int k ;  
    /*1*/ k = i+1;  
    /*2*/ firstkey = a[i].key;  
    /*3*/ while ( (k <= j) && (a[k].key == firstkey) ) k++;  
    /*4*/ if (k > j) return -1;  
    /*5*/ if (a[k].key<firstkey) return k;  
    /*6*/ return i;  
}
```

Giả sử mảng có chốt thì hàm này sẽ trả về

-> Chỉ số của phần tử có khoá nhỏ nhất trong hai phần tử có khoá khác nhau đầu tiên

24. Yếu tố tiên quyết trong kỹ thuật quy hoạch động là :

-> Xây dựng được công thức truy hồi.

25. Đối với hàm nhân 2 số nguyên lớn. Nếu dùng thuật toán chia để trị, tổng hợp theo công thức

$$XY = AC \cdot 10^n + [(A-B)(D-C) + AC+BD] \cdot 10^{n/2} + BD$$

thì độ phức tạp thời gian của thuật toán là:

$$\rightarrow O(n^{\log 3})$$

26. Xét hàm phân hoạch mảng được viết như sau:

```
int Partition (recordtype a[], int i, int j){  
    int pivot = a[i].key;      // "chốt"  
    int L = i+1;  
    int R = j;  
    while(L<=R){  
        while(L<=R && a[L].key<=pivot) L++;  
        while(L<=R && a[R].key>=pivot) R--;  
        if(L<R) Swap(&a[L], &a[R]);  
    }  
    Swap(&a[R], &a[i]);  
    return R;  
}
```

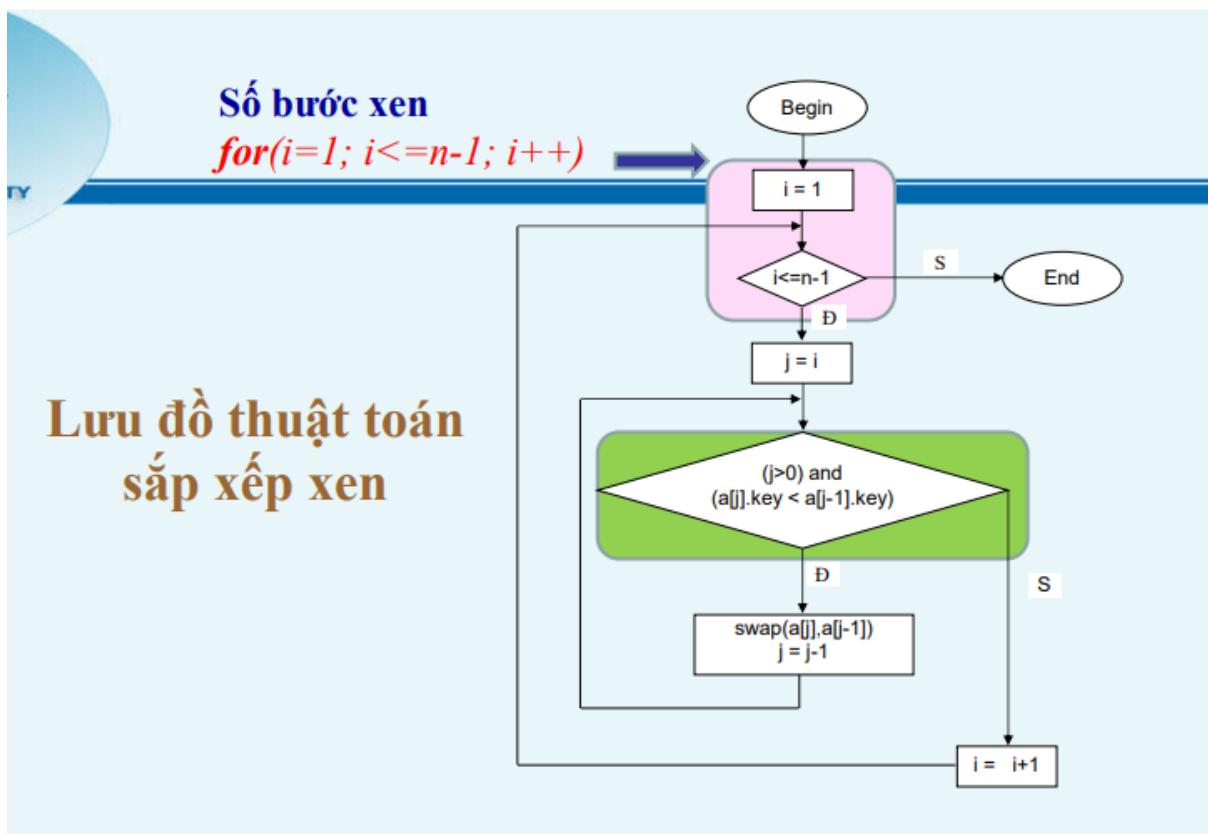
Sau khi thực hiện phân hoạch, phần tử "chốt" sẽ nằm ở vị trí nào trong mảng

-> Ở vị trí giữa mảng

27. Xác định độ phức tạp của đoạn chương trình sau

-> $O(n^3)$ "Ô n lập phương"

28. Hãy cho biết lưu đồ sau là của giải thuật nào?



-> Sắp xếp xen

29. Trong kỹ thuật nhánh cận giải bài toán tìm MAX, chúng ta ưu tiên phân nhánh cho nút nào trước?

-> Nút có cận trên lớn nhất trong tất cả các nút cùng cấp trên cây

30. Phương trình đệ quy của hàm QuickSort trong trường hợp phân hoạch đều là

$T(1) = 1$ và

-> $T(n) = 2T(n/2) + n$

31. Hãy cho biết đoạn chương trình sau được viết theo giải thuật nào?

```
void InsertionSort(recordtype a[], int n){  
    int i,j;  
{1}    for(i=1; i<=n-1; i++){  
        j=i;  
        {2}        while ((j>0)&&(a[j].key<a[j-1].key)) {  
            {3}            Swap(a[j],a[j-1]);  
            {4}            j--;  
        }  
    }  
}
```

$$\rightarrow T(n) = O(n^2)$$

20

-> Sắp xếp xen

32. Khi giải bài toán đường đi của người giao hàng bằng kỹ thuật tham ăn, xây dựng chu trình từ những cạnh có độ dài nhỏ nhất. Một cạnh sẽ không được chọn nếu

-> Tạo thành đỉnh cấp 3 hoặc tạo thành chu trình thiếu

33. Khi giải bài toán cái ba lô bằng kỹ thuật nhánh cận, công thức tính cận trên của mỗi nút trên cây là

-> Tổng giá trị của các vật được chọn ứng với nút đó + trọng lượng còn lại của ba lô * đơn giá của vật kế tiếp

34. Nghiệm của phương trình đệ quy với $T(1) = 1$ và $T(n) = 4T(n/2) + n^2$ là

-> $O(\log n^2)$ "Ô logarit cơ số 2 của n nhân n bình phương"

35. Nếu giải bài toán tìm đường đi của người giao hàng với n thành phố bằng phương pháp "vết cạn" thì chúng ta phải xét bao nhiêu phương án?

-> $(n-1)!/2$

36. Nếu $T_1(n)$ và $T_2(n)$ là thời gian thực hiện của hai đoạn chương trình P_1 và P_2 , đồng thời $T_1(n)$ là $O(f(n))$, $T_2(n)$ là $O(g(n))$ thì độ phức tạp của hai đoạn chương trình đó lồng nhau là

-> $O(f(n) * g(n))$

37. Khi giải bài toán đường đi của người giao hàng bằng kỹ thuật tham ăn, xây dựng chu trình từ những cạnh có độ dài nhỏ nhất. Một cạnh sẽ được chọn nếu thỏa mãn 2 điều kiện

-> Không tạo thành đỉnh cấp 3 và không tạo thành chu trình thiếp

38. Xét hàm tìm chốt được viết như sau:

```
int FindPivot(recordtype a[], int i,int j){  
    keytype firstkey;  
    int k ;  
/*1*/ k = i+1;  
/*2*/ firstkey = a[i].key;  
/*3*/ while ( (k <= j) && (a[k].key == firstkey) ) k++;  
/*4*/ if (k > j) return -1;  
/*5*/ if (a[k].key>firstkey) return k;  
/*6*/ return i;  
}
```

Giả sử mảng có chốt thì hàm này sẽ trả về

-> Chỉ số của phần tử có khoá lớn nhất trong hai phần tử có khoá khác nhau đầu tiên

39. Cho công thức tính số tổ hợp chập k của n đệ quy như sau:

$C_n^k = 1$ nếu $k=0$ hoặc $k=n$

$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ nếu $0 < k < n$

Độ phức tạp của thuật toán đệ quy tính C_n^k là

-> $O(2n)$ "Ô 2 mũ n"

40. Tại sao khi đánh giá giải thuật, người ta lại dùng đại lượng độ phức tạp mà không dùng thời gian thực hiện của chương trình?

- Vì thời gian thực hiện của chương trình phụ thuộc vào tốc độ của máy tính.
- Vì độ phức tạp của giải thuật chỉ phụ thuộc vào kích thước và tính chất của dữ liệu vào.
- Vì tương quan giữa các hàm thời gian thay đổi theo kích thước dữ liệu vào.

-> Tất cả các phương án khác liệt kê ở đây đều đúng.

41. Kỹ thuật quy hoạch động nhằm giải quyết vấn đề:

-> Một số bài toán con phải giải nhiều lần trong một số giải thuật đệ quy.

42. Biết rằng hàm max có độ phức tạp $O(1) = 1$. Hãy xác định độ phức tạp của hàm `phan_tu_ln` để tìm phần tử lớn nhất trong mảng `a` có `n` phần tử. Chương trình hàm `phan_tu_ln` như sau:

*không có ảnh

-> $O(n)$

43. Độ phức tạp của thuật toán tìm kiếm trên cây nhị phân có `n` nút là:

-> $O(n)$

44. Phương pháp xác định nghiệm của phương trình đệ quy thuộc dạng phương trình tổng quát nhưng hàm $d(n)$ không phải là một hàm nhân:

-> Phải tính trực tiếp nghiệm riêng rồi so sánh với nghiệm thuần nhất để chọn nghiệm lớn hơn.

45. Thuật toán tìm kiếm tuần tự một phần tử trong một mảng có `n` phần tử có độ phức tạp là:

-> $O(n)$

46. Trong việc vận dụng kỹ thuật nhánh cận để giải bài toán tìm MIN, chúng ta sẽ cắt tỉa một nút nếu :

-> Cận dưới của nút lớn hơn hoặc bằng giá nhỏ nhất tạm thời.

47. Đoạn chương trình sau được viết theo giải thuật nào?

```
void BubbleSort(recordtype a[ ], int n) {  
    int i,j;  
    {1}   for(i= 0; i<= n-2; i++)  
    {2}     for(j=n-1;j>=i+1; j--) → T(n) = O(n2)  
    {3}       if (a[j].key < a[j-1].key)  
    {4}         Swap(a[j],a[j-1]);  
}
```

-> Sắp xếp nổi bọt

48. Phương trình đệ quy của hàm QuickSort trong trường hợp phân hoạch lệch là

$T(1) = 1$ và

-> $T(n) = T(n-1) + T(1) + n$

49. Trong phương trình đệ quy tổng quát, với trường hợp $a > d(b)$, nghiệm của phương trình là $O(n^{\log_b a})$, muốn cải tiến giải thuật chúng ta phải làm gì ?

-> Giảm a

50. Độ phức tạp của thuật toán tìm kiếm trên cây tìm kiếm nhị phân có n nút là:

-> $O(\log n)$

51. Cho phương trình đệ quy với $T(1) = C_1$ và $T(n) = 2T(n-1) + C_2$
Nghiệm của phương trình đã cho là:

-> $O(2n)$ "Ô 2 mũ n"

52. Cho công thức tính số tổ hợp chập k của n đệ quy như sau:

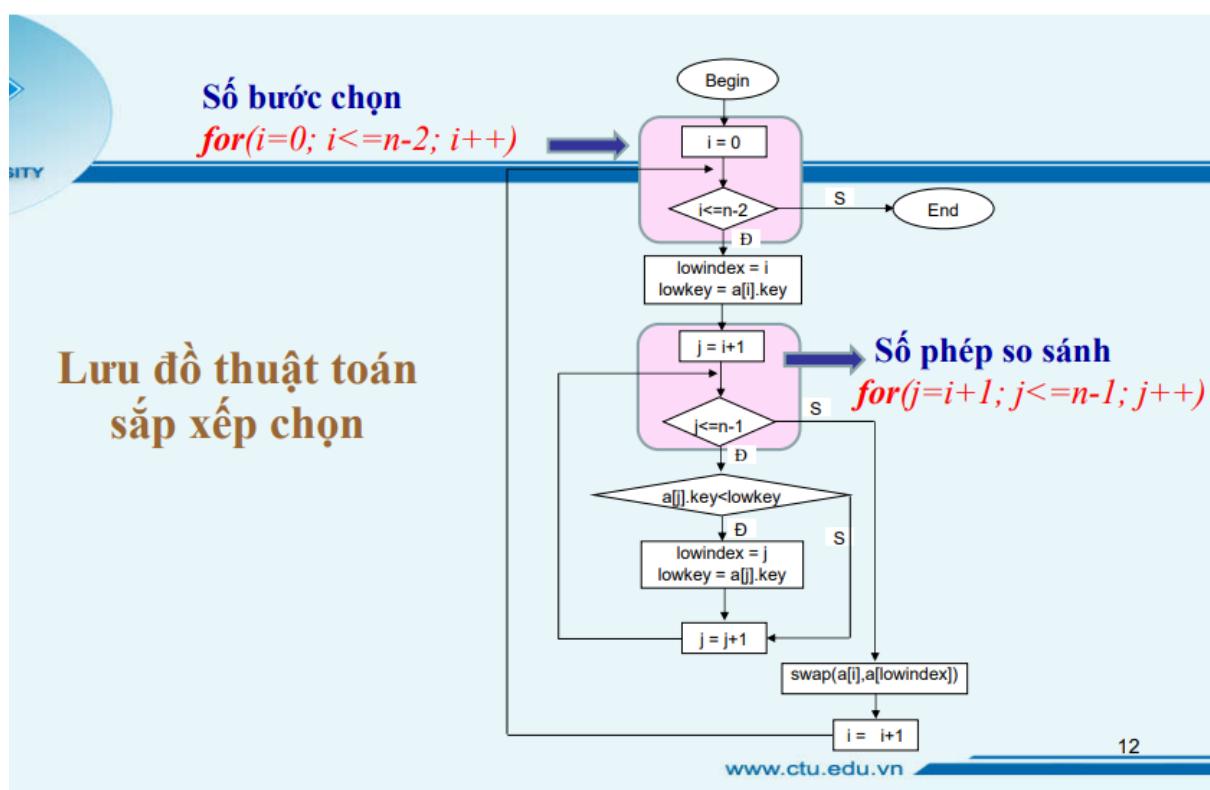
$C_n^k = 1$ nếu $k=0$ hoặc $k=n$

$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ nếu $0 < k < n$

Độ phức tạp của thuật toán tính C_n^k theo kỹ thuật quy hoạch động là

-> $O(n^2)$ "Ô n bình phương"

53. Hãy cho biết lưu đồ sau là của giải thuật nào?



-> Sắp xếp chọn

54. Thế nào là một Max-heap?

-> Max-heap là cây nhị phân mà giá trị tại mỗi nút (khác nút lá) đều lớn hơn hoặc bằng giá trị của các con của nó.

55. Mối quan hệ giữa ngôn ngữ lập trình và việc phân tích giải thuật như thế nào?

-> Khi phân tích giải thuật, không nhất thiết phải dựa vào ngôn ngữ lập trình.

56. Xét về thời gian thực hiện của tìm kiếm nhị phân so với tìm kiếm tuần tự thì

-> Tìm kiếm nhị phân nhanh hơn nếu dữ liệu đã có thứ tự.

57. Trong khi phân tích giải thuật đệ quy để tính số tổ hợp chập k của n (C_n^k), trường hợp xấu nhất là :

-> $k <> 0$ và $k <> n$

58. Xác định số lần lặp của lệnh: `for(i=1; i<=n; i = 2*i)`

-> $\log_2 n$ (logarit cơ số 2 của n) lần

59. Xét hàm phân hoạch mảng được viết như sau:

```
int Partition (recordtype a[], int i, int j){  
    int pivot = a[i].key;      // "chốt"  
    int L = i+1;  
    int R = j;  
    while(L<=R){  
        while(L<=R && a[L].key<=pivot) L++;  
        while(L<=R && a[R].key>=pivot) R--;
```

```
    if(L < R) Swap(&a[L], &a[R]);  
}  
Swap(&a[R], &a[i]);  
return R;  
}
```

Hàm này sẽ phân hoạch mảng thành

-> Mảng con bên trái có khóa nhỏ hơn hoặc bằng "chốt" và mảng con bên phải có khóa lớn hơn hoặc bằng "chốt".

60. Khi đánh giá một giải thuật đệ quy, chúng ta phải :

-> Thành lập phương trình đệ quy và giải phương trình đệ quy đó.

61. Tại sao cần phải phân tích đánh giá giải thuật?

-> Để lựa chọn giải thuật tốt nhất hoặc cải tiến giải thuật.

62. Thuật toán tìm kiếm nhị phân một phần tử trong một mảng có n phần tử có độ phức tạp là:

-> $O(\log n)$