**Bài 3:** Lập trình cho giải thuật chia để trị giải các bài toán:

***1. Tìm kiếm phần tử trên mảng được sắp.***

**+** *Phát biểu bài toán*: cho mảng n phần tử và một giá trị x bất kì. Kiểm tra xem x có trong dãy không?

**+** *Ý tưởng*: xét dãy có thứ tự tăng, so sánh giá trị x với phần tử giữa của dãy tìm kiếm. Dựa vào giá trị này sẽ quyết định giới hạn tìm kiếm ở bước kế tiếp.

**+** *Thuật toán*:

BinarySearch( a, x, left, right) ≡

if(L=L) return (x=? L: -1)

else

middle= (L+ R)/2;

if( x== ) return (M);

else

if( x<) BinarySearch( a, x, L, M)

else BinarySearch( a, x, M+1, R)

endif;

endif;

End.

+ *Đánh giá độ phức tạp thuật toán:*

Trường hợp tốt nhất: x= số lần so sánh là 1.

Trường hợp xấu nhất: x nằm ở cuối mảng hoặc không tìm thấy số lần so sánh là O(log n).

Trường hợp trung bình: O(log n/2).

* Độ phức tạp thuật toán là O(log n).

+ *Đánh giá sự phức tạp theo thời gian bằng biểu đồ sau:*

1. ***Tìm max, min của dãy số:***

+ *Phát biểu bài toán:* cho mảng n phần tử. Tìm phần tử lớn nhất và nhỏ nhất trong dãy.

+ *Ý tưởng:* Tại mỗi bước chia đôi doạn cần tìm rồi tìm min, max của từng đoạn sau đó tổng hợp lại kết quả. Nếu đoạn chia chỉ có 1 phần tử thì min=max bằng giá trị phần tử đó.

+ *Thuật toán:*

MinMax(a, L, R, Min, Max) ≡

if(L==R) Min=Max=a l;

else

{

M=(L+R)/2;

MinMax(a, L, M, Min1, Max1);

MinMax(a, M+1, R, Min2, Max2);

if( Min1< Min2) Min= Min1;

else Min= Min2;

if( Max1> Max2) Max= Max1;

else Max= Max2;

}

End.

+ *Đánh giá độ phức tạp thật toán:* gọi T(n) là số phép toán cần thực hiện

2T(n/2) + 2 khi n>2

T(n)= 1 khi n=2

* Độ phức tạp thuật toán là O(n).

+ *Đánh giá sự phức tạp theo thời gian bằng biểu đồ sau:*