NỘI DUNG THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

Chương 1: TỔNG QUAN VỀ CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

Nội dung:

1. C++: Mảng và vector

- Các thao thác cơ bản trên mảng và vector

Bài toán 1.1.1. Cho một mảng đã sắp xếp, kiểm tra trong mảng có tồn tại phần 2 phần tử các nhau độ dài d.

Sử dụng kỹ thuật 2 con trỏ

Ví dụ: A = [1, 2, 3, 7, 8].

d = 6 thì tồn tại cặp (1,8) thỏa đề

```
left = 0
for right = 0 to n-1 do
  while (A[right]-A[left] > d) do left++
  if (A[right] – A[left] == d) then return {left, right}
return {-1, -1}
```

1.2. Java: Arrays và ArrayList

- Các thao tác, thuộc tính và phương thức cơ bản của Arrays và ArrayList

1.3. C#: Array

Các thao tác, thuộc tính và phương thức cơ bản của Arrays

1.4. Javascript: Array

Các thao tác, thuộc tính và phương thức cơ bản của Arrays

1.5. Python List

Các thao tác, thuộc tính và phương thức cơ bản của List

Chương 2: Tìm kiếm và sắp xếp

1. Tìm kiếm

2.1. Tìm kiếm tuần tự

2.2. Tìm kiếm nhị phân

2.3. Bài tập

Bài toán 2.3.1. Cho mảng chưa sắp xếp, tìm kiếm cặp phần tử có trị tuyệt đối của hiệu đúng bằng x.

Bài toán 2.3.2. Cho mảng đã sắp xếp và một giá trị x, tìm kiếm k phần tử gần với x nhất trong mảng a. Nếu x có trong mảng thì bỏ qua giá trị x.

Input: k = 4, x = 35, arr[] = {12, 16, 22, 30, 35, 39, 42, 45, 48, 50, 53, 55, 56}

output: 30, 39, 42, 45

Bài toán 2.3.3. Cho 2 mảng đã sắp xếp A, B có n phần tử, tìm trung vị của mảng gộp 2 mảng A, B.

2. Sắp xếp

2.1. Các thuật toán sắp xếp $O(n^2)$

2.1.1. Thuật toán interchange sort – đổi chỗ trực tiếp

2.1.2. Thuật toán sắp xếp chọn - Selection sort

1. First we find the smallest element and swap it with the first element. This way we get the smallest element at its correct position.
2. Then we find the smallest among remaining elements (or second smallest) and swap it with the second element.
3. We keep doing this until we get all elements moved to correct position.

2.1.3. Thuật toán sắp xếp nổi bọt – Bubble sort

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity are quite high.

- We sort the array using multiple passes. After the first pass, the maximum element goes to end (its correct position). Same way, after second pass, the second largest element goes to second last position and so on.
- In every pass, we process only those elements that have already not moved to correct position. After k passes, the largest k elements must have been moved to the last k positions.

- In a pass, we consider remaining elements and compare all adjacent and swap if larger element is before a smaller element. If we keep doing this, we get the largest (among the remaining elements) at its correct position.

2.1.4. Sắp xếp chèn - Insertion sort

Insertion sort is a simple sorting algorithm that works by iteratively inserting each element of an unsorted list into its correct position in a sorted portion of the list. It is like sorting playing cards in your hands. You split the cards into two groups: the sorted cards and the unsorted cards. Then, you pick a card from the unsorted group and put it in the right place in the sorted group.

- We start with the second element of the array as the first element is assumed to be sorted.
- Compare the second element with the first element if the second element is smaller then swap them.
- Move to the third element, compare it with the first two elements, and put it in its correct position
- Repeat until the entire array is sorted.

2.2. Các thuật toán sắp xếp có độ phức tạp O(nlogn)

2.2.1. Sắp xếp trộn - Merge sort

1. ***Divide:*** *Divide the list or array recursively into two halves until it can no more be divided.*
2. ***Conquer:*** *Each subarray is sorted individually using the merge sort algorithm.*
3. ***Merge:*** *The sorted subarrays are merged back together in sorted order. The process continues until all elements from both subarrays have been merged.*