BỘ XÂY DỰNG
TRƯỜNG ĐHXD MIỀN TÂY

ĐÁP ÁN ĐỀ THI CHÍNH THỨC

ĐÁP ÁN – THANG ĐIỂM
KỲ THI KTHP HỌC KỲ II NĂM HỌC 2023-2024
Trình độ: ĐẠI HỌC; Ngày thi: 29/5/2024
Môn: PHÂN TÍCH THIẾT KẾ THUẬT TOÁN
(Đáp án - thang điểm gồm 04 trang)

Câu	Nội dung	Thang điểm
1	Thế nào là bài toán tìm kiếm?	**0.5**
	Viết thuật toán tìm kiếm tuần tự	**0.5**
	Vẽ sơ đồ thuật toán tìm kiếm tuần tự	**0.5**
	Đánh giá độ phức tạp thuật toán tuần tự	**0.5**
2	Thực hiện các bước với thuật toán Insertion Sort để sắp xếp mảng. Có 10 bước, mỗi bước thực hiện đúng được 0.4 điểm	**2.0**
3	Ý tưởng thuật toán Bubble Sort	**1.0**
	Viết thuật toán Bubble Sort	**1.0**
	Đánh giá độ phức tạp của thuật toán Bubble Sort	**1.0**
4	Trình bày giải pháp dựa trên thuật toán vét cạn để giải quyết bài toán "Cái ba lô"	**3.0**
	Tổng điểm	**10,0đ**

Câu 1:
Bài toán tìm kiếm là một loại bài toán tìm ra một đối tượng hoặc một tập hợp các đối tượng từ một không gian tìm kiếm, thoả mãn một số điều kiện cụ thể.
Phân Loại Bài Toán Tìm Kiếm

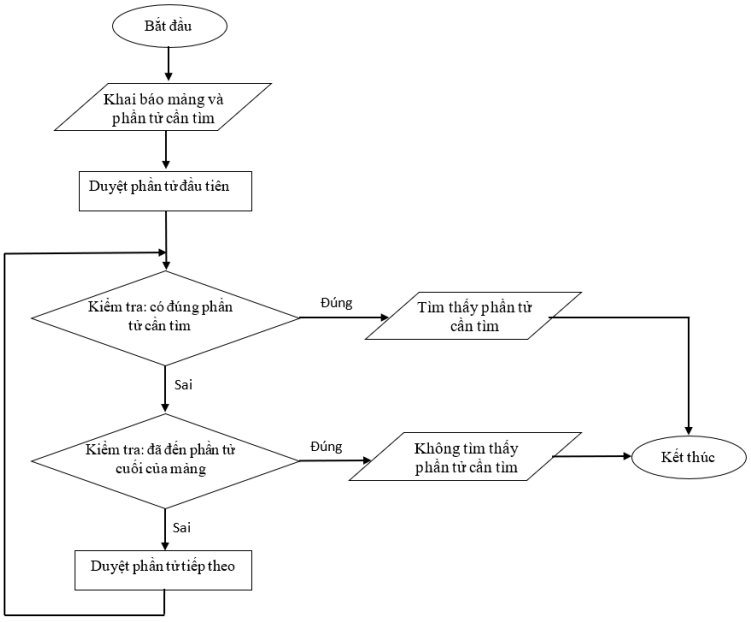
- Tìm kiếm tuần tự (Sequential Search hoặc Linear Search): Duyệt qua từng phần tử trong tập hợp từ đầu đến cuối để tìm kiếm phần tử cần tìm.
- Tìm kiếm nhị phân (Binary Search): Sử dụng trên các tập hợp đã được sắp xếp, bằng cách liên tục chia đôi tập hợp để giảm phạm vi tìm kiếm.
- Tìm kiếm theo băm (Hash Search): Sử dụng các bảng băm để tìm kiếm phần tử một cách nhanh chóng thông qua các hàm băm.
- Tìm kiếm bằng cây (Tree Search): Sử dụng các cấu trúc cây như cây nhị phân tìm kiếm (Binary Search Tree) để tổ chức và tìm kiếm dữ liệu.

Các bài toán tìm kiếm rất phổ biến và xuất hiện trong nhiều ứng dụng thực tế, từ cơ sở dữ liệu, hệ thống tệp tin, cho đến các thuật toán tối ưu hóa và trí tuệ nhân tạo.

- Thuật toán tìm kiếm tuần tự:
(Có nhiều cách viết tùy tư duy của mỗi sinh viên)

```
#include <iostream>
using namespace std;
// Hàm tìm kiếm tuần tự
int linearSearch(int arr[], int size, int target) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            return i; // Trả về chỉ số của phần tử nếu tìm thấy
        }
    }
    return -1; // Trả về -1 nếu không tìm thấy
}
int main() {
    int arr[] = {4, 2, 5, 1, 3, 9, 7};
    int size = sizeof(arr) / sizeof(arr[0]);
    int target = 5;
    int result = linearSearch(arr, size, target);
    if (result != -1) {
        cout << "Phan tu " << target << " duoc tim thay tai chi so " << result << endl;
    } else {
        cout << "Phan tu " << target << " khong co trong mang" << endl;
    }
    return 0;
}
```

- Sơ đồ thuật toán:



- Độ phức tạp của thuật toán:

O(n), với n là số lượng phần tử trong mảng.

Câu 2: Các bước thực hiện của thuật toán Insertion Sort áp dụng cho mảng sau:

[4, 7, 2, 10, 15, 14, 8, 10, 8, 5]

Bước 01: [**4**, 7, 2, 10, 15, 14, 8, 10, 8, 5]
Bước 02: [4, **7**, 2, 10, 15, 14, 8, 10, 8, 5]
Bước 03: [**2**, 4, 7, 10, 15, 14, 8, 10, 8, 5]
Bước 04: [2, 4, 7, **10**, 15, 14, 8, 10, 8, 5]
Bước 05: [2, 4, 7, 10, **15**, 14, 8, 10, 8, 5]
Bước 06: [2, 4, 7, 10, **14**, 15, 8, 10, 8, 5]
Bước 07: [2, 4, 7, **8**, 10, 14, 15, 10, 8, 5]
Bước 08: [2, 4, 7, 8, 10, **10**, 14, 15, 8, 5]
Bước 09: [2, 4, 7, 8, **8**, 10, 10, 14, 15, 5]
Bước 10: [2, 4, **5**, 7, 8, 8, 10, 10, 14, 15]
Kết quả: **[2, 4, 5, 7, 8, 8, 10, 10, 14, 15]**

Câu 3: Thuật toán Buble Sort

- Ý tưởng thuật toán:

Duyệt qua từng cặp phần tử liên tiếp trong mảng và đổi chỗ các phần tử nếu chúng không theo thứ tự mong muốn (lớn hơn hoặc nhỏ hơn, tùy thuộc vào chiều sắp xếp).

Thuật toán Bubble Sort hoạt động như sau:

- Duyệt qua từng phần tử của mảng từ đầu đến trước phần tử cuối.
- So sánh các phần tử lân cận của mảng.
- Nếu cặp phần tử hiện tại không theo thứ tự mong muốn (ví dụ: phần tử đứng trước lớn hơn phần tử đứng sau trong trường hợp sắp xếp tăng dần), thì đổi chỗ chúng.
- Sau mỗi lần lặp, phần tử lớn nhất (hoặc nhỏ nhất tùy theo chiều sắp xếp) sẽ "nổi" lên đầu của mảng.
- Lặp lại quá trình cho đến khi không còn phần tử nào cần đổi chỗ nữa.
- Thuật toán sẽ dừng lại khi không còn có sự đổi chỗ nào xảy ra trong một vòng lặp.

- Thuật toán BubbleSort()

(Có nhiều cách viết tùy tư duy của mỗi sinh viên)

```
#include <iostream>
using namespace std;
// Hàm sắp xếp mảng bằng thuật toán Bubble Sort
void bubbleSort(int arr[], int n) {
   for (int i = 0; i < n - 1; ++i) {
      // Biến swapped để kiểm tra xem đã có sự đổi chỗ hay chưa
      bool swapped = false;
      // Lặp qua từng cặp phần tử liền kề và đổi chỗ nếu cần
      for (int j = 0; j < n - i - 1; ++j) {
         if (arr[j] > arr[j + 1]) {
            // Đổi chỗ các phần tử
            swap(arr[j], arr[j + 1]);
            swapped = true;
         }
      }
      // Nếu không có sự đổi chỗ nào trong vòng lặp thì mảng đã được sắp xếp
      if (!swapped) break;
   }
}
// Hàm in ra mảng sau khi sắp xếp
```

```
void printArray(int arr[], int n) {
   for (int i = 0; i < n; ++i) {
      cout << arr[i] << " ";
   }
   cout << endl;
}
int main() {
   int arr[] = {4, 7, 2, 10, 15, 14, 8, 10, 8, 5};
   int n = sizeof(arr) / sizeof(arr[0]);
   cout << "Mang truoc khi sap xep: ";
   printArray(arr, n);
   bubbleSort(arr, n);
   cout << "Mang sau khi sap xep: ";
   printArray(arr, n);
   return 0;
}
```

- Đánh giá độ phức tạp:

Thuật toán Bubble Sort lặp qua mảng và so sánh từng cặp phần tử liền kề. Tổng số lần lặp của thuật toán là n (số lượng phần tử trong mảng).

Mỗi lần lặp, thuật toán thực hiện n−i−1 lần so sánh (với i là số lần lặp bên ngoài).

Do đó, tổng số so sánh là:

$$\sum_{i=0}^{n-2}(n-i-1) = (n-1) + (n-2) + ... + 1$$

Đây là tổng của dãy số hạng từ 1 đến n−1, tức là (n×(n−1))/2, có thể gọi là $O(n^2)$ trong trường hợp trung bình và trường hợp xấu nhất.

Vì vậy, độ phức tạp thời gian của Bubble Sort là $O(n^2)$.

Câu 4: Bài toán "Cái ba lô"

Phương án:

- Gọi $X=(X_1, X_2, \ldots, X_n)$ với X_i là số nguyên không âm, là một phương án.
- X_i là số đồ vật thứ i
- Cần tìm X sao cho:

$X_1g_1 + X_2g_2 + \ldots + X_ng_n <= W$

$F(X) = X_1v_1 + X_2v_2 + \ldots + X_nv_n => Max$

Ta có:

- W = 53
- X=(Xa, Xb, Xc, Xd)

Xa=53/18=2 => Trọng lượng còn lại là: 53-2*18=17

Xb=17/10=1 => Trọng lượng còn lại là: 17-1*10=7

Xc=7/4=1 => Trọng lượng còn lại là: 7=1*4=3

Xd=3/2=1 => Trọng lượng còn lại là: 3-1*2=1

Ta có: X=(2, 1, 1, 1)

Vậy với ba lô đã cho ta có thể mang được:

2 đồ vật A; 1 đồ vật B; 1 đồ vật C; 1 đồ vật D

Tổng trọng lượng có thể mang là: 2*18+1*10+1*4+1*2 = 52 < W

Tổng giá trị tương ứng ba lô mang được là: 2*30+1*24+1*9+1*3 = 96

---------- Hết ----------