**Đề cương ôn tập tin học cơ sở thạc sĩ**

Phần A: Cấu trúc dữ liệu và giải thuật

1. Cấu trúc dữ liệu cơ bản
   1. Danh sách tuyến tính:
      * Là một dãy các phần tử có cùng kiểu dữ liệu được sắp xếp liên tiếp nhau trong bộ nhớ
      * Các phần tử của danh sách nằm liên tục nhau trong bộ nhớ, giữa các phần tử này không có khoảng trống
      * Kích thước của danh sách sẽ được cấp phát theo khai báo
      * Cài đặt danh sách tuyến tính: array
      * Thể hiện của danh sách tuyến tính: mảng một chiều
      * Không có khoảng trống giữa các liên kết
   2. Danh sách liên kết:
      * Có hai loại danh sách liên kết: Đơn và đôi
      * Tập hợp các node được sắp xếp sao cho mỗi node chứa 1 data và một con trỏ trỏ đến phần tử kế tiếp của danh sách liên kết đó
      * Nếu con trỏ trỏ đến NULL thì đó là phần tử cuối cùng của danh sách
      * Bản chất giống mảng, nhưng kích thước có thể thay đổi trong quá trình thêm, xóa dữ liệu
      * Kích thước tối đa tùy thuộc vào bộ nhớ
      * Được cấp phát động
   3. Ngăn xếp:
      * Là một cấu trúc dữ liệu dạng thùng chứa hoạt động theo nguyên lý Last in first out
      * Hai phép toán cơ bản: push, pop
      * Push: Đẩy dữ liệu vào đầu ngăn xếp
      * Pop: Lấy dữ liệu trên cùng của ngăn xếp
   4. Hàng đợi:
      * Là cấu trúc dữ liệu dùng để chứa các dữ liệu làm việc theo cơ chết First in first out
      * Thêm một dữ liệu: enqueue
      * Lấy dữ liệu: dequeue
2. Sắp xếp:
   1. Sắp xếp chọn: <https://nguyenvanhieu.vn/thuat-toan-sap-xep-chon/>
      * Từ vị trí i, chạy lần lượt đến hết chuỗi tìm số bé hơn i và gán là min
      * Tìm số bé hơn min
      * Nếu có, thì số mới tìm được sẽ gán là min
      * Nếu không có, thì swap giá trị vị trí i và min
   2. Sắp xếp chèn: <https://nguyenvanhieu.vn/thuat-toan-sap-xep-chen/>
      * Phải luôn đảm bảo chuỗi số được sắp xếp tăng dần
      * So sánh i với số i-1
      * Nếu lớn hơn: bỏ qua, chạy tiếp số tiếp theo
      * Nếu nhỏ hơn: dịch lần lượt các số i-1 sang phải
   3. Sắp xếp nổi bọt: <https://nguyenvanhieu.vn/thuat-toan-sap-xep-noi-bot/>
      * So sánh 1 cặp số i và i+1
      * Nếu i > i+1 thì đổi chỗ
      * Nếu không thì lại so sánh cặp tiếp theo i+1 với i+2
      * Lặp lại liên tục về đầu vòng lặp
   4. Sắp xếp nhanh: <https://nguyenvanhieu.vn/thuat-toan-sap-xep-quick-sort/>
      * Đặt pivot là số đầu tiên hoặc cuối cùng trong mảng
      * i và j là 2 bộ đếm đầu và cuối của mảng
      * Nếu a[i] < pivot thì sẽ chạy i tiếp để tìm số lớn hơn
      * Nếu a[i] > pivot, thì sẽ chạy j từ cuối về đầu mảng
      * Nếu a[j] > pivot thì sẽ chạy tiếp
      * Nếu a[j] < pivot thì sẽ swap a[i] và a[j] và i++, j--
      * Nếu j chạy qua i thì swap a[j] = pivot
      * Lặp lại các bước
   5. Sắp xếp vun đống:
      * Biểu diễn bằng cây nhị phân
      * Node cha sẽ là n
      * Node con trái là 2n + 1
      * Node con phải là 2n + 2
      * So sánh node con xem node nào là max
      * Rồi đem so sánh với node cha
      * Nếu node cha < node con => swap(cha,con)
      * Làm lần lượt đến khi root là lớn nhất, sau đó swap(root, lá cuối cùng)
      * Sau đó cho lá cuối cùng vào 1 mảng, đó là số lớn nhất
      * Tiếp tục đệ quy
   6. Sắp xếp trộn:
      * Chia để trị
      * So sánh các node con sau đó swap lần lượt cho nhau
3. Tìm kiếm:
   1. Tìm kiếm tuần tự:
      * So sánh chuỗi được cho với lần lượt từng phần tử của mảng
   2. Tìm kiếm nhị phân:
      * Bắt buộc chuỗi phải được sắp xếp theo thứ tự tăng dần hoặc giảm dần
      * Gọi số cần tìm là x
      * Tìm index ở giữa: m = (left + right) / 2
      * Nếu x > m 🡺 x ở khoảng bên trái, lặp tiếp các bước trên
      * Nếu x < m 🡺 x ở khoảng bên phải, lặp tiếp các bước trên
4. Đồ thị và cây
   1. Phương pháp duyệt chiều rộng: <https://vimentor.com/vi/lesson/duyet-cay-theo-chieu-rong-1>
      * Duyệt lần lượt từ trái qua phải từng phần tử của cây
      * Sử dụng queue
      * Bước 1: Add node cha vào queue
      * Bước 2: Kiểm tra queue có rỗng không, có thì dừng chương trình, không thì sang bước 3
      * Bước 3: Lấy phần tử đầu tiên ra khỏi queue và chuyển trạng thái là đã duyệt
      * Bước 4: enqueue các con của vần tử vừa lấy ra
      * Bước 5: Làm lần lượt cho đến khi trong queue không còn phần tử

|  |
| --- |
|  |
|  |
|  |

* 1. Phương pháp duyệt chiều sâu:
     + Sử dụng heap
     + Bước 1: Add node cha vào queue
     + Bước 2: Kiểm tra queue có rỗng không, có thì dừng chương trình, không thì sang bước 3
     + Bước 3: Lấy phần tử đầu tiên ra khỏi queue và chuyển trạng thái là đã duyệt
     + Bước 4: enqueue các con của vần tử vừa lấy ra
     + Bước 5: Làm lần lượt cho đến khi trong queue không còn phần tử
  2. Cây nhị phân
     + Là một cây tuân thủ quy tắc các phần tử bên trái sẽ bé hơn các phần tử bên phải
  3. Cây tìm kiếm nhị phân
     + So sánh giá trị cần tìm với 2 node trái phải
     + Sẽ đi sâu lần lượt vào các node con của cây

1. Kỹ thuật băm
   1. Bảng băm, hàm băm:
      * Hỗ trợ việc lưu giá trị theo kiểu key – value
      * Tuy sử dụng cơ chế lưu key – value nhưng các key sẽ được chuyển đổi thành index để lưu vào mảng
      * Công thức tính index: key.length % độ dài mảng
   2. Khắc phục xung đột bằng phương pháp duyệt tuyến tính
      * Tính index theo công thức trên
      * Nếu index đó đã có dữ liệu, nó sẽ là stopIndex
      * Duyệt 1 vòng từ stopIndex đó xem còn chỗ nào trống không để chèn vào
      * Tính theo công thức mới để tìm vị trí tiếp theo: (haskey + 1) % độ dài mảng
2. Đánh giá, thiết kế thuật toán
   1. Độ phức tạp của thuật toán
      * O(1): Nhanh nhất
      * O(log2n)
      * O(√n)
      * O(n)
      * O(nlog2n)
      * O(n2)
      * O(2n)
      * O(n!): chậm nhất