**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
      * Lưu ý: Phải loại bỏ các hằng số
      * O(1): Nhanh nhất

* Cộng hai số
* Không có vòng lặp
* VD:

|  |
| --- |
| int getData(int[] arr){  return arr[0];  } |

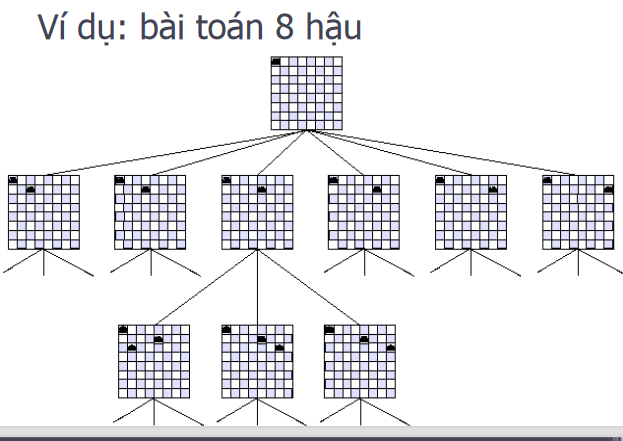
* + - O(log2n):
* Giảm kích thước dữ liệu đầu vào
* Không cần phải duyệt hết các giá trị
* Hành động tiếp theo có thể thực hiện theo điều kiện nào đó

|  |
| --- |
| int binarySearch(int[] arr, int target){  int left = 0;  int right = arr.length – 1;  int middle = 0;  while(left < right){  middle = (left + right)/2;  if (arr[mid] == target) {  return arr[mid];  }  else if (arr[mid] < target) {  left = mid + 1;  }  else if (arr[mid] > target) {  right = mid – 1;  }  }  return -1;  } |

* + - O(√n)
    - O(n): Tìm kiếm tuyến tinh
* Số lượng hoạt động diễn ra tỷ lệ tuyến tính với kích thước của n
* VD: Thao tác in ra 100 phần tử 🡺 duyệt 100 lần
  + - O(nlog2n): Thời gian chuẩn tính
* Mỗi thao tác trong dữ liệu đầu vào có độ phức tạp về thời gian logarit
* VD: Đối với mỗi giá trị trong dữ\_liệu\_1 (O(n)) sử dụng tìm kiếm nhị phân (O(log2n)) để tìm kiếm cùng một giá trị trong dữ\_liệu\_2
* Ứng dụng: MergeSort, HeapSort,
  + - O(n2):
* Hai vòng for lồng nhau
* Thực hiện phép toán thời gian tuyến tính cho mỗi giá trị trong dữ liệu đầu vào
* Đối với danh sách n mục, chúng ta thực hiện n thao tác cho mỗi mục
* Ứng dụng: QuickSort, BubbleSort, InsertionSort, SelectionSort
* VD: 10 mục, mỗi mục trong 10 mục kia sẽ thực hiện 10 phép toán 🡺 102

|  |
| --- |
| for(int i = 0 ; i < n ; i ++) {  for (int j = 0 ; j < n ; j ++) {  System.out.println(y + j);  }  } |

* + - O(2n):
* Tăng trưởng nhân đôi với mỗi lần bổ sung vào tập dữ liệu đầu vào
* Đệ quy
* VD: “Tower of Hanoi”, Fibonacci
  + - O(n!): chậm nhất
* Dựa trên kích thước của dữ liệu đầu vào
* Phát triển nhanh chóng ngảy cả đối với đầu vào kích thước nhỏ
* VD: HeapSort
  1. Phương pháp chia để trị:
  2. Phương pháp vét cạn:
     + Tìm lời giải tối ưu trong số các lời giải
     + Mỗi lời giải gom thành n thành phần
     + Quá trình xây dựng lời giải được xem như qua trình tìm n thành phần, mỗi thành phần được tìm kiếm trong 1 bước
     + Các bước phải có dạng giống nhau
     + Ở mỗi bước, ta có thể dễ dàng chọn lựa một thành phần
     + Sau khi thực hiện đủ n bước, ta được 1 lời giải
     + Khi một lời giải bị sai, áp dụng phương pháp quay lui
     + VD:



* 1. Phương pháp tham lam:
     + Mục đích:
* Tìm một lời giải tốt trong thời gian chấp nhận được ( độ phức tạp đa thức thay vì lũy thừa )
  + - Ý tưởng:
* Chia quá trình tìm lời giải thành nhiều bước như kỹ thuật quay lui
  + - Với mỗi bước
* Sắp xếp các lựa chọn cho bước đó theo thứ tự nào đó “có lợi” (tăng dần hoặc giảm dần tùy theo cách lập luận)
* Chọn lựa chọn tốt nhất rồi đi tiếp bước kế (**Không quay lui**)
  1. Phương pháp quy hoạch động:
     + Mục đích:
* Cải tiến thuật toán chia để trị hoặc quay lui, vét cạn để giảm thời gian thực hiện.

Phần C: Lý thuyết Cơ sở dữ liệu

1. Các khái niệm cơ bản
   1. Cơ sở dữ liệu:
      * Là một hệ thống bao gồm rất nhiều thông tin, dữ liệu được xây dựng theo một cấu trúc nhất định nhằm đáp ứng nhu cầu khai thác, sử dụng của nhiều người
   2. Mô hình cơ sở dữ liệu:
      * Mô hình phân cấp: Gồm nhiều nút, mỗi nút biểu diễn một thực thể nhất định
      * Mô hình dữ liệu mạng: Có cấu trúc dữ liệu tổ chức thành một đồ thị hướng
      * Dựa trên lý thuyết tập hợp và đại số quan hệ. Sử dụng các phép toán thao tác trên dữ liệu dựa trên lý thuyết tập hợp toán học
   3. Hệ quản trị cơ sở dữ liệu:
      * Là một gói phần mềm dùng để thiết kế, xác định, thao tác, truy xuất và quản lý CSDL