**Tuần 1**

***Bài 1: Cho 2 mạng đã được sắp xếp. Tìm giao của hai mảng với độ phức tạp là O(mlog(n)) với m và n tương ứng là độ dài của hai mảng.***

* Ý tưởng bài toán:
  + Duyệt tuần tự mảng có m thành phần. Với mỗi thành phần tìm được, sử dụng thuật toán tìm kiếm nhị phân cho phần tử đó với mảng có n phần tử.
  + Duyệt tuần tự mảng m phần tử có độ phức tạp O(m), với mỗi lần duyệt 1 phần tử thì sẽ thực hiện thuật toán tìm kiếm nhị phân có độ phức tạp O(log(n)) nên độ phức tạp của bài toán là O(m\*log(n)).
* Kiến thức cần có:
  + Độ phức tạp thuật toán:
    - Định nghĩa: số phép toán của giải thuật so với kích thước đầu vào.
  + Tìm kiếm nhị phân:
    - Thuật toán tìm kiếm sử dụng trong mảng đã sắp xếp bằng cách chia đôi mảng cần tìm kiếm nhiều lần.
    - Mảng được chia thành 2 phần left và right tương ứng với phần tử đứng giữa là mid.
    - So sánh giá trị mid với giá trị key cần tìm kiếm, nếu mid nhỏ hơn giá trị key thì phần ta tìm kiểm trong mảng left, ngược lại.
* Áp dụng:
  + Với thuật toán tìm kiếm nhị phân, ta triển khai thành hàm riêng với kiểu trả về là Integer, giá trị trả về là vị trí của phần tử được tìm thấy trong mảng, nếu không tìm thấy, trả về -1.
  + Code: A computer screen with text

    Description automatically generated
  + Độ phức tạp thuật toán:
    - Với đầu vào là một mảng có n phần tử, mỗi lần thực hiện vòng lặp, ta chia đôi mảng (n/2), chia 2 cho đến khi tìm được phần tử cần tìm. Trong trường hợp tệ nhất, phép toán dần tiến đến 1, nghĩa là, với x lần chia cho 2, ta được n / 2x = 1 => x = log2(n). Vậy độ phức tạp của thuật toán trên là O(log(n)).
  + Với hàm tìm kiếm sẵn có, áp dụng vào bài toán, ta duyệt từng phần tử của mảng có m phần tử như một key để tìm kiếm nhị phân trong mảng có n phần tử.
  + Code: A computer screen shot of a code

    Description automatically generated
  + Độ phức tạp:
    - Ta xét vòng for m lần tương ứng với m phần tử, vậy nên ở vòng for ta có độ phức tạp là O(m). Vì bên trong vòng for có chứa hàm binary search nên với m phần tử tương ứng, ta có độ phức tạp thuật toán là O(m\*log(n)).
* Triển khai:
  + Giả sử mảng a[] kiểu int có m = 6 phần tử từ 1 đến 6, mảng b[] kiểu int có n = 5 phần tử từ 4 đến 8. Ta có: A computer screen with text and numbers

    Description automatically generated
* Kết quả:
  + Kết quả tương ứng là phần giao của 2 mảng là: {4, 5, 6}.A screen shot of a computer

    Description automatically generated

***Bài 2: Một url là url thỏa mãn các yếu tố***

***(ví dụ 1 url hợp lệ:*** [***https://tiki.vn/dien-thoai-may-tinh-bang/c1789?src=mega-menu***](https://tiki.vn/dien-thoai-may-tinh-bang/c1789?src=mega-menu)***):***

* ***Bắt đầu bằng http hoặc https***
* ***Có thể chứa www hoặc không***
* ***Tên miền của url (ví dụ tiki) chỉ chứa các ký tự la tinh hoa/thường từ a-z và chữ số.***
* ***Phần domain extension (ví dụ .vn) phải chứa dấu chấm ở đầu, thêm sau là các ký tự in thường có độ dài từ 2 đến 6 ký tự.***
* ***Phần path (ví dụ /dien-thoai-may-tinh-bang/c1789?src=mega-menu) không được chứa dấu cách.***

***Viết đoạn regex để kiểm tra xem một url bất kỳ có là hợp lệ không?***

* Ý tưởng:
  + Sử dụng các cú pháp regex để giải quyết bài toán.
* Kiến thức:
  + Regex:
    - Là các mẫu (pattern) thay vì một chuỗi cụ thể, miêu tả một bộ chuỗi các ký tự theo những quy tắc và cú pháp nhất định.
* Áp dụng:
  + Viết một hàm kiểm tra tính khả dụng của một địa chỉ URL với kiểu tra về là boolean (true/false).
  + A computer screen shot of a computer code

    Description automatically generatedCode:
  + Giải thích:
    - Bắt đầu bằng http hoặc https: “http(s)?” với dấu “?” đằng sau “(s)” biểu thị cho việc có thể có 0 lần “s” hoặc 1 lần “s”.
    - Có thể chứa www hoặc không: “(www\\.)?” với “\\.” biểu thị dấu “.” đằng sau www nếu như có www.
    - Tiền miền của url chỉ chứa các ký tự la tinh hoa/thường từ a-z và chữ số: “[\\w]” – thay thế cho “[a-zA-Z0-9]”, dấu “+” ngay sau thể hiển cho việc có thể có 1 hoặc nhiều hơn 1 ký tự.
    - Phần domain extension: “(\\.[a-z]{2,6})+” – biểu thức “{2,6}” ngay sau “[a-z]” thể hiện cho việc các ký tự từ a tới z có thể có ít nhất là 2 và nhiều nhất là 6 ký tự.
    - Phần path: “(\\/[^\\s]\*)\*” – biểu thức “\\/” thể hiện dấu “/” cho đường dẫn, “[^\\s]” biểu thị các ký tự không phải dấu cách (space), theo sau đó là dấu “\*” thể hiện biểu thức trước nó có thể xuất hiện 0 hoặc nhiều lần.
* Triển khai:
  + Tạo vòng lặp while với điều kiện dừng là khi người dùng nhập vào “exit”, vòng lặp sẽ nhận URL người dùng nhập để kiểm tra tính đúng sai và trả về kết quả true hoặc false.
  + A computer screen shot of text

    Description automatically generatedCode:
* Kết quả:
  + Với đầu vào như đề bài:
  + Bỏ “s” ở https:
  + Thêm www:
  + Thêm ký tự “?” vào tiki:
  + Domain dài hơn 6 ký tự:
  + Path chứa dấu cách:
  + A black background with white text

    Description automatically generatedThoát chương trình:

***Bài 3: Tìm vị trí đầu tiên và cuối cùng của một thành phần trong mảng đã sắp xếp***

***Input: một mảng đã được sắp xếp tăng dần và một số***

***Output: vị trí đầu tiên và cuối cùng của số đã cho***

***Nếu không tồn tại thì trả về -1***

***Ví dụ:***

***Input: nums = [5, 6, 7, 8, 8, 15], target = 8***

***Output: [3, 4]***

***Yêu cầu độ phức tạp: O(log n)***

* Ý tưởng:
  + Sử dụng tìm kiếm nhị phân (binary search)
* Áp dụng:
  + Vì đề bài yêu cầu tìm vị trí đầu tiên và cuối cùng của số đã cho nên bài toán chia thành 2 bài toán nhỏ: tìm vị trí đầu tiên và tìm vị trí cuối cùng của số đã cho.
  + Với vị trí đầu tiên, áp dụng tìm kiếm nhị phân bên trái. Nếu phần tử ở vị trí mid bằng với giá trị tìm kiếm và phần từ thứ mid-1 cũng bằng giá trị tìm kiếm thì sẽ chuyển vị trí end thành mid-1 và ngược lại thì trả về kết quả mid.
  + A computer screen with text and numbers

    Description automatically generatedCode:
  + A computer screen shot of a code

    Description automatically generatedVị trí cuối cùng cũng tương tự như vậy:
  + A screen shot of a computer code

    Description automatically generatedTạo hàm trả về mảng gồm 2 phần tử int là 2 vị trí thỏa mãn đề bài:
* Triển khai:
  + A computer screen shot of a code

    Description automatically generatedTạo mảng gồm các phần tử và giá trị key như đề bài:
  + Độ phức tạp của hàm firstSearch() là O(log n), hàm lastSearch() là O(log n) nên khi chạy tuần tự, độ phức tạp của bài toán hiện tại là vẫn là O(log n).
* A screen shot of a computer

  Description automatically generatedKết quả: