|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ĐẠI HỌC BÁCH KHOA HÀ NỘI  TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG  ---🙢🙠---  A picture containing icon  Description automatically generated  **BÁO CÁO**  MÔN: PROJECT I  **Chủ đề 1: Hàm và đệ quy**   |  |  | | --- | --- | | GVHD: | TS. Nguyễn Khánh Phương | | Sinh viên – MSSV: | Phạm Xuân Bách – 20204633 | | Lớp: | Khoa học máy tính 01 – K65 | | Mã lớp: | 721012 | |  |  |   Hà Nội, 10/2022 |

Contents

[**1.** **Bài tập 1.** 1](#_Toc117539027)

[**1.1. Yêu cầu đề bài.** 1](#_Toc117539028)

[**1.2. Mô tả thuật toán.** 1](#_Toc117539029)

[**1.3. Hình ảnh kết quả.** 1](#_Toc117539030)

[**2.** **Bài tập 2.** 1](#_Toc117539031)

[**2.1. Yêu cầu đề bài.** 1](#_Toc117539032)

[**2.2. Mô tả thuật toán.** 2](#_Toc117539033)

[**2.3. Hình ảnh kết quả.** 2](#_Toc117539034)

[**3.** **Bài tập 3.** 2](#_Toc117539035)

[**3.1. Yêu cầu đề bài.** 2](#_Toc117539036)

[**3.2. Mô tả thuật toán.** 2](#_Toc117539037)

[**3.3. Hình ảnh kết quả.** 3](#_Toc117539038)

[**4.** **Bài tập 4.** 3](#_Toc117539039)

[**4.1. Yêu cầu đề bài.** 3](#_Toc117539040)

[**4.2. Mô tả thuật toán.** 3](#_Toc117539041)

[**4.3. Hình ảnh kết quả.** 5](#_Toc117539042)

# **Bài tập 1.**

## **1.1. Yêu cầu đề bài.**

Viết hàm đệ quy tính tổng các số trong dãy số gồm n số nguyên cho trước: a[0], a[1],…, a[n-1]. Viết chương trình sử dụng hàm này như sau: yêu cầu người dùng nhập từ bàn phím n, và n số nguyên; sau đó sử dụng hàm đã xây dựng được để in kết quả ra màn hình.

## **1.2. Mô tả thuật toán.**

Với bài toán này, em sử dụng hàm đệ quy sumArray(int a[], int n, int k), trong đó a là mảng chứa các giá trị nhập vào, n là số phần tử trong mảng, k là vị trí của phần tử hiện tại. Điều kiện để thoát khỏi đệ quy là khi k đang ở vị trí phần tử cuối cùng trong mảng (k=n-1) thì trả về giá trị của phần tử a[k-1]. Nếu k chưa ở vị trí cuối cùng thì tiếp tục gọi vòng đệ quy cho giá trị tiếp theo cộng với giá trị ở vị trí hiện tại: a[k] + sumArray(a, n, k+1).

## **1.3. Hình ảnh kết quả.**

Text

Description automatically generated

# **Bài tập 2.**

## **2.1. Yêu cầu đề bài.**

Xây dựng hàm cộng hai ma trận A, B có cùng kích thước m\*n, hàm này sẽ in ra ma trận tổng C =A + B ra màn hình dưới dạng bảng. Viết chương trình sử dụng hàm này như sau: yêu cầu người dùng nhập từ bàn phím lần lượt giá trị m, n và hai ma trận A, B. Sau đó sử dụng hàm đã xây dựng, in kết quả ra màn hình.

## **2.2. Mô tả thuật toán.**

Bài toán này em có sử dụng hai hàm input, output ma trận để thuận tiện hơn cho chương trình.

Đối với hàm tính tổng hai ma trận chính, sử dụng 2 vòng lặp for xét từng vị trí theo hàng, cột của ma trận và tính tổng giá trị ở 2 vị trí đó trong hai ma trận A và B, gán vào ma trận C.

## **2.3. Hình ảnh kết quả.**

Text

Description automatically generated

# **Bài tập 3.**

## **3.1. Yêu cầu đề bài.**

Hãy viết hàm đệ quy int countDigits(int num) trả về số lượng chữ số của số nguyên lưu trong biến num.

Ví dụ: num = 1230, thì hàm countDigits(num) sẽ trả về giá trị 4.

## **3.2. Mô tả thuật toán.**

Với bài toán này, số chữ số được xác định với số lần giá trị đó chia cho 10, lấy phần kết quả nguyên của nó và lặp lại liên tục cho đến khi số cuối cùng là số có 1 chữ số. Với cách đó mỗi vòng lặp sẽ là một lần đếm chữ số cuối cùng và loại nó đi để đếm tiếp. Điều kiện để thoát khỏi đệ quy là giá trị hiện tại nhỏ hơn 10, khi đó trả về số nguyên 1. Nếu không thì sẽ trả về 1 và gọi tiếp đệ quy với giá trị đó chia cho 10, lấy phần nguyên.

## **3.3. Hình ảnh kết quả.**

Graphical user interface, text

Description automatically generated

# **Bài tập 4.**

## **4.1. Yêu cầu đề bài.**

Cho hai xâu kí tự word1 và word2, hãy xác định số bước cần thực hiện ít nhất để biến đổi word1 thành word2, biết rằng mỗi bước chỉ được thực hiện 1 trong 3 thao tác sau: (1) thêm 1 kí tự, (2) xóa 1 kí tự, (3) thay thế 1 kí tự bởi 1 kí tự khác.

Ví dụ: word1 = “horse” và word2 = “ros”. Khi đó tối thiểu phải thực hiện 3 bước để có thể biến đổi horse thành ros.

* Bước 1: horse 🡪 rorse (thay thế “h” bới “r”)
* Bước 2: rorse 🡪 rose (xóa kí tự “r” ở vị trí thứ 3)
* Bước 3: rose 🡪 ros (xóa kí tự “e”)

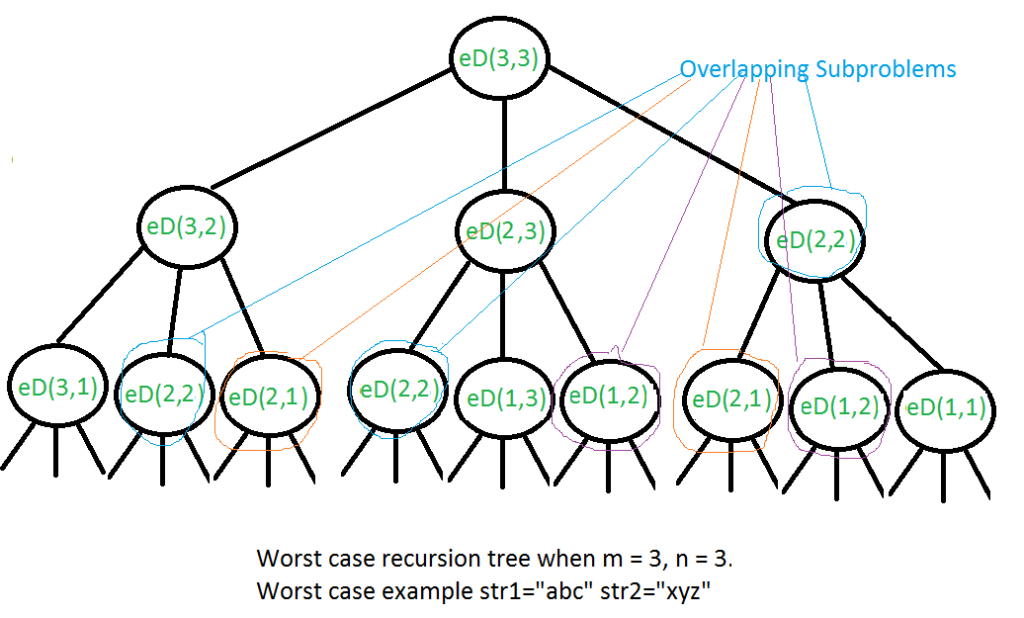
## **4.2. Mô tả thuật toán.**

Ý tưởng của thuật toán là xét từng chữ cái một từ cuối lên trên đầu của cả hai từ. Giả sử hai từ có độ dài lần lượt là m và n. Nếu như hai chữ cái thứ m và n ở từ thứ nhất và thứ hai giống nhau thì có thể bỏ qua và xét tiếp chữ cái thứ m – 1 và n - 1, không có gì ảnh hưởng và thay đổi cả. Ví dụ “house” và “rose” có cùng chữ cái cuối là ‘e’ thì bỏ qua nó và xét tiếp “hors” và “ros”. Nếu hai chữ cái khác nhau thì có thể xét ba trường hợp :

* Trường hợp 1: thay thế bởi 1 kí tự khác, khi đó coi như là 1 bước và xét tiếp kí tự m – 1 và n – 1.
* Trường hợp 2: thêm 1 kí tự ở từ thứ nhất để kí tự m + 1 ở từ thứ nhất và n ở từ thứ hai giống nhau, khi đó có thể bỏ qua và duyệt sang m và n - 1. Ví dụ: “ant” và “end”, ta sẽ thêm chữ ‘d' để từ thứ nhất trở thành “antd”. Lúc này hai từ đều có kí tự ‘d’ giống nhau và tiếp tục xét “ant” và “en”.
* Trường hợp 3: xóa 1 kí tự ở từ thứ nhất, khi đó từ thứ nhất sẽ chuyển sang kiểm tra kí tự m – 1 còn từ thứ hai vẫn ở kí tự n. Ví dụ: “cor” và “evo”, ta xóa kí tự ‘r’ ở từ thứ nhất để tiếp tục kiểm tra “co” và “evo”.

Ba bước trên được gọi đệ quy liên tục và trả về số lần thực hiện ít nhất có thể. Điều kiện để thoát khỏi đệ quy là khi một trong hai từ không còn kí tự để xét thì sẽ trả về số kí tự của từ còn lại, chính là số kí tự phải thêm vào để được từ tương đương.

Dưới đây là một sơ đồ cây mô phỏng các lần gọi đệ quy:



*(Nguồn: Geeksforgeeks)*

Khi gọi đệ quy như vậy, nhiều hàm trùng lặp nhau đều phải chạy gây lãng phí thời gian. Để tránh được điều đó, em sử dụng đệ quy có nhớ để lưu trữ những giá trị đá được tính trước đó.

Hàm đệ quy sẽ được truyền vào độ dài của từ thứ nhất và thứ hai, lần lượt là m, n, đó đồng thời cũng là vị trí chứa giá trị của hàm đó. Nếu m = 0, tức là từ thứ nhất không còn kí tự thì sẽ cần n bước thêm n kí tự để hai xâu giống nhau, do đó giá trị tại vị trí (0, n) là n. Tương tự với giá trị tại vị trí (m, 0) = m. Đây sẽ là các giá trị cơ bản để tìm được các giá trị còn lại. Kết quả ở vị trí (m, n) sẽ là kết quả cuối cùng cần tìm của bài toán.

Thuật toán tương tự như ở trên, lần này em sẽ lưu các giá trị vào một mảng hai chiều động gán trước giá trị -1. Nếu ở vị trí đó mang giá trị khác -1 thì sẽ trả luôn kết quả, còn nếu bằng -1 thì sẽ tính giá trị tại đó theo đệ quy.

## **4.3. Hình ảnh kết quả.**

Text

Description automatically generated