

Toán rời rạc và thuật toán

Đại học Khoa học Tự nhiên

Khoa Toán - Cơ - Tin học

Khoa học dữ liệu K4

Tháng 8 năm 2022

Bài tập số 1

Nguyễn Mạnh Linh, Nguyễn Thị Đông, Triệu Hồng Thúy

1 Bài 1

- **Bài toán** được cấu tạo bởi hai thành phần cơ bản:
 - Thông tin vào (input): Cung cấp cho ta các dữ liệu đã có.
 - Thông tin ra (output): Những yếu tố cần xác định.
- **Thuật toán** là một dãy hữu hạn các thao tác đơn giản được sắp xếp theo một trình tự xác định dùng để giải một bài toán.
- **Giải bài toán bằng thuật toán** là quá trình từ thông tin đầu vào (input), dùng một dãy hữu hạn các thao tác đơn giản được sắp xếp theo một trình tự xác định để tìm được thông tin đầu ra (output) theo yêu cầu của bài toán đề ra.

2 Bài 2

2.1 Chia để trị

2.1.1 Ý tưởng

Phương pháp chia để trị dựa trên 2 thao tác chính:

- Chia (*divide*): phân rã bài toán ban đầu thành các bài toán con có kích thước nhỏ hơn, có cùng cách giải.
- Trị (*conquer*): giải từng bài toán con (theo cách tương tự bài toán đầu - đệ quy) rồi tổng hợp các lời giải để nhận kết quả của bài toán ban đầu.

Việc “Phân rã”: thực hiện trên miền dữ liệu (chia miền dữ liệu thành các miền nhỏ hơn tương đương 1 bài toán con)

2.1.2 Mô hình và lược đồ

Xét bài toán P trên miền dữ liệu R .

Gọi $D_C(R)$ là thuật giải P trên miền dữ liệu R .

Nếu R có thể phân rã thành n miền con: $R = R_1 \cup R_2 \cup \dots \cup R_n$

Với R_0 là miền đủ nhỏ để $D_C(R)$ có lời giải, ta có lược đồ giải thuật chia để trị như sau:

```

Divide_Conquer( $R$ ):
    if ( $R = R_0$ ):
        solve Divide_Conquer( $R_0$ )
    else
        divide  $R$  to  $R_1, R_2, \dots, R_n$ 
        for ( $i = 1, 2, \dots, n$ ):
            Divide_Conquer( $R_i$ )
        Combine and get result
end

```

2.1.3 Phân tích và đánh giá

Để phân tích và đánh giá độ phức tạp của thuật toán, ta thực hiện 2 công đoạn

- Xây dựng công thức truy hồi đánh giá độ phức tạp thuật toán
- Giải công thức truy hồi xác định độ phức tạp thuật toán.
 - Phép thế liên tiếp
 - Sử dụng định lý chính

2.1.4 Ví dụ

Ta xét bài toán *tìm kiếm nhị phân trên một mảng được sắp xếp*.

- Cho dãy n phần tử được sắp theo thứ tự (*tăng dần*) và một giá trị x bất kỳ. Kiểm tra xem phần tử x có trong dãy không?
- Phân tích ý tưởng: so sánh giá trị x với phần tử giữa của dãy tìm kiếm. Dựa vào giá trị này sẽ quyết định giới hạn tìm kiếm ở bước kế tiếp là nửa trước hay nửa sau dãy.
- Lược đồ của thuật toán như sau:

```

BinarySearch( $a, x, L, R$ ):
    // Search element  $x$  in array  $a$  from position  $L$  to  $R$ 
    if ( $L = R$ ):
        return ( $x = a_L$  ?  $L$  :  $-1$ )
    else
         $M = (L + R) / 2$ 
        if ( $x = a_M$ )
            return ( $M$ )

```

```

        else
            if (x < a_M)
                BinarySearch(a, x, L, R)
            else
                BinarySearch(a, x, M + 1, R)
            endif
        endif
    endif
end

```

Tính đúng của thuật toán

Ta chứng minh bằng quy nạp như sau

- Cơ sở quy nạp: $n = R - L + 1 = 1$ (dãy có 1 phần tử)
 - Câu lệnh `return (x = a_L ? L : -1)` trả về giá trị L hoặc -1
- Giả thiết quy nạp: Thuật toán đúng với mọi dãy có độ dài $n = R - L + 1$. Hay hàm `BinarySearch(a, x, L, R)` trả về đúng kết quả tìm kiếm x với mọi dãy có độ dài $1 \leq n' \leq n = R - L + 1$
- Tổng quát: Chứng minh thuật toán đúng với $n + 1 = R - L + 2$
 - Đặt $M = (L + R + 1)/2$, ta có $L \leq M \leq R$
 - Nếu $x = a_M$ thì kết quả trả về là M : đúng
 - Nếu $x < a_M$ thì kết quả là của bài toán tìm x trong tập a_L, \dots, a_M . Theo giả thiết quy nạp thì `BinarySearch(a, x, L, R)` đúng vì $1 \leq M - L + 1 = (R - L + 1)/2 + 1 \leq R - L + 1$
 - Tương tự với $x > a_M$

Độ phức tạp của thuật toán

$$T(n) = \begin{cases} 1 & \text{when } n = 1 \\ T(n/2) + 1 & \text{when } n > 1 \end{cases}$$

Do đó $T(n) = O(\log n)$

Source code: https://github.com/batman0911/dma_homework/blob/master/hw_01/src/main.ipynb

2.2 Quay lui

2.2.1 Ý tưởng

Ý tưởng của phương pháp Quay lui (Back tracking): Theo nguyên tắc vét cạn (xét qua tất cả các trường hợp có thể xảy ra để tìm kết quả), nhưng chỉ xét những trường hợp “khả quan”.

Tại mỗi bước, nếu có một lựa chọn được chấp thuận thì ghi nhận lại lựa chọn này và tiến hành các bước thử tiếp theo. Còn ngược lại không có lựa chọn nào thích hợp thì quay lại bước trước.

Phương pháp quay lui được sử dụng để giải bài toán liệt kê các cấu hình:

- Mỗi cấu hình được xây dựng bằng cách xác định từng phần tử
- Mỗi phần tử được chọn bằng cách thử các khả năng **có thể**
- Độ dài cấu hình tùy thuộc bài toán
 - Xác định trước: sinh dãy độ dài n
 - Không xác định trước: đường đi

2.2.2 Mô hình

Không gian nghiệm của bài toán (tập khả năng) $X = \{(x_1, x_2, \dots, x_n)\}$ gồm các cấu hình liệt kê có dạng (x_1, x_2, \dots, x_n) cần được xây dựng:

- Cho x_1 nhận lần lượt các giá trị có thể. Với mỗi giá trị thử gán cho x_1 thì:
- Cho x_2 nhận lần lượt các giá trị có thể. Với mỗi giá trị thử gán cho x_2 thì xét khả năng chọn $x_3, \dots, x_n, \Rightarrow$ cấu hình tìm được (x_1, x_2, \dots, x_n) .
- **Tổng quát:** Tại mỗi bước i : Xây dựng thành phần x_i
 - Xác định x_i theo khả năng v .
 - Nếu gặp điều kiện dừng ($i = n$ hoặc x_i thỏa mãn điều kiện dừng) thì ta có được một lời giải, ngược lại thì tiến hành bước $i+1$ để xác định x_{i+1}
 - Nếu không có một khả năng nào chấp nhận được cho x_i thì lại lùi lại bước trước để xác định lại thành phần x_{i-1} – sự quay lui

2.2.3 Lược đồ

Lược đồ phương pháp quay lui như sau:

```

Try(i)  $\equiv$  //Sinh thành phần thứ  $i$  của cấu hình
    for ( $v$  thuộc tập khả năng thành phần nghiệm  $x_i$ )
        if ( $x_i$  chấp nhận được giá trị  $v$ )
             $x_i = v$ ;
            <Ghi nhận trạng thái chấp nhận  $v$ >;
            if (gặp điều kiện dừng) // $i=n$  hoặc  $x_i$  thỏa mãn điều kiện dừng
                <xử-lí-nghiệm>;
            else //lời gọi sinh thành phần tiếp theo của cấu hình
                Try ( $i + 1$ )
            <Khôi phục trạng thái chưa chấp nhận  $v$ >;
        endif;
    endfor;
End.
```

Lời gọi ban đầu: Try(1)

2.2.4 Phân tích và đánh giá

Ta thực hiện 2 công đoạn phân tích và đánh giá thuật toán như sau:

- Công thức truy hồi

$$T(n) = \begin{cases} 1 & \text{khi } n \leq 1 \\ dT(n-1) + 1 & \text{khi } n > 1 \end{cases}$$

d: max (hoặc giá trị trung bình) lực lượng của tập khả năng của các thành phần nghiệm x_i

- Giải công thức truy hồi: $T(n) = O(d^n)$
 - Là trường hợp xấu nhất (vét cạn)
 - Trong triển khai thuật toán thời gian thực tế có thể giảm xuống
for (v thuộc tập khả năng thành phần nghiệm x_i)
if (x_i chấp nhận được giá trị v)

Tính đúng của thuật toán: Sự hiển nhiên trong tính đúng của thuật toán quay lui:

- Mỗi phần tử trong cấu hình nghiệm được chọn trong tập khả năng
for (v thuộc tập khả năng thành phần nghiệm x_i)
if (x_i chấp nhận được giá trị v)
 $x_i = v$;
- Tính dừng của lời gọi đệ quy và cho ra kết quả đúng
if (gặp điều kiện dừng) //i=n hoặc xi thỏa mãn mãn dk dừng
<xử-lí-nghiệm>;
else //lời gọi sinh thành phần tiếp theo của cấu hình
Try (i + 1)
- Đệ quy tiến Try(1)/ $x_1 \rightarrow$ Try(2)/ $x_2 \rightarrow \dots \rightarrow$ Try(m)/ x_m
 - Dủ thành phần nghiệm m=n (n - hữu hạn)
 - Đến thành phần thỏa mãn điều kiện tức là x_m thỏa mãn điều kiện nghiệm

2.2.5 Ví dụ

Ta xét bài toán *Liệt kê dãy k_ary có độ dài n*

- Phân tích
 - Input: n, k
 - Output: Các dãy $X = (x_1, x_2, \dots, x_n)$ trong đó $x_i = 0, 1, \dots, k-1$
 - Dùng giải thuật Try(i) để sinh giá trị x_i
 - Nếu i=n thì in giá trị nghiệm X, ngược lại sinh tiếp x_{i+1} bằng Try(i+1)

- Lược đồ giải thuật:

```

Try(i)  $\equiv$ 
    for (v=0..k-1) //v nhận giá trị từ 0 đến k-1
         $x_i = v$  ;
        if (i=n) printResult (X) ;
        else Try(i+1) ;
    endfor;
End.

```

Lời gọi ban đầu **Try(1)**

2.3 Quy hoạch động

2.3.1 Ý tưởng

- Quy hoạch động (Dynamic programming) là phương pháp giải các bài toán bằng cách kết hợp lời giải của các bài toán con theo nguyên tắc:
 - Giải tất cả các bài toán con (một lần)
 - Lưu lời giải của các bài toán vào một bảng
 - Phối hợp các bài toán con để nhận lời giải bài toán ban đầu
- Cách phát biểu khác: Một bài toán giải bằng quy hoạch động được phân rã thành các bài toán con và bài toán lớn sẽ được giải quyết thông qua các bài toán con này (bằng các phép truy hồi)
- Phương pháp quy hoạch động thường được dùng cho các bài toán tìm giá trị (hoặc giá trị tối ưu)

2.3.2 Các bước giải bài toán

1. Nhận dạng bài toán

Các bài toán có các đặc trưng sau thì có thể giải bằng QHD

- Bài toán có thể giải bằng đệ qui và tìm lời giải tối ưu.
- Bài toán có thể phân rã thành nhiều bài toán con mà sự phối hợp lời giải của các bài toán con sẽ cho ta lời giải của bài toán ban đầu.
- Quá trình tìm ra lời giải của bài toán ban đầu từ các bài toán con đơn giản hơn được thực hiện qua một số hữu hạn các bước có tính truy hồi.

2. Xây dựng công thức truy hồi

- Đưa bài toán về 1 dạng cơ bản, và triển khai ý tưởng của dạng bài toán đó để nhanh chóng nhận ra hướng thiết lập công thức truy hồi.
- Đây là bước khó nhất và cũng quan trọng nhất trong toàn bộ quá trình thiết kế thuật toán cho bài toán.

3. Xác định và xây dựng cơ sở QHD

- Dựa vào công thức truy hồi để nhận ra các bài toán cơ sở.
- Dựa vào ý nghĩa của công thức truy hồi để thiết lập giá trị cho cơ sở.

4. Dựng bảng phương án

- Dựa vào công thức truy hồi để tính giá trị các ô trong bảng phương án.
- Chú ý: bảng phương án có thể 1 chiều, 2 chiều hoặc nhiều hơn

5. Tìm kết quả tối ưu

- Xác định vị trí chứa kết quả tối ưu của bài toán trên bảng phương án.
- Chú ý: ngoài kết quả tối ưu, ô chứa kết quả tối ưu còn là điểm bắt đầu cho quá trình truy vết tìm nghiệm \Rightarrow lưu tọa độ của ô đó.

6. Truy vết liệt kê thành phần nghiệm

- Từ điểm bắt đầu là vị trí chứa kết quả tối ưu
- Truy ngược lại về điểm bắt đầu của nghiệm: có thể là những ô đầu tiên trong bảng phương án (bài toán cơ sở), có thể là ô của bảng phương án đạt giá trị đầu.

2.3.3 Ví dụ

1. Bài toán "Dãy con đơn điệu tăng dài nhất"

Bài toán: Tìm dãy con dài nhất của một dãy đã cho. Các phần tử có thể không liên tiếp.

Phân tích:

- Input: (a, n)
- Output: số lớn nhất các phần tử của dãy theo thứ tự tăng dần
- Hàm tối ưu $L(i)$: Độ dài dãy con đơn điệu tăng dài nhất đến phần tử i
Là độ dài các dãy con dài nhất đến j cộng 1 khi ghép thêm a_i vào sau, với điều kiện $j < i, a_j < a_i$
- Công thức truy hồi: $L(i) = \max L(j) + 1$ với $j < i, a_j < a_i$
- Cơ sở của thuật toán: $L(0) = 0; L(1) = 1$

```
def lis(_list):
    longest = [_list[0]]
    current = [_list[0]]
    for i in _list[1:]:
        if i >= current[-1]:
            current.append(i)
        else:
            if len(longest) < len(current):
                longest = current
            current = [i]
    if len(longest) < len(current):
        longest = current
    return longest
```

2. Bài toán "Xếp balo 0-1"

Bài toán: Có N đồ vật với trọng lượng và giá trị tương ứng (w_i, v_i) . Tìm cách cho các vật vào balo có trọng lượng W sao cho đạt giá trị cao nhất. Mỗi vật chỉ được chọn 1 lần

Phân tích:

- Input: n đồ vật, (w_i, v_i) $i=1..n$, Túi có trọng lượng tối đa W
- Output: V =tổng giá trị lớn nhất của các đồ vật vào balo
- Hàm tối ưu $dp[i, j]$: Giá trị lớn nhất khi chọn đồ vật từ 1 tới i với trọng lượng balo j ($i=0..n, j=0..W$)
 Nếu không chọn đồ vật thứ i thì: $dp[i, j] = dp[i-1, j]$
 Nếu chọn đồ vật thứ i thì: $dp[i, j] = dp[i-1, j-w_i] + v_i$ (Điều kiện $w_i \leq j$)
- Công thức truy hồi:
 $dp[i, j] = \text{Max}(dp[i-1, j], dp[i-1, j-w_i] + v_i)$
- Cơ sở quy hoạch động
 $dp[i, 0] = 0$
 $dp[0, j] = 0$

```
def solve(weights, values, capacity):
    items = list(zip(weights, values))
    def dp(i, cp):
        if i == len(items):
            return 0.0
        w, v = items[i]
        ans = dp(i + 1, cp)
        if cp >= w:
            ans = max(ans, dp(i + 1, cp - w) + v)
        return ans
    return int(dp(0, capacity))
```

3 Bài 3

Trong bài này chúng ta sẽ xem xét bài toán đóng hàng toàn cục 2 chuỗi DNA sử dụng phương pháp quy hoạch động

This homework answers the problem set sequentially.

1. Download the US Presidential Elections data set `uspresidentialelections.dta` from the course ILIAS site. Load the data set in R.

Copy your R Code to answer the question here.

2. Describe the dataset. What variables does it contain? How many observations are there? What time span does it cover?

Please type your answer here.

Put the right R command here.

3. *Compute measures of central tendency and variability of the variables **vote** and **growth** using R. Use the numerical measures of central tendency and variability discussed in class. Describe them in your own words and make a nice table. Plot the distribution of both variables using a boxplot and histogram. Make sure to make your plots as nice-looking as possible. Especially, include a title and label the axes.*

Your answer goes here

3 R commands here

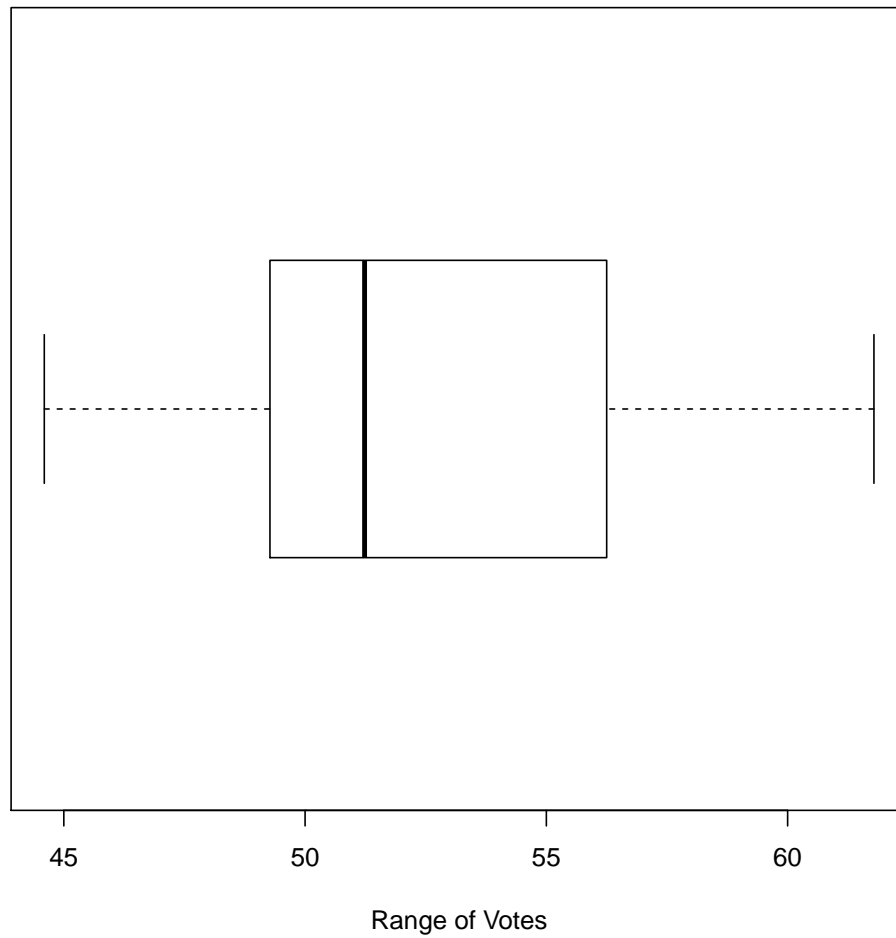
Potentially, your answer continues here.

4 R commands here

And more of your answer here.

And more space for your R commands.

```
# This is the code to produce the first boxplot.
pdf(file = "box1.pdf")
boxplot(us_data$vote, horizontal = T,
        main = "A Boxplot of the Variable Vote",
        names = "Vote",
        xlab = "Range of Votes")
dev.off()
```

A Boxplot of the Variable Vote

Hình 1: Boxplot of Incumbent Vote share

Variable	<i>Mean</i>	<i>Median</i>	<i>Mode</i>	<i>Var</i>	<i>SD</i>	<i>Range</i>	<i>IQR</i>
Vote	x	x	x	x	x	x	x
Growth	x	x	x	x	x	x	x

Bảng 1: Measures of central tendency and variability.

4. *Make a bar plot of the party affiliation of incumbent presidential candidates.*

Include the code for the bar plot and the plot here.

5. *During the presidential campaign in 1992, Bill Clinton's campaign coined the phrase "It's the economy, stupid!" Let's investigate the relationship between the economy and electoral success. Generate a nice-looking scatterplot of economic growth and vote share. Label the data points with the year of the election. Describe the pattern that you see in your own words.*

Include the code for the scatterplot as well as the plot here.

Then, describe the pattern you see. In the scatterplot we can see that...

R-Code

Finally, copy and paste the entire script here.