طراحی الگوریتم

(برنامه‌ریزی پویا)

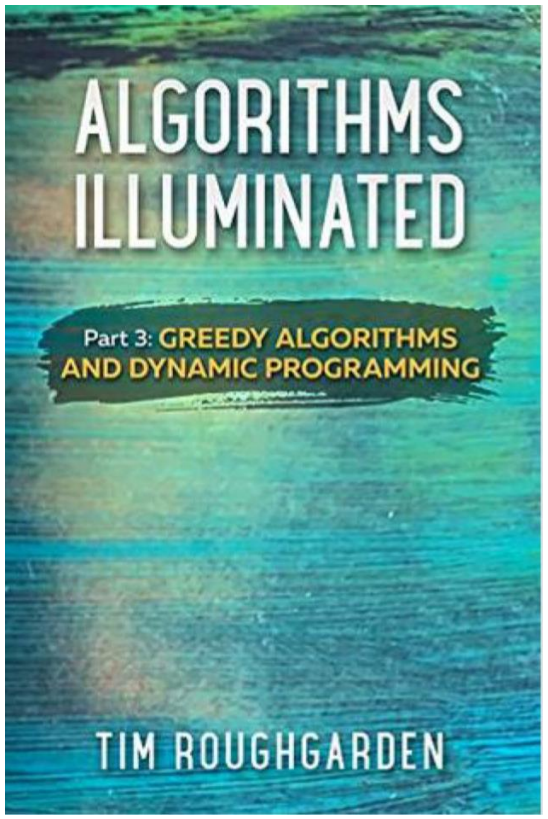
دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان

بهار ۹۹

نگاه کلی به برنامه‌ریزی پویا

- یک مجموعه نسبتا کوچک از زیر مساله‌ها را مشخص می‌نماییم.
- نشان دادن اینکه چگونه با داشتن جواب زیرمساله‌های کوچکتر می‌توان در زمان کمی جواب صحیح برای زیرمساله‌های بزرگتر را به دست آورد.
- چگونه می‌توان جواب صحیح نهایی را سریع از جواب تمام زیرمساله‌ها به دست آورد.

هم‌ترازی دنباله‌ها



فصل هفدهم، صفحه ۱۳۷

ورودی: دو دنباله X و Y روی یک الفبا و یک تابع جریمه.

هدف: یک هم‌ترازی برای X و Y با کمترین جریمه.

ساختار جواب بهینه

ساختار جواب بهینه

رویکرد برنامه‌ریزی پویا

NW

Input: strings $X = x_1, x_2, \ldots, x_m$ and $Y = y_1, y_2, \ldots, y_n$ over the alphabet $\Sigma = \{A, C, G, T\}$, a penalty α_{xy} for each $x, y \in \Sigma$, and a gap penalty $\alpha_{gap} \geq 0$.
Output: the NW score of X and Y.

```
// subproblem solutions (indexed from 0)
```
$A := (m+1) \times (n+1)$ two-dimensional array
```
// base case #1 (j = 0)
```
for $i = 0$ to m **do**
    $A[i][0] = i \cdot \alpha_{gap}$
```
// base case #2 (i = 0)
```
for $j = 0$ to n **do**
    $A[0][j] = j \cdot \alpha_{gap}$
```
// systematically solve all subproblems
```
for $i = 1$ to m **do**
    for $j = 1$ to n **do**
```
        // use recurrence from Corollary 17.2
```
        $A[i][j] :=$

$$\min \left\{ \begin{array}{ll} A[i-1][j-1] + \alpha_{x_i y_j} & \text{(Case 1)} \\ A[i-1][j] + \alpha_{gap} & \text{(Case 2)} \\ A[i][j-1] + \alpha_{gap} & \text{(Case 3)} \end{array} \right\}$$

return $A[m][n]$ `// solution to largest subproblem`

$X_i = x_1 x_2 \ldots x_i$

$Y_j = y_1 y_2 \ldots y_j$

نتیجه:

$P_{i,j} =$ جریمه کمترین همترازی X_i و Y_j باشد.

برای هر i از ۱ تا m و j از ۱ تا n

$$P_{i,j} = \min \begin{cases} ① \; P_{i-1,j-1} + \alpha_{x_m y_n} \\ ② \; P_{i-1,j} + \alpha_{gap} \\ ③ \; P_{i,j-1} + \alpha_{gap} \end{cases}$$

$$P_{i,0} = P_{0,i} = i \cdot \alpha_{gap}$$