# ДИНАМИЧНО ПРОГРАМИРАНЕ

## ПРИМЕРНО КОНТРОЛНО № 4 ПО "ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ" — СУ, ФМИ (ЗА СПЕЦИАЛНОСТ "КОМПЮТЪРНИ НАУКИ", 1. ПОТОК, ФЕВРУАРИ – ЮНИ 2019 Г.)

**Зад. 1.** Дадени са три масива $A[1...n]$, $B[1...n]$ и $C[1...n]$. Намерете най-голямата стойност на сбор от $n$ събираеми, който може да се образува, като се спазват следните правила:

1) $k$-тото събираемо е $A[k]$, $B[k]$ или $C[k]$;
2) не може да има две поредни събираеми от един и същи масив (ако например $k$-тото събираемо е $A[k]$, то $(k+1)$-ото събираемо е или $B[k+1]$, или $C[k+1]$, но не и $A[k+1]$).

Предложете итеративен алгоритъм. Опишете го на псевдокод като функция

`maxSum(A[1...n], B[1...n], C[1...n]: arrays of int): int`

с време $O(n)$ и динамична таблица с $O(n)$ клетки.

Демонстрирайте алгоритъма при

$$A = (2; 3; 5; 1; 8; 7; 6);$$
$$B = (6; 9; 8; 0; 5; 9; 7);$$
$$C = (4; 1; 2; 8; 4; 2; 5).$$

Оптимизирайте паметта до константен брой променливи от целочислен тип. Опишете оптимизирания алгоритъм на псевдокод.

**Решение:**

```
maxSum(A[1...n], B[1...n], C[1...n]: arrays of int): int
dyn['A'...'C'][1...n]: array of int
dyn['A'][1] ← A[1]
dyn['B'][1] ← B[1]
dyn['C'][1] ← C[1]
for ñ ← 2 to n do
  dyn['A'][ñ] ← A[ñ] + max(dyn['B'][ñ-1], dyn['C'][ñ-1])
  dyn['B'][ñ] ← B[ñ] + max(dyn['C'][ñ-1], dyn['A'][ñ-1])
  dyn['C'][ñ] ← C[ñ] + max(dyn['A'][ñ-1], dyn['B'][ñ-1])
return max(dyn['A'][n], dyn['B'][n], dyn['C'][n])
```

В клетката $dyn[X][\tilde{n}]$ стои най-големият възможен сбор от първите $\tilde{n}$ елемента, ако последното събираемо е от масива с име $X$.

Демонстрация на алгоритъма при

$$A = (2 ; 3 ; 5 ; 1 ; 8 ; 7 ; 6) ;$$
$$B = (6 ; 9 ; 8 ; 0 ; 5 ; 9 ; 7) ;$$
$$C = (4 ; 1 ; 2 ; 8 ; 4 ; 2 ; 5) :$$

| dyn | $\tilde{n} = 1$ | $\tilde{n} = 2$ | $\tilde{n} = 3$ | $\tilde{n} = 4$ | $\tilde{n} = 5$ | $\tilde{n} = 6$ | $\tilde{n} = 7$ |
|---|---|---|---|---|---|---|---|
| 'A' | 2 | 9 | 18 | 18 | 34 | 38 | (49) |
| 'B' | 6 | 13 | 17 | 18 | 31 | 43 | 45 |
| 'C' | 4 | 7 | 15 | 26 | 22 | 36 | 48 |

Отговорът на задачата е най-голямото число в последния стълб, тоест 49. Това е най-големият сбор, който можем да получим при спазване на правилата от условието. За да намерим кои събираеми образуват този сбор, трябва още при построяване на динамичната таблица да пазим във всяка клетка указател към по-голямата от двете сравнявани клетки от предишния стълб. Тръгвайки от най-голямата клетка в последния стълб и движейки се по тези указатели, получаваме път (оцветените в жълто клетки), който ни дава информация за събираемите на максималния сбор:

$$C[1] + B[2] + A[3] + C[4] + A[5] + B[6] + A[7] =$$
$$= 4 + 9 + 5 + 8 + 8 + 9 + 6 = 49.$$

Оптимизация по памет (но не и по време) можем да постигнем, като пазим само един стълб от динамичната таблица (но така губим пътя и събираемите).

```
maxSum(A[1...n], B[1...n], C[1...n]: arrays of int): int
oldMaxA ← A[ 1 ]
oldMaxB ← B[ 1 ]
oldMaxC ← C[ 1 ]
for ñ ← 2 to n do
  newMaxA ← A[ ñ ] + max(oldMaxB, oldMaxC)
  newMaxB ← B[ ñ ] + max(oldMaxC, oldMaxA)
  newMaxC ← C[ ñ ] + max(oldMaxA, oldMaxB)
  oldMaxA ← newMaxA
  oldMaxB ← newMaxB
  oldMaxC ← newMaxC
return max(newMaxA, newMaxB, newMaxC)
```

**Зад. 2.** В множеството $\{ a , b , c \}$ е въведено неасоциативно и некомутативно умножение**:**

| | × | втори множител | | |
|---|---|---|---|---|
| | | $a$ | $b$ | $c$ |
| първи множител | $a$ | $b$ | $c$ | $b$ |
| | $b$ | $b$ | $a$ | $a$ |
| | $c$ | $c$ | $c$ | $a$ |

Даден е низ $S[1...n]$, съставен от буквите $a$, $b$ и $c$. Ако $S$ се тълкува като произведение, можем ли да сложим скоби тъй, че $S$ да получи стойност $a$? Например при $S = abcab$ това е възможно**:** $(ab)((ca)b) = c(cb) = cc = a$**.** Но при $S = abb$ е невъзможно**:** $(ab)b = cb = c$**;** $a(bb) = aa = b$**.**

Предложете итеративен алгоритъм. Опишете го на псевдокод като функция `isA(`$S$`[1...`$n$`]: array of char): bool`

с време $O(n^3)$, без да извиквате наготово алгоритъма CYK**.**

Демонстрирайте своя алгоритъм при $S = bcaab$**.**

Решете задачата и като ползвате наготово алгоритъма CYK**.** За целта съставете безконтекстна граматика в нормална форма на Чомски**.**

**Решение:** Безконтекстна граматика (в нормална форма на Чомски), пораждаща изразите, които могат да имат стойност $a$ при подходящо поставяне на скоби**:**

$$A \to a$$
$$B \to b$$
$$C \to c$$
$$A \to BB$$
$$A \to BC$$
$$A \to CC$$
$$B \to AA$$
$$B \to AC$$
$$B \to BA$$
$$C \to AB$$
$$C \to CA$$
$$C \to CB$$

Стартовият символ е $A$**.** Първите три правила дефинират значението на нетерминалите. Останалите девет правила кодират таблицата за умножение.

Ако не викаме наготово алгоритъма CYK, можем да използваме идеята му.

```
isA(S[1...n]: array of char): bool
dyn[1...n][1...n]['a','b','c']: array of bool
for i ← 1 to n do
  for j ← i to n do
    dyn[ i ][ j ]['a'] ← false
    dyn[ i ][ j ]['b'] ← false
    dyn[ i ][ j ]['c'] ← false
for i ← 1 to n do
  dyn[ i ][ i ][ S[ i ]] ← true
for L ← 2 to n do
  for i ← 1 to n + 1 - L do
    j ← i + L - 1
    // разцепваме S[i...j] на всички възможни места:
    // S[i...k] и S[k+1...j]
    for k ← i to j - 1 do
      // таблицата за умножение
      for p ← 'a' to 'c' do
        for q ← 'a' to 'c' do
          if dyn[ i ][ k ][ p ] and dyn[ k + 1 ][ j ][ q ]
            dyn[ i ][ j ][ p × q ] ← true
return dyn[ 1 ][ n ]['a']
```

Идеята е, че dyn[ $i$ ][ $j$ ][ $p$ ] = true $\Leftrightarrow$ произведението $S[\, i \ldots j \,]$ може да приеме стойност $p$, където $p$ е някой от елементите $a$, $b$ или $c$.

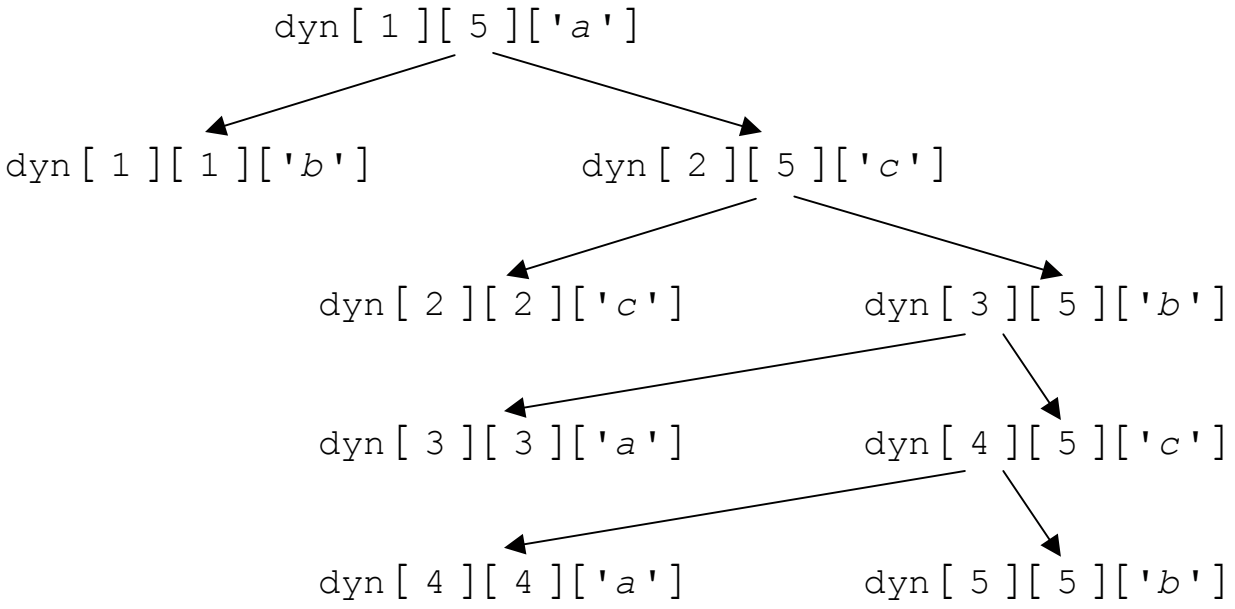
Демонстрация на алгоритъма при $S = bcaab$:

| dyn | $j = 1$ | $j = 2$ | $j = 3$ | $j = 4$ | $j = 5$ |
|---|---|---|---|---|---|
| $i = 1$ | $b$ | $a$ | $b$ , $a$ | $a$ , $c$ , $b$ | $a$ , $b$ , $c$ |
| $i = 2$ | | $c$ | $c$ | $c$ | $c$ , $a$ |
| $i = 3$ | | | $a$ | $b$ | $b$ , $a$ |
| $i = 4$ | | | | $a$ | $c$ |
| $i = 5$ | | | | | $b$ |

За удобство третото измерение е проектирано върху равнината на първите две. Тъй като в горния десен ъгъл присъства стойността $a$, то възможно е да поставим скоби така, че стойността на израза $S$ да бъде равна на $a$.

Ако искаме да намерим самото разположение на скобите, трябва във всяка клетка със стойност `true` да пазим указател към онези клетки, от които е получена тази стойност. С други думи, на предпоследния ред от псевдокода трябва в клетката `dyn[ i ][ j ][ p × q ]` да запазим два указателя към клетките `dyn[ i ][ k ][ p ]` и `dyn[ k + 1 ][ j ][ q ]`.

За да не претрупваме демонстрацията, ще покажем само дървото на израза, което се получава от клетката `dyn[ 1 ][ n ]['a']`.




Това дърво съответства на следното разположение на скоби: $S = b(c(a(ab)))$.

Действително, $S = b(c(a(ab))) = b(c(ac)) = b(cb) = bc = a$.

## СХЕМА ЗА ОЦЕНЯВАНЕ

**Задача 1** носи 2 точки, разпределени по 0,50 т. за всяка от следните стъпки:
— описание на алгоритъма на псевдокод;
— възстановяване на решението;
— демонстрация на алгоритъма върху конкретните входни данни;
— оптимизация на количеството използвана допълнителна памет.

**Задача 2** носи 2 точки, разпределени по 0,50 т. за всяка от следните стъпки:
— описание на алгоритъма на псевдокод;
— възстановяване на решението;
— демонстрация на алгоритъма върху конкретните входни данни;
— съставяне на безконтекстна граматика в нормална форма на Чомски.

**Оценката = 2 + броя на получените точки.**