DAA семинар 4

• Някои полезни суми:

$$\sum_{i=0}^{n} k^i \begin{cases} \asymp k^n, & k>1 \\ = n, & k=1 \\ \asymp 1, & k<1 \end{cases}$$

$$\sum_{i=1}^{n} i^k = \begin{cases} \Theta(n^{k+1}), & k>-1 \\ \Theta(\lg n), & k=-1 \\ \Theta(1), & k<-1 \end{cases}$$

$\sum_{i=1}^{n} 1 = n = \Theta(n)$ $\quad \sum_{i=1}^{n} \Theta(1) = \Theta(n)$ $\quad \sum_{i=c}^{n} = n - c + 1 = \Theta(n)$

$\sum_{i=c}^{n} \Theta(1) = \Theta(n)$ $\quad \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)$ $\quad \sum_{i=1}^{n} \Theta(i) = \Theta(n^2)$

$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3)$ $\quad \sum_{i=1}^{n} \Theta(i^2) = \Theta(n^3)$ $\quad \sum_{\substack{i=1 \\ i=i+k}}^{n} 1 = \lfloor \frac{n}{k} \rfloor = \Theta(n)$

$\sum_{\substack{i=1 \\ i=i+k}}^{n} \Theta(1) = \Theta(n)$ $\quad \sum_{i=1}^{n} \frac{1}{i} = \Theta(ln(n))$ $\quad \sum_{i=1}^{n} \Theta(\frac{1}{i}) = \Theta(ln(n))$

Анализ на сложността по време и памет на итеративни алгоритми

- Всяко число заема $\Theta(1)$ памет
- Всички прости операции с числа и паметта са $\Theta(1)$ (аритметични, сравнения, присвояване, достъп в масив и т.н)
- Търсим функция над големината на входа
- Ще мерим само допълнителната памет, т.е. без вход/изход.

Зад 1 Да се намери сложността по време и памет на следния алгоритъм.

```
Add1(n ∈ Z⁺)
1.   a ← 0
2.   for i ← 1 to n
3.     for j ← 1 to n
4.       a ← a+1
5.   return a
```

Решение:

$$T(n) = \Theta(1) + \sum_{i=1}^{n}\left(\Theta(1) + \sum_{j=1}^{n}\Theta(1)\right) = 1 + \sum_{i=1}^{n}\left(1 + \sum_{j=1}^{n} 1\right) = 1 + \sum_{i=1}^{n}(1+n) =$$

$$= 1 + n + n(n+1) = \underline{\underline{\Theta(n^2)}}$$

$S(n) = \Theta(1)$ // може и $M(n)$

Допълнение към зад. 1: Нека ред 3 е for $j \leftarrow i+1$ to n

$$T(n) = \Theta(1) + \sum_{i=1}^{n}\left(\Theta(1) + \sum_{j=i+1}^{n}\Theta(1)\right) = 1 + \sum_{i=1}^{n}\left(1 + \sum_{j=1}^{n} 1 - \sum_{j=1}^{i+1-1} 1\right) =$$

$$// \sum_{i=a}^{b} f(i) = \sum_{i=1}^{b} f(i) - \sum_{i=1}^{a-1} f(i)$$

$$= 1 + \sum_{i=1}^{n}(1+n-i) = 1 + \sum_{i=1}^{n}(n+1) - \sum_{i=1}^{n} i = 1 + n(n+1) - \frac{n(n+1)}{2} =$$

$$= 1 + n^2 + n - \frac{n^2}{2} - \frac{n}{2} = \frac{1}{2}n^2 + \frac{1}{2}n + 1 = \Theta(n^2)$$

Зад. 2 Какво прави следният алгоритъм? Анализирайте сложността му по време.

Alg2 $(A[1..n][1..n], B[1..n][1..n])$, $n \geq 1$

```
1.   C[1..n][1..n] ← празен масив
2.   for i ← 1 to n
3.     for j ← 1 to n
4.        s ← 0
5.        for k ← 1 to n
6.           s ← s + A[i][k].B[k][j]
7.        C[i][j] ← s
8.   return C
```

Решение: Алгоритъмът умножава две квадратни матрици.

$$T(n) = \Theta(1) + \sum_{i=1}^{n}\left(\Theta(1) + \sum_{j=1}^{n}\left(\Theta(1) + \sum_{k=1}^{n}\Theta(1)\right)\right) =$$

$$= 1 + \sum_{i=1}^{n}\left(1 + \sum_{j=1}^{n}\left(1 + \sum_{k=1}^{n} 1\right)\right) = 1 + \sum_{i=1}^{n}\left(1 + \sum_{j=1}^{n}(1+n)\right) =$$

$$= 1 + \sum_{i=1}^{n}(1 + n(n+1)) = 1 + n + n \cdot n(n+1) = \Theta(n^3)$$

Ако разгл. размера на входа $m = n^2 \Rightarrow T(m) = \Theta(m\sqrt{m})$

Зад 3 Да се изчисли сложността по време на следният фрагмент код:

```
for(int i=1; i<=n; i++)
  for(int j=1; j<=n; j+=i)
     print("a");
```

Решение: $T(n) = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j=j+i}}^{n} 1 = \sum_{i=1}^{n} \left\lfloor \frac{n}{i} \right\rfloor \asymp \sum_{i=1}^{n} \frac{n}{i} = n\sum_{i=1}^{n} i = n.\lg(n) =$

$= \Theta(n\lg(n))$

Зад 4 Изследвайте коректността и сложността на следния алгоритъм:

```
   Fib(n≥0)
1    if n<2
2      return n
3    current ← 1, next ← 1
4    for i ← 2 to n
5       next ← next + current
6       current ← next − current
7    return current
```

Решение: Ще покажем, че $\overbrace{Fib(n) = F_n}^{(*)}$ – n-тото число на Фибоначи.

Инв. При всяко достигане на ред 4 е вярно, че:

$current = F_{i-1} \qquad next = F_i$

– База: При първото достигане на ред 4: $i=2$, т.е. $F_i = F_2$, $F_{i-1} = F_1$
От ред 3: $current = 1 = F_1 = F_{i-1}$ и $next = 1 = F_2 = F_i$

$\Rightarrow$ инв. е верен

– Поддръжка: Нека инвариантът е изпълнен за някое достигане на ред 4, което не е последно.

След изпълнение на ред 5: $next' = next + current \overset{ИП}{=} F_i + F_{i-1} = F_{i+1}$

След ред 6: $current' = next' - current \underset{\text{и горното}}{\overset{ИП}{=}} F_{i+1} - F_{i-1} = F_i$

При следващо достигане на ред 4: $i' = i+1$

$\Rightarrow next' = F_{i+1} = F_{i'}$ и $current' = F_i = F_{i'-1}$

$\Rightarrow$ Инв. е верен.

– Терминация: Нека инв. е верен за последното достигане на ред 4.
Тогава $i = n+1 \Rightarrow current = F_n$ и $next = F_{n+1}$

Последно достигане има, защото i се увеличава с 1 на всяка стъпка и ще достигне горната граница след краен брой итерации.

□ Инв.

Док на твърдението (*):

- I сл: $n < 2 \rightarrow$ Алгоритъмът изпълнява ред 2 и връща съответното число на Фибоначи: $F_0 = 0$ $F_1 = 1$

- II сл: $n \geq 2$. Ще се изпълни другата част от кода. Тогава от доказания инвариант, при достигане на ред 7 $current = F_n$, което дава върнато

$\Rightarrow$ Твърдението е вярно, т.е $Fib(n) = F_n$

Сложност по време: $T(n) = \Theta(1) + \sum_{i=2}^{n} \Theta(1) = 1 + \sum_{i=2}^{n} 1 = \Theta(n)$

Зад 5 Докажете коректността на алгоритъма на Kadane. Намерете сложността му по време и памет.

```
Kadane(A[1..n], където n>0 и A[1]≥0)
1. curr ← A[1], best ← A[1]
2. for i ← 2 to n
3.     curr ← curr + A[i]
4.     if curr < 0
5.         curr ← 0
6.     if curr > best
7.         best ← curr
8. return best
```

Тв. Алгоритъмът връща максимална сума на подмасив на A.

Решение: Ще докажем твърдението чрез следния инвариант:

Инв: При всяко достигане на ред 2 curr съдържа максимална сума на някой суфикс на $A[1..i-1]$, а best съдържа максимална сума на някой непразен подмасив на $A[1..i-1]$

– База: При първото достигане на ред 2: $i = 2$, т.е. разглеждаме $A[1..1]$ – съдържа единствен елемент
$\Rightarrow$ A има единствен непразен подмасив със сума $A[1]$ и два суфикса – с 0 и 1 елемента – този с макс. сума е вторият, т.е със сума $A[1]$.

От ред 1 curr и best съдържат именно тази стойност.
$\Rightarrow$ Инв. е изпълнен.

- Поддръжка: Нека инвариантът е изпълнен за някое достигане на ред 2, което не е последно.

• Суфиксите на $A[1..i]$ са или празният, или $A[i]$, добавен към някой суфикс на $A[1..i-1]$. (*)

При достигането на ред 3 имаме, че curr съдържа макс. сума на някой суфикс на $A[1..i-1]$. След ред 3, curr' е максимална сума на $A[i]$, добавен към суфикс на $A[1..i-1]$.

След редове 4 и 5 curr' съдържа макс. сума на суфикс измежду (*).

Ако $A[i]$ „допринася" за макс. сума, тоест условието на ред 4 е лъжа, то curr' съдържа макс. сума на непразен суфикс на $A[1..i]$.
В противен случай разглеждаме празния суфикс със сума 0.

• Непразните подмасиви на $A[1..i]$ са или:
1) $A[i]$, добавен към суфикс на $A[1..i-1]$, или
2) някой непразен подмасив на $A[1..i-1]$
$\}$ (**)

Но 1) е същото като суфикс на $A[1..i]$, а от горното имаме, че curr' съдържа макс. сума на тях.

От ИП имаме също, че best съдържа макс. сума на 2.

На редове 6 и 7 взимаме по-голямото от 2-те (Очевидно?) $\Rightarrow$ новата стойност best' съдържа макс. сума на (**)

При следващо достигане на ред 2, за новата стойност на i, инвариантът остава верен.

- Терминация: При последното достигане на ред 2 $i=n+1$, следователно curr съдържа макс. сума на суфикс $A[1..n]$, а best на непразен подмасив на $A[1..n]$.

При достигане на ред 8 от инв. знаем, че best съдържа макс. сума на непразен подмасив на $A[1..n]$; тя е >0, заради $A[1]>0$
$\Rightarrow$ Тази сума е максимална измежду сумите на всички подмасиви на A (вкл. празен).
Тогава твърдението е вярно, защото връща точно тази сума.

$$T(n) = \Theta(1) + \sum_{i=2}^{n} \Theta(1) = \Theta(n)$$

$$S(n) = \Theta(1)$$