ДАА Семинар 5

Рекурентни уравнения

зад 1 Даден е следният алгоритъм:

```
Alg(a, n): // a∈ℝ; n∈ℕ₀
1. if n = 0 then
2.     return 1;
3. if n ≡ 0 (mod 2) then
4.     return Alg(a*a, n/2);
5. else
6.     return a*Alg(a, n-1);
```

а) Какво връща Alg(a,n)

б) Да се докаже твърдението от а)

Решение:

а) При извикване $Alg(a,0)$ се връща 1

При извикване $Alg(a,1)$ се връща $a * Alg(a,0) = a$.

При извикване $Alg(a,2)$ се връща $Alg(a*a, 1) = a*a = a^2$

От тези наблюдения твърдим, че $Alg(a,n)$ връща a^n

б) Alg е рекурсивен алгоритъм, който извършва константна работа извън рекурсивните извиквания - ще използваме единствено пълна индукция за доказателството за коректност

1) База: $n=0$ ✓

В този случай $Alg(a,0)$ връща 1 //изпълнява се инструкцията на ред 2

2) Допускане: Нека е изпълнено следното: $(\forall m \leq n)(Alg(a,m) = a^m)$

3) Стъпка:

Ще докажем, че $Alg(a, n+1)$ връща a^{n+1}. Имаме следните два изчерпателни случая:

1сл) $n+1 \equiv 0 \pmod 2$

При изпълнение на $Alg(a, n+1)$ в този случай, тъй като $n+1 > 0$ и $n+1 \equiv 0 \pmod 2$, то ще се изпълни инструкцията на ред 4, тоест ще се върне като резултат $Alg(a*a, \frac{n+1}{2})$, което връща стойността $[a^2]^{\frac{n+1}{2}} = a^{n+1}$. Последното е в сила от допускането. (П)

2сл) $n+1 \equiv 1 \pmod 2$

При извикване на Alg(a, n+1) в този случай изпълнението отива на ред 6, където Alg(a, n) * a бива връщано като резултат. От допускането имаме, че Alg(a, n) връща a^{n}. Следователно, Alg(a, n+1) връща $a * a^{n} = a^{n+1}$ (o)

От (□) и (o) следва, че твърдението е изпълнено.

Рекурентни уравнения

Описват асимптотичната сложност по време на много алгоритми, изградени по схемата разделяй и владей

„Правилото" за съставяне на подходящо рекурентно уравнение, изразяващо сложността на рекурсивен алгоритъм е следното:

Лявата страна, T(n), изразява работата на алгоритъма върху вход с размер n, а дясната страна има по едно T(m), ~~кх~~ за всяко рекурсивно викате с подходящ аргумент m, събрано с израз, отразяващ работата на алгоритъма извън рекурсивните викания.

// Пример

```
Fact(n) // n∈N₀
1. if n < 2
2.     return 1
3. else
4.     return n*fact(n-1)
```

Рекурентно уравнение, изразяващо времето за работа на fact(n), е $T(n) = \underbrace{T(n-1)}_{\text{имаме едно рекурсивно извикване на ред 4.}} + \underbrace{\theta(1)}_{\text{допълнителната константна работа}} = T(n-1) + 1$

// Пример - от домашно 2023

Да се намери асимптотичната сложност на следния програмен фрагмент като функция на n.

```
int f(int n) {
int i, t=0;
if(n<2) return 2;
t += f(n/3);
for (i=2; i<n; i*=2)
    t++;
t *= f(n/3);
return t; }
```

Решение:

За стойности $n: n<2$ функцията работи за константно време.

В едно изпълнение на f се извършват две рекурсивни извиквания със стойност $\frac{n}{3}$: f(n/3)

Допълнителната работа е $\asymp \theta(1) + \sum_{\substack{i=2 \\ i*=2}}^{n} 1 \overset{(\Delta)}{=} \theta(1) + \sum_{k=1}^{\log n} 1 \asymp \theta(\log n)$

//(Δ) Итераторът i заема последователно стойностите $2, 4, 8, 16, \dots, n$

Тоест, $i \in \{2^1, 2^2, 2^3, 2^4, \dots, 2^{\log n}\} =: A$, $|A| = \log n$

ако n е точна степен на 2, но за асимптотиката е без значение

Следователно, сложността по време може да се опише със следното рекурентно уравнение:

$$T(n) = 2T\left(\frac{n}{3}\right) + \theta(\log n)$$ - трябва да се реши.

- Решение на рекурентно уравнение е затворена формула (без поява на функционалния символ отдясно на равенството), описваща сумата редица/общия общ член (разглеждаме целочислени ст-ти n)
- Асимптотиката на решението НЕ зависи от началните ~~случа~~ условия в случаите, които ще разглеждаме
- Няма единствен формален универсален метод за решаване на рекурентни уравнения

Методи за решаване

1) Чрез развиване } неформални
2) Чрез дърво на рекурсията
3) По индукция, имайки първоначална хипотеза
4) Чрез характеристично уравнение } не покриват всички видове
5) Чрез Мастър теорема

зад ① Да се намери асимптотиката на

$T(n) = T(n-1) + n$ — срез развиване + индукция

Решение:

неформално

$$T(n) = \underbrace{T(n-1)} + n = \underbrace{T(n-2) + (n-1)} + n = \underbrace{T(n-3) + (n-2)} + (n-1) + n = \dots$$

$$= T(0) + 1 + 2 + \dots + (n-2) + (n-1) + n = T(0) + \frac{n(n+1)}{2} = \underset{\theta(1)}{T(0)} + \theta(n^2) = \theta(n^2)$$

↳ предположение: $T(n) = \theta(n^2)$ (П)

За да бъде формално решението, ще докажем (П) по индукция:

Ще докажем поотделно $T(n) = O(n^2)$ и $T(n) = \Omega(n^2)$

База няма, тъй като игнорираме началните условия - не влияят на асимптотиката

1) $T(n) = O(n^2)$

Ще покажем, че съществува $c > 0$ и n_0, такова, че за всяко $n \geq n_0$ е в сила:

$T(n) \leq c.n^2$

1.1) Индуктивно предположение

Допускаме, че е изпълнено $T(n-1) \leq c(n-1)^2$

1.2) Индуктивна стъпка

Ще докажем, че е изпълнено за n, тоест, че $T(n) \leq cn^2$

$$T(n) \overset{def}{=} T(n-1) + n \overset{ИП}{\leq} c(n-1)^2 + n = cn^2 - 2cn + c + n \overset{(*)\, n \geq c}{\leq} cn^2 - 2cn + 2n = cn^2 + 2n(1-c)$$
$$= cn^2 + 2n(1-c) \overset{2n(1-c) \leq 0}{\leq} cn^2$$

Последното неравенство е в сила когато:

$2n(1-c) \leq 0 \iff 1 - c \leq 0 \iff c \geq 1$. Избираме $c = 1$.

Неравенство (*) е в сила когато $n \geq c$.

Тоест, свидетели са $c = 1$ и $n_0 \geq 1$

$\Rightarrow$ $(\forall n \geq n_0)(T(n) \leq cn^2)$ е изпълнено $\Rightarrow$ $T(n) = O(n^2)$ (□)

↳ // $T(n) \geq 0$, тъй като вс. рекурсивни алгоритми, които ще разглеждаме не работят са отрицателно време

2) $T(n) = \Omega(n^2)$

Ще покажем, че съществува $c>0$ и n_0, такива че за всяко $n \geq n_0$ е в сила: $cn^2 \leq T(n)$

2.1) Индуктивно предположение

Допускаме, че е изпълнено $c(n-1)^2 \leq T(n-1)$

2.2) Индуктивна стъпка

Ще докажем, че е изпълнено за n, тоест, че $cn^2 \leq T(n)$

$$T(n) \overset{def}{=} T(n-1) + n \overset{ИП}{\geq} c(n-1)^2 + n = cn^2 - 2cn + c + n \overset{(□)\; c=\frac{1}{2}}{=} \frac{1}{2}n^2 - n + c + n =$$

$$= \frac{1}{2}n^2 + c \overset{c>0}{\geq} \frac{1}{2}n^2 = cn^2$$

Неравенство (□) е в сила при $c \leq \frac{1}{2}$ - избираме $c = \frac{1}{2}$

Горните неравенства са в сила за всяко $n \geq 0$. Избираме $n=0$

Тоест, свидетели са $c = \frac{1}{2}$ и $n_0 = 0$

$\Rightarrow$ $(\forall n \geq n_0)(T(n) \geq cn^2)$ е изпълнено $\Rightarrow$ $T(n) = \Omega(n^2)$ (○)

От (□) и (○) $\Rightarrow$ $T(n) = \theta(n^2)$ □.

Зад ② Да се намери асимптотиката на

$T(n) = T(n-1) + \frac{1}{n}$ чрез развиване

Решение:

$$T(n) = T(n-1) + \frac{1}{n} = T(n-2) + \frac{1}{n-1} + \frac{1}{n} = T(n-3) + \frac{1}{n-2} + \frac{1}{n-1} + \frac{1}{n} = \dots =$$

$$= T(0) + \frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{n-2} + \frac{1}{n-1} + \frac{1}{n} = T(0) + \sum_{i=1}^{n} \frac{1}{i} \asymp \underset{const}{T(0)} + \lg n \asymp \lg n$$

Зад. 3 Да се намери асимптотиката на

$T(n) = 2T(\frac{n}{2}) + 1$ чрез развиване + индукция.

Решение:

$$T(n) = 2T(\underbrace{\tfrac{n}{2}}) + 1 = 2\left[\underbrace{2T(\tfrac{n}{4}) + 1}\right] + 1 = 2\left[2\left[\underbrace{2T(\tfrac{n}{8}) + 1}\right] + 1\right] + 1 = \dots =$$

$$= 2\left[2\left[2\left[\cdots\left[2T(1) + 1\right]\cdots\right] + 1\right] + 1\right] + 1 = 2^{\lg n} T(1) + 2^{\lg n - 1} + 2^{\lg n - 2} + \dots + 2^2 + 2^1 + 2^0 \asymp$$

$$\asymp 2^{\lg n} + 2^{\lg n - 1} + 2^{\lg n - 2} + \dots + 2^2 + 2^1 + 2^0 = 2^{\lg n} \underbrace{\sum_{i=1}^{\lg n} \frac{1}{2^i}}_{\text{ограничена от константите}} \asymp 2^{\lg n} = n$$

Твърдим, че $T(n) \asymp n$ (☆)

Ще докажем по индукция, че $T(n) = O(n)$ и $T(n) = \Omega(n)$

1) $T(n) = O(n)$

Търсим $c > 0, n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : 0 \leq T(n) \leq cn$

1.1) Индуктивно предположение

Допускаме, че е изпълнено $T(\frac{n}{2}) \leq c\frac{n}{2}$

1.2) Индуктивна стъпка

$T(n) = 2T(\frac{n}{2}) + 1 \overset{\text{ИП}}{\leq} 2c\frac{n}{2} + 1 = cn + 1 \not\leq cn$ → Няма такива c и n_0.

Ще засилим индуктивното предположение:

Допускаме, че е изпълнено $T(\frac{n}{2}) \leq c\frac{n}{2} - b$; $b, c > 0$

Тогава $T(n) = 2T(\frac{n}{2}) + 1 \overset{\text{ИП}}{\leq} 2(c\frac{n}{2} - b) + 1 = cn - 2b + 1 \overset{?}{\leq} cn - b$

Последното неравенство е в сила когато $-2b + 1 \leq -b$

$\Leftrightarrow b \geq 1$. и за всяко $c > 0$ и $n \in \mathbb{N}_0$

Доказахме, че $\exists n_0 \exists c \exists b : \forall n \geq n_0 : 0 \leq T(n) \leq cn - b$

От друга страна $cn - b \leq cn$, при $b \geq 1$ за всеки $\overset{0}{n}$ и $\overset{0}{c}$

$\Rightarrow \exists n_0 \exists c \forall n \geq n_0 : 0 \leq T(n) \leq cn \Rightarrow T(n) = O(n)$ (□)

2) $T(n) = \Omega(n)$

Търсим $c > 0; n_0 \in \mathbb{N}_0 : \forall n \geq n_0 : 0 \leq cn \leq T(n)$

2.1) Индуктивно предположение

Допускаме, че е изпълнено $c(\frac{n}{2}) \leq T(\frac{n}{2})$ за някое n

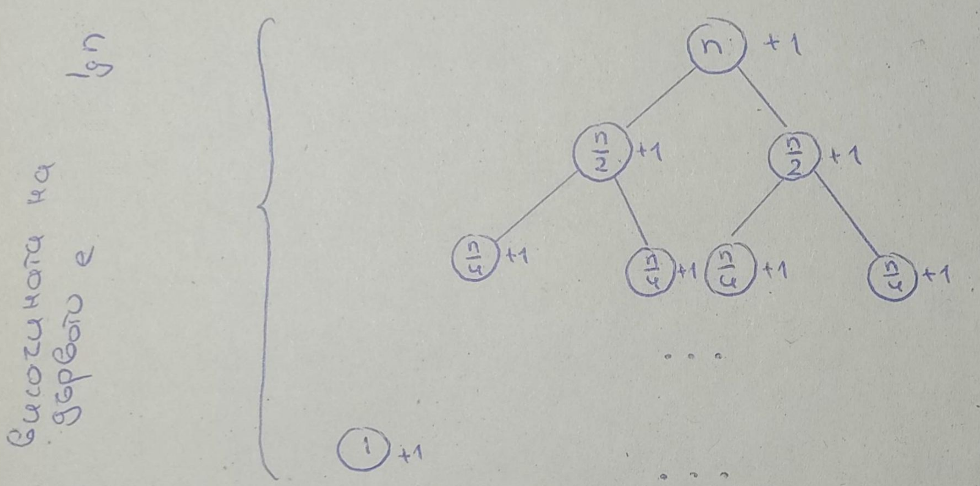
2.2) Индуктивна стъпка

$$T(n) = 2T\left(\frac{n}{2}\right) + 1 \overset{ИП}{\geq} 2\left(c\left(\frac{n}{2}\right)+1\right) = cn+1 \geq cn \quad , \forall c>0, \forall n>0$$

$$\Rightarrow T(n) = \Omega(n) \quad (o)$$

От (D) и (o) $\Rightarrow T(n) = \Theta(n)$

Вместо с развиване, твърдение (*) може да бъде направено чрез дърво на рекурсия - върховете ще са отбелязани с големината на входа на текущото изпълнение, отдясно на всеки връх пишем "+k", където k е работата, която се извършва извън рекурсивното извикване на текущото изпълнение:



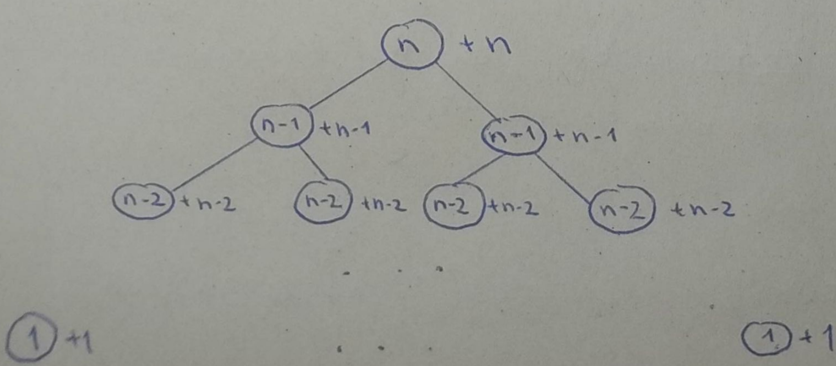
Може да сумираме работата на целия алгоритъм по нива:

$$1+2+4+\dots+n = 2^0+2^1+2^2+\dots+2^{\lg n} = \sum_{i=1}^{\lg n} 2^i = 2^{\lg n}\sum_{i=1}^{\lg n}\frac{1}{2^i} \approx 2^{\lg n} = n$$

Зад 4. Да се намери асимптотиката на

$T(n) = 2T(n-1) + n$ чрез дърво на рекурсията + индукция

Решение:



Сумираме работата на целия алгоритъм по нива:

$$1.n + 2.(n-1) + 4(n-2) + \dots + 2^{n-1}.1 = 2^0.n + 2^1(n-1) + 2^2(n-2) + \dots + 2^{n-1}.1 =$$

$$= 2^0(n-0) + 2^1(n-1) + 2^2(n-2) + \dots + 2^{n-1}(n-(n-1)) = \sum_{i=0}^{n-1} 2^i.(n-i) \quad (\square)$$

Разглеждаме сумата $(\square)$

$$\sum_{i=0}^{n-1} 2^i(n-i) = 2^{n-1}\sum_{i=0}^{n-1}\frac{(n-i)}{2^{i+n+1}} = 2^{n-1}\sum_{i=1}^{n}\frac{i}{2^i}$$

$$\sum_{i=0}^{n-1} 2^i(n-i) = \sum_{i=1}^{n} 2^{n-i}.i = 2^{n-1}\underbrace{\sum_{i=1}^{n}\frac{i}{2^{i-1}}}_{\text{сходяща сума}} \asymp 2^{n-1} \asymp 2^n \quad \square$$

Твърдим, че $T(n) \asymp 2^n$ $(\square)$

Ще докажем, че $T(n) = O(2^n)$ и $T(n) = \Omega(2^n)$ по индукция

1) $T(n) = O(2^n)$

Търсим $c>0,\ n_0 \in \mathbb{N}$: $(\forall n \ge n_0)(0 \le T(n) \le c2^n)$ е изпълнено

1.1) Индуктивно предположение:

Допускаме, че е в сила $T(n-1) \le c2^{n-1}$

1.2) Индуктивна стъпка

$$T(n) \overset{def}{=} 2T(n-1) + n \overset{ип}{\le} 2(c2^{n-1}) + n = c2^n + n \overset{?}{\nleq} c2^n$$ за никое $c>0$ и $n>0$

Ще засилим твърдението:

Допускаме, че е в сила $T(n-1) \le c2^{n-1} - b$

$$T(n) \overset{def}{=} 2T(n-1) + n \overset{ип}{\le} 2(c2^{n-1} - b) + n = c2^n - 2b + n \overset{?}{\nleq} c2^n - b$$

Последното неравенство е изпълнено при $-2b + n \le -b \iff$

$n \le b$, но b е константа $\times$

тоест, не съществува такова b и n

Засилваме:

Допускаме, че $T(n-1) \le c2^{n-1} - b(n-1)$

$$T(n) \overset{def}{=} 2T(n-1) + n \overset{ип}{\le} 2(c2^{n-1} - b(n-1)) + n = c2^n - 2b(n-1) + n =$$

$$= c2^n - 2nb + 2b + n \overset{?}{\le} c2^n - bn$$

Последното неравенство е в сила когато

$-2nb + \underline{2b+n} \le -bn \iff \underline{2b+n} \le bn$ Последното е в сила, например за $b=2$ и $n \ge 2.b = 4$

$\Rightarrow$ $\exists c>0,\ \exists n_0 = 4: (\forall n \ge n_0)$

$\Rightarrow$ $\exists c>0, \exists n_0=4, \forall n \ge n_0$: $T(n) \le c2^n - 2n \le c2^n$ $\Rightarrow$ $T(n) = O(2^n)$

// За засилването може да прилагаме следния метод:

1) $T(n) \le cf(n)$ за $c>0, b>0$ - const

2) $T(n) \le cf(n)+b$

3) $T(n) \le cf(n)+bn$ → ако някое не работи, пробваме следващите

4) $T(n) \le cf(n)+bn^2$

...

2) $T(n) = \Omega(2^n)$

Търсим $c>0$, $n_0 \in \mathbb{N}$, $\forall n \ge n_0$: $0 \le c2^n \le T(n)$

2.1) Индуктивно предположение

Допускаме, че $T(n-1) \ge c2^{n-1}$

2.2) Индуктивна стъпка

$$T(n) \overset{def}{=} 2T(n-1)+n \overset{ИП}{\ge} 2c2^{n-1}+n = c2^n+n \ge c2^n$$, за всяко c

$\Rightarrow T(n) = \Omega(2^n)$

$\Rightarrow T(n) = \Theta(2^n)$ □