Первый курс, осенний семестр 2015/16
Конспект лекций по алгоритмам

Собрано 17 сентября 2015 г. в 13:11

Содержание

Лекция по алгоритмам #1
Асимптотика

12 сентября

1.1. $\mathcal{O}$-обозначния

Def 1.1.1. $f = \mathcal{O}(g) \quad \exists N > 0, C > 0 \colon \forall n \geq N, f(n) \leq C \cdot g(n)$

Def 1.1.2. $f = \Omega(g) \quad \exists N > 0, C > 0 \colon \forall n \geq N, f(n) \geq C \cdot g(n)$

Def 1.1.3. $f = o(g) \quad \forall C > 0 \exists N > 0 \colon \forall n \geq N, f(n) \leq C \cdot g(n)$

Def 1.1.4. $f = \omega(g) \quad \forall C > 0 \exists N > 0 \colon \forall n \geq N, f(n) \geq C \cdot g(n)$

Def 1.1.5. $f = \Theta(g) \quad \exists N > 0, C_1 > 0, C_2 > 0 \colon \forall n \geq N, C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$

Следствие 1.1.6. $f = \Theta(g) \Leftrightarrow g = \Theta(f)$

Следствие 1.1.7. $f = \mathcal{O}(g), g = \mathcal{O}(f) \Leftrightarrow f = \Theta(g)$

Следствие 1.1.8. $f = \Omega(g) \Leftrightarrow g = \mathcal{O}(f)$

Следствие 1.1.9. $f = \omega(g) \Leftrightarrow g = o(f)$

Следствие 1.1.10. $f = \mathcal{O}(g), g = \mathcal{O}(h) \Rightarrow f = \mathcal{O}(h)$

Следствие 1.1.11. Обобщение: $\forall \beta \in \{\mathcal{O}, o, \Theta, \Omega, \omega\} \colon f = \beta(g), g = \beta(h) \Rightarrow \boxed{f = \beta(h)}$

1.2. $\mathcal{O}$-обозначния через пределы

Def 1.2.1. $f = o(g)$ *Определение через предел:* $\lim\limits_{n \to +\infty} \frac{f(n)}{g(n)} = 0$

Def 1.2.2. $f = \mathcal{O}(g)$ *Определение через предел:* $\overline{\lim\limits_{n \to +\infty}} \frac{f(n)}{g(n)} < +\infty$

Здесь необходимо пояснение: $\overline{\lim\limits_{n \to +\infty}} f(n) = \lim\limits_{n \to +\infty} (\sup\limits_{x \in [n..+\infty]} f(x))$, где sup – верхняя грань.

1.3. Суммы и интегралы

Lm 1.3.1. $\forall f(x) \nearrow [a..a+1] \Rightarrow f(a) \leq \int\limits_a^{a+1} f(x)dx \leq f(a+1)$

Lm 1.3.2. $\forall f(x) \nearrow [a..b] \Rightarrow \sum\limits_{i=a}^{b} f(i) \leq \int\limits_a^{b+1} f(x)dx$

Доказательство. Сложили неравенства из 1.3.1 ■

Lm 1.3.3. $\forall f(x) \nearrow [a..b], f > 0 \Rightarrow \int\limits_a^{b} f(x)dx \leq \sum\limits_{i=a}^{b} f(i)$

Доказательство. Сложили неравенства из 1.3.1, выкинули $[a-1, a]$ из интеграла. ■

Lm 1.3.4. $\forall f(x) \nearrow [a..b] \int\limits_a^{b+1} f(x)dx - \sum\limits_{i=a}^{b} f(i) \le f(b+1) - f(a)$

Теорема 1.3.5. *Замена суммы на интеграл*

$$\forall f(x) \nearrow [1..\infty), f > 0, S(n) = \sum_{i=1}^{n} f(i), I_1(n) = \int_1^n, I_2(n) = \int_1^{n+1}, I_1(n) = \Theta(I_2(n)) \Rightarrow \boxed{S(n) = \Theta(I_1(n))}$$

Доказательство. Из лемм 1.3.2 и 1.3.3 имеем $I_1(n) \le S(n) \le I_2(n)$.
$C_1 I_1(n) \le I_2(n) \le C_2 I_1(n) \Rightarrow \min(C_1, 1) I_1(n) \le S(n) \le C_2 I_1(n)$ ■

1.4. Примеры

Вложенные циклы for

```
#define forn(i, n) for (int i = 0; i < n; i++)
int counter = 0, n = 100;
forn(i, n)
  forn(j, i)
    forn(k, j)
      forn(l, k)
        forn(m, l)
          counter++;
cout << counter << endl;
```

Чему равен `counter`? Во-первых, есть точный ответ: $\binom{n}{5} \approx \frac{n^5}{5!}$. Во-вторых, мы можем сходу посчитать число циклов и оценить ответ как $\mathcal{O}(n^5)$, правда константа $\frac{1}{120}$ важна, оценка через $\mathcal{O}$ не даёт полное представление о времени работы.

Число делителей числа

```
vector<int> divisors[n + 1]; // все делители числа
for (int a = 1; a <= n; a++)
  for (int b = a; b <= n; b += a)
    divisors[b].push_back(a);
```

За сколько работает программа?

$$\sum_{a=1}^{n} \lceil \tfrac{n}{a} \rceil = \mathcal{O}(n) + \sum_{a=1}^{n} \tfrac{n}{a} = \mathcal{O}(n) + n \sum_{a=1}^{n} \tfrac{1}{a} \overset{1.3.5}{=} \mathcal{O}(n) + n \cdot \Theta(\int_1^n \tfrac{1}{x} dx) = \Theta(n \log n)$$

1.5. Сравнение асимптотик

Def 1.5.1. *Линейная сложность* $\mathcal{O}(n)$

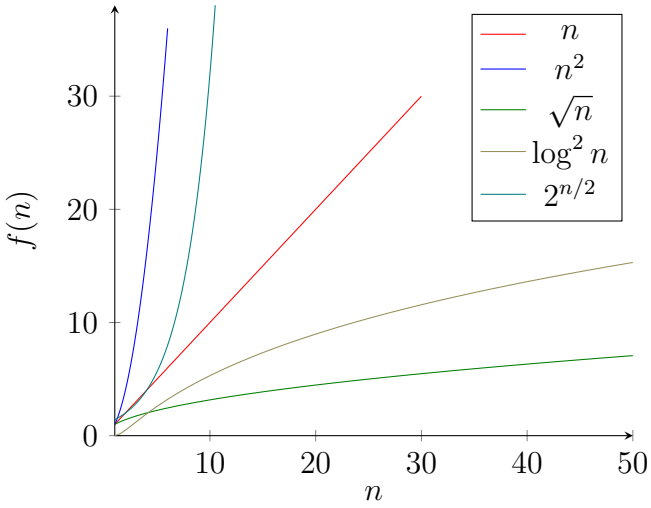
Def 1.5.2. *Квадратичная сложность* $\mathcal{O}(n^2)$

Def 1.5.3. *Полиномиальная сложность* $\exists k > 0 \colon \mathcal{O}(n^k)$

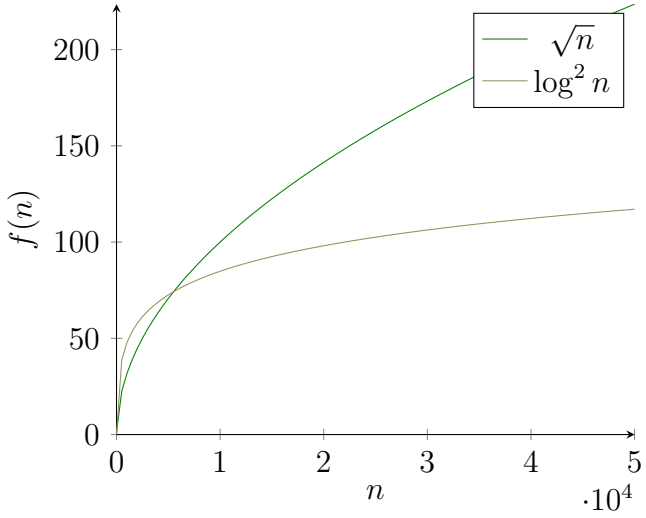
Def 1.5.4. *Полилогарифм* $\exists k > 0 \colon \mathcal{O}(\log^k n)$

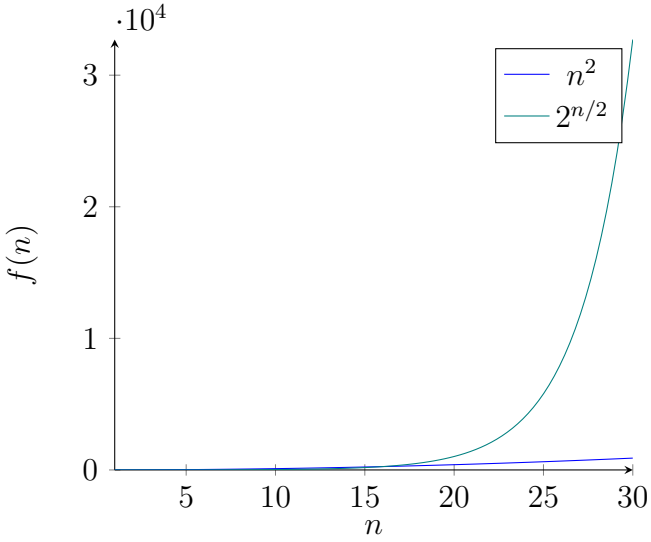
Def 1.5.5. *Экспоненциальная сложность* $\exists c > 0 \colon \mathcal{O}(2^{cn})$

Попробуем сравнить функции:



Заметим, что $2^{n/2}, n^2$ и $\log^2 n, \sqrt{n}$ на бесконечности ведут себя иначе:





Теорема 1.5.6. *Сравнение асимптотик* $\forall a > 0, b > 0, c > 1, \log^b = \mathcal{O}(n^a), n^a = \mathcal{O}(c^n)$

Доказательство. Смотри в следующей лекции... ■

1.6. Реккурентности

Пример: алгоритм Карацубы

Чтобы перемножить два десятичных числа A и B длины n, разделим их на части по $k = \frac{n}{2}$ цифр – A_1, A_2, B_1, B_2. Заметим, что $A \cdot B = (A_1 + 10^k A_2)(B_1 + 10^k B_2) = A_1 B_1 + 10^k (A_1 B_2 + A_2 B_1) + 10^{2k} A_2 B_2$. Если написать рекурсивную функцию умножения (числа длины 1 будем умножать за $\mathcal{O}(1)$), то получим время работы:

$$T_1(n) = 4T_1(\tfrac{n}{2}) + \Theta(n)$$

Из последующей теоремы мы сделаем вывод, что $T_1(n) = \Theta(n^2)$. Алгоритм можно улучшить, заметив, что $A_1 B_2 + A_2 B_1 = (A_1 + A_2)(B_1 + B_2) - A_1 B_1 - A_2 B_2$, где вычитаемые величины уже посчитаны. Итого три умножения вместо четырёх:

$$T_2(n) = 3T_2(\tfrac{n}{2}) + \Theta(n)$$

Из последующей теоремы мы сделаем вывод, что $T_2(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.585..})$.

Lm 1.6.1. *Доказательство по индукции*
Есть простой метод решения реккурентных соотношений: угадать ответ, доказать его по индукции. Рассмотрим на примере $T(n) = 4T(\frac{n}{2}) + \Theta(n)$. Из определения Θ получаем $\exists C$: $T(n) \le 4T(\frac{n}{2}) + Cn$.
Докажем, что $T(n) = \mathcal{O}(n^2)$, для этого достаточно доказать $T(n) \le n^2 + Cn$:
База: $T(1) = 1 \le 1 + Cn$.
Переход: $T(n) \le 4 \cdot 4\big((\frac{n}{2})^2 + C\frac{n}{2}\big) + Cn$

Теорема 1.6.2. *О простом реккурентном соотношении*
Пусть $T(n) = aT(\frac{n}{b}) + n^c \log^d n$. При этом $a > 0, b > 1, c \ge 0, d \in \mathbb{R}$.
Определим глубину рекурсии $k = \log_b n$, обозначим $f(n) = n^c \log^d n$. Тогда верно одно из трёх:

$$\begin{cases} T(n) = a^k = n^{\log_b a} & a > b^c \\ T(n) = f(n) & a < b^c \\ T(n) = k \cdot f(n) & a = b^c \end{cases}$$

Доказательство. Раскроем реккурентность:
$T(n) = f(n) + aT(\frac{n}{b}) = f(n) + af(\frac{n}{b}) + a^2 f\big((\frac{n}{b})^2\big) + \dots = n^c \log^d n + a\frac{n}{b}^c \log^d \frac{n}{b} + a^2 \frac{n}{b^2}^c \log^d \frac{n}{b^2} + \dots$
Все полилогарифмы примерно равны, поговорим об этом позже.
Тогда $T(n) = f(n)(1 + \frac{a}{b^c} + \big(\frac{a}{b^c}\big)^2 + \dots + \big(\frac{a}{b^c}\big)^k)$. При этом в сумме $k+1$ слагаемых.
Обозначим $q = \frac{a}{b^c}$ и оценим сумму $S(q) = 1 + q + \dots + q^k$.
Если $q = 1$, то $S(q) = k + 1 = \log_b n + 1 = \Theta(\log_b n)$
Если $q < 1$, то $S(q) = \frac{1-q^{k+1}}{1-q} = \Theta(1)$
Если $q > 1$, то $S(q) = q^k + \frac{q^k - 1}{q-1} = \Theta(q^k)$ ■

Теорема 1.6.3. *О экспоненциальном реккурентном соотношении*
Пусть $T(n) = \sum b_i T(n - a_i)$. При этом $a_i > 0, b_i > 0, \sum b_i > 1$.
Тогда $T(n) = \Theta(\alpha^n)$, при этом α можно найти бинарным поиском.

Доказательство. Предположим, что $T(n) = \alpha^n$, тогда
$\alpha^n = \sum b_i \alpha^{n-a_i} \Leftrightarrow 1 = \sum b_i \alpha^{-a_i} = f(\alpha)$.
Если $\alpha = 1$, то $f(\alpha) = \sum b_i > 1$, если $\alpha = +\infty$, то $f(\alpha) = 0 < 1$. Кроме того $f(\alpha) \nearrow [1, +\infty)$. Получаем, что на $[1, +\infty)$ есть единственный корень уравнения $1 = f(\alpha)$ и его множно найти бинарным поиском. Теперь можно по индукции доказать, что $T(n) = \mathcal{O}(\alpha^n)$ (оценку сверху) и ??? (оценку снизу). ■