Вопросы к зачоту по структурам

1 Асимптотические оценки времени работы алгоритмо[в] и их смысл

Пусть f, g функции принимающие только положитльные значения

Определение 1 (Тета) $f = \Theta(g) : \exists c_1, c_2 > 0 n_0 : \forall n > n_0 \; c_1 g(n) \leq f(n) \leq c_2 g(n)$

Пример 1.1

$$\frac{n^2}{2} - 3n = \Theta(n^2).$$

Чтоб доказать нужно найти нужные константы

$$c_1 n^2 \leq \frac{n^2}{2} - 3n \leq c_2 n^2.$$

$$c_1 < \frac{1}{2} - \frac{3}{n} \leq c_2.$$

$$c_1 = 1/4.$$

$$c_2 = 1/2.$$

при $n \geq 12$

Пример 1.2

$$c_1 2^n \leq 2^{n+1} \leq c_2 2^n.$$

$$c_1 \leq 2 \leq c_2.$$

Определение 2 (O)

$$f = O(g) : \exists c\, n_0 \forall n > n_0 : f(n) \leq cg(n).$$

Определение 3 (Омега)

$$f = \Omega(n) : \exists c, n_0 \forall n > n_0 f(n) \geq cg(n).$$

Определение 4 (o-малое)

$$f = o(g) : \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0.$$

2 Теорема о рекурсии

Есть рекурсивный алгоритм. Его время работы зависит от параметра n (размера входных данных)

1. Если $n < k, k = const$ решаем задачу нерекурсивно относительно n
2. иначе разбиваем задачу на a подзадач размера $\frac{n}{b}$ каждую решаем рекрсувно

Время работы такого алгоритма описываем такой формулой

$$T(n) = \begin{cases} g(n), n < k \\ a * T(\frac{n}{b}) + f(n), n \geq k \end{cases}.$$

$g(n)$ время для нерекурсивного решения , $f(n)$ время для определения подзадач и собирания результатов рекурсивных вызовов

Теорема 1 *Пусть время работы рекурсивного алгоритма выражается формулой*

$$T(n) = aT(\frac{n}{b}) + f(n).$$

1. *Если* $f(n) = O(n^c), c < \log_b a$ *то* $T(n) = \Theta(n^{\log_b a})$
2. *Если* $f(n) = \Theta(n^c (\log_b a)^k$ *для* $k \geq 0, c = \log_b a,$ *то* $T(n) = \Theta(n^c (\log n)^{k+1})$

3. $f(n) = \Omega(n^c) c > \log_b a$, *то* $T(n) = \Theta(f(n))$ *для это еще должно быть*

$$af(\frac{n}{b}) < kf(n)\ k < 1.$$

Частный случай при $a = b$

$$T(n) = aT(\frac{n}{a}) + f(n).$$

1. *Если* $f(n) = O(n^c), c < 1$ *то* $T(n) = \Theta(n)$
2. *Если* $f(n) = \Theta(n)$ *то* $T(n) = \Theta(n \log n)$
3. *Если* $f(n) = \Omega(n^c), c > 1$, *То* $T(n) = \Theta(f(n))$

Пример 1.1

$$T(n) = 9T(\frac{n}{3}) + n.$$

$$\log_b a = 2.$$

$$n = O(n) \iff T(n) = \Theta(n^2).$$

Пример 1.2

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

$$f(n) = 1.$$

$$a = 1.$$

$$b = \frac{3}{2}.$$

$$\log_{3/2} 1 = 0.$$

3 Сортировка слиянием

Оценка времени работы $T(n) = 2T(\frac{n}{2}) + n$

$$n = \Theta(n).$$

По теореме $T = \Theta(\log n)$

4 Длинная арифметика

Число представляем как список int, каждое число меньше некого числа, которое помешается в int. $a = \{a_0, a_1, \ldots, a_{n-1}\}, \forall i < n - 1\ a_i < b$

$$a = \sum_{i=0}^{n-1} a_i b^i.$$

За b удобно брать большую степень 10 , например миллион. Еще удобнее брать большую степень двух как пример 2^{31}

4.1 Сложение

Сложение двух длинных чисел происходит точно так как в столибик. Для сложения нужно сделать n элементарных операций, где n количесвто цифр самого длинного числа.

4.2 Деление с остатотком

Опять алгоритм деления в столбик

4.3 Умножение

Если умножать в столбик, придется сделать n^2 элементарных операций

4.3.1 Алгоритм карацубы

Пусть есть числа a, b длины n, c основание системы счисления

$$x = c^{n/2}.$$

$$a = \alpha_1 x + \alpha_2.$$

$$b = \beta_1 x + \beta_2.$$

$$ab = \alpha_1 \beta x^2 + \alpha_1 \beta_2 x + \alpha_2 \beta_1 x + \alpha_2 \beta_2 = \alpha_1 \beta_1 x^2 + (\alpha_1 \beta_2 + \alpha_2 \beta_1)x + \alpha_2 \beta_2.$$

$$T(n) = 4T(n/2) + O(n).$$

$$\log_2 4 = 2.$$

$$T(n) = \Theta(n^2).$$

$$\alpha_1\beta_1 x^2 + (\alpha_1\beta_2 + \alpha_2\beta_1)x + \alpha_2\beta_2 = \alpha_1\beta_1 x^2 + ((\alpha_1 + \beta_1)(\alpha_2 + \beta_2) - \alpha_1\beta_1 - \alpha_2\beta_2) + \alpha_2\beta_2.$$

$$T(n) = 3T(\frac{n}{2}) + O(n).$$

5 Алгоритм рабина карпа

Есть строка, есть подстрока, есть хешфункция, все хеши подстрок нужной длины сранвивем с хешем искомой, если равны, сравниваем посимвольно. Для опитимизации, надо сделать хорошую хеш функцию, чтоб было малок коллизий, чтобы она например зависела от позиций.

6 Грамматики,Регулярные выражения

Можно определить язык через регулярные выражения. Рассмотрим регулярки в джаве. В джаве есть класс Pattern и класс Mathcher. Pattern содержит компилированны выражения, Mathcer ищут в тексте штуки по регулярке.

Рассмотрим задачу, есть строка s, нужно проверить подходит ли под регулярку

```
s.mathcher("kjdskjdsk");
```

такая фигня возвращает boolean

7 Задача

Есть две квадратные матрицы, хотим сосчитать произведение. За какое время можем сделать Наивный алгоритм $\Theta(n^3)$

7.1 Алгоритм Штрассена для умножения матриц

Сначала рассмотрим умножение комплексных чисел

$$c_1 = (a + bi).$$

$$c_2 = (c + di).$$

Хотим меньше умножений

$$A_1 = (a + b)(c - d).$$

$$A_2 = ad.$$

$$A_3 = bc.$$

$$(ac - bd) = (ac + bc - ad - bd) + ad - bc = A_1 + A_2 - A_3.$$

$$ad + bc = A_2 + A_3.$$

Мы уменшили количество умножений, за счет увеличений количеств сложений. Пусть есть две вещественно значные матрицы, считаем что размер матрицы есть степень 2, если не степень, дополняем нулями. Каждую матрицу делим на 4 квадрата

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

$$B = \begin{pmatrix} e & g \\ f & h \end{pmatrix}.$$

$$A \times B = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & g \\ f & h \end{pmatrix} = \begin{pmatrix} ae + bf & ag + bh \\ ce + df & cg + dh \end{pmatrix}.$$

Потребовалось 8 матричных умножений и n^2 сложений

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2).$$

по теореме о рекурии $T(n) = \Theta(n^3)$. Херня полная, нужно уменшить рекурсивные вызовы до 7

$$r = ae + bg.$$

$$s = ag + bh.$$

$$t = ce + df.$$

$$u = cg + dh.$$

i	A_i	B_i	$P_i = A_i \times B_i$
1	a	g - h	ag - ah
2	$a + b$	h	$ah + bh$

Бля, лень в випедии посмотрю

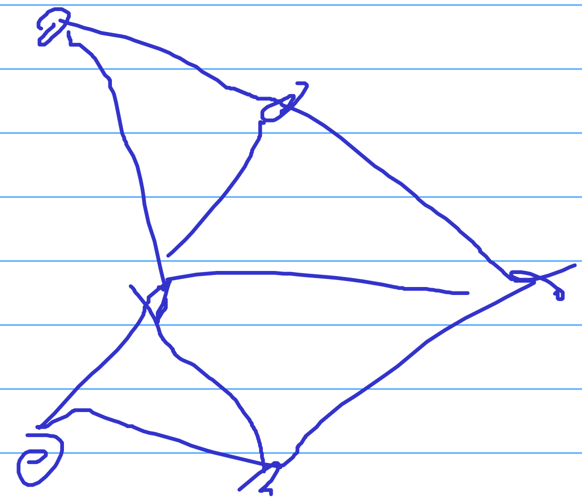
У булевых матриц используют прикол, так как у булевых операций нет вычитания. Булевы значения меняют на $0, 1$ перемножают ка числовые и ненули заменяют на true

8 Алгоритм без полиноминального решения

Такие алгоритмы при больших числа не имеют смысла, поэтому используют приближенный алгоритм

8.1 Задача коммивояжера

Есть связный граф, вершины соеденны ребрами. Есть вершина, надо пройти по всем хотя бы один раз и вернуться в исходную. Так е все ребра имееют вес, A



Задача, решается , если в графе выполняется неравенство треугольника

1. Можно пребрать все гамильтоновы циклы, очень долго и гавно вообще.

2. Жадный алгоритм, дает рещультат не более чем в два раза хуже оптимального.

3. Построение пути по минимальному скелету, не более чем в два раза хуже оптимального

4. Линейное программирование, но может скатиться в полнй перебор

8.2 Жадный алгоритм

Жадный алгоритм на каждом шаге пытается доавить в имеющийся цикл одну вершину, ближайшую к циклу. Этот алгоримт хорошо работает в случае полного графа

9 Слова и алфавиты

$$\text{Алфавит } A = \{a_1, a_2, \dots, a_n\}.$$

$$b \in A.$$

множество символов (букв)

$$\text{Слово } \alpha = a_1 a_2 \dots a_k.$$

Последоватлеьность символов

$$\mid \alpha \mid := k.$$

длина слова

$$\mid \epsilon \mid := 0.$$

пустое слово, не содержит символов

$$A^k.$$

множество слов над алфавитом A длины k Введем операцию котенации (конкатенации)

$$\alpha = a_1 a_2 \dots a_n \in A^n.$$

$$\beta = b_1 b_2 \dots b_m \in A^m.$$

$$\alpha\beta = a_1 a_2 \dots a_n b_1 b_2 \dots b_m \in A^{n+m}.$$

$$\alpha\epsilon = \epsilon\alpha = \alpha.$$

$$A^+ := \bigcup_{i=1}^{\infty} A^i.$$

Множество всех непустых слов над алфавитом A

$$A^* := \bigcup_{i=0}^{\infty} A^i.$$

Множество всех слов над алавитом A

$$\alpha \in A^*, \beta \in B^* \alpha\beta \in (A \cup B)^*.$$

$$L \subset A^* \text{ Язык над алфавитом.}$$

$$L, M \subset A^*.$$

$$LM = \{\alpha\beta \mid \alpha \in L, \beta \in M\}.$$

котенация языков

9.1 Регуляные язык

Рассмотрим алфавит $A = \{a, b, \dots, \}$. Регулярные выражения включают символы языка и операции объединения и котенации

1. Выражение $a \in A$ задает язык $L = \{a\}$

2. E_1 выражение задает язык L_1, E_2 задает язык L_2

$$E_1 | E_2, \text{задает } L_1 \cup L_2.$$

3. E_1E_2 задает L_1L_2

4. $*, +$ переносятся с выражений на языки

9.1.1 Примеры

1. $A = a, b, c, E = a(a|b|c|) * c$ задает язык, определяющй множество слов, которые начинаются на а,кончаются на с

$$A = a|b|c.$$

$$E = aA^*c.$$

2.

$$\lambda = \{\epsilon\}.$$

$$[A] = A|\lambda.$$

$$A^{[k]} = \bigcup_{i=0}^{k} A^i.$$

$$A = \{0, 1, 2, \dots, 9, +, -, E, .\}.$$

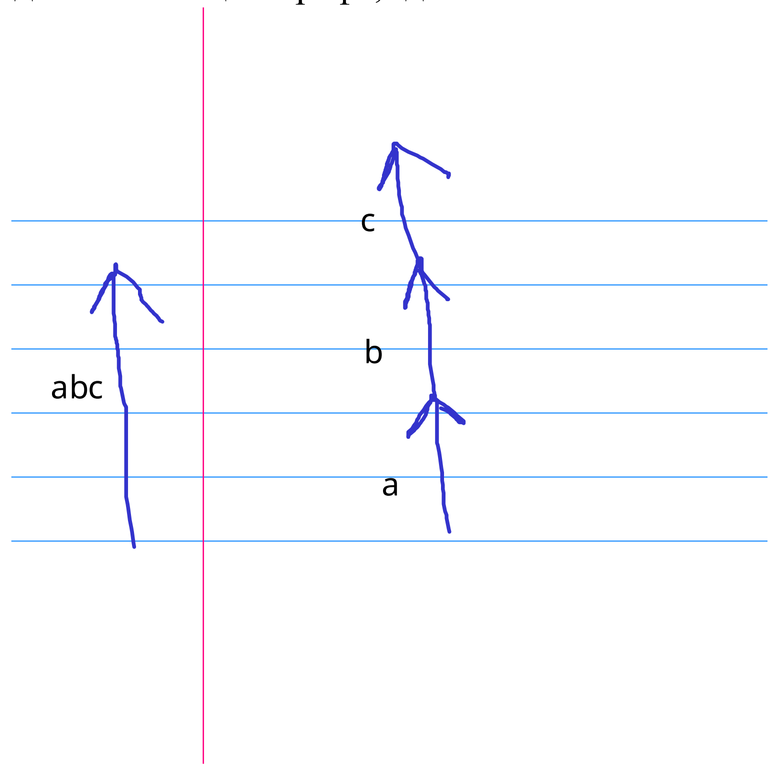
Задаем регулярку для правильных вещественных чисел

$$D = 0|1|\dots|9.$$

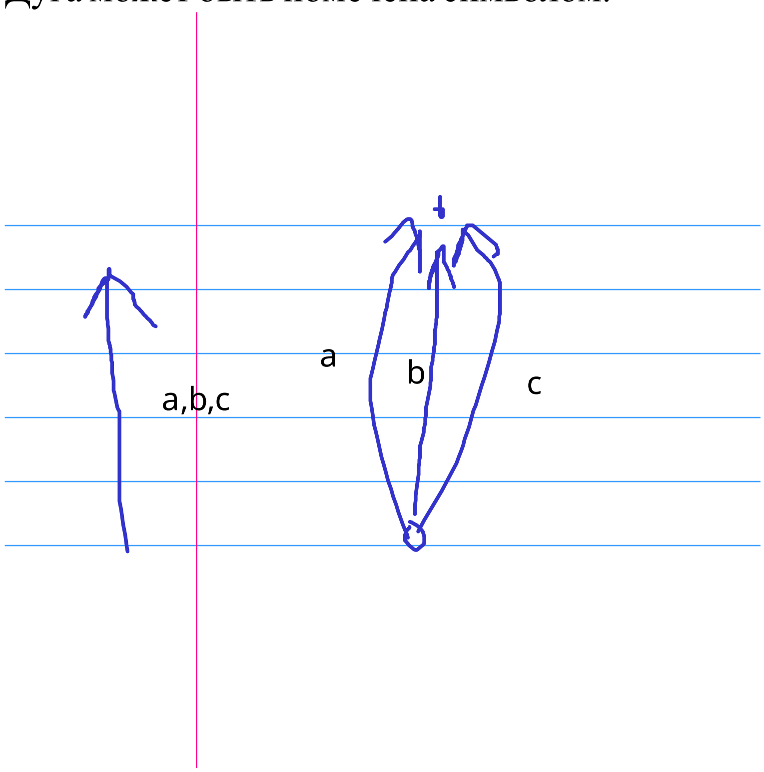
$$[+|-]D^+[.D^+][E[+-]D^+].$$

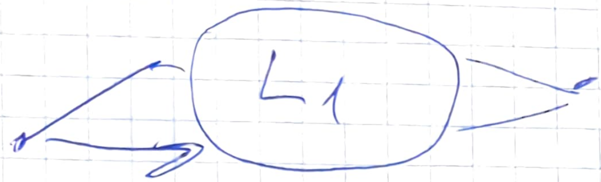
9.2 Графическое задание регулярного выражения

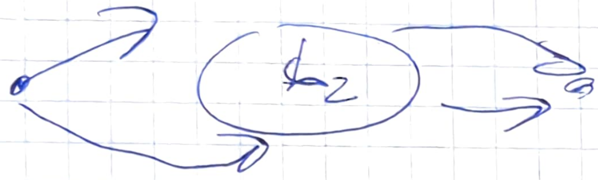
Регулярное выражение можно задать с помощью графа, где есть сток и исток



Дуга может быть помечена символом.

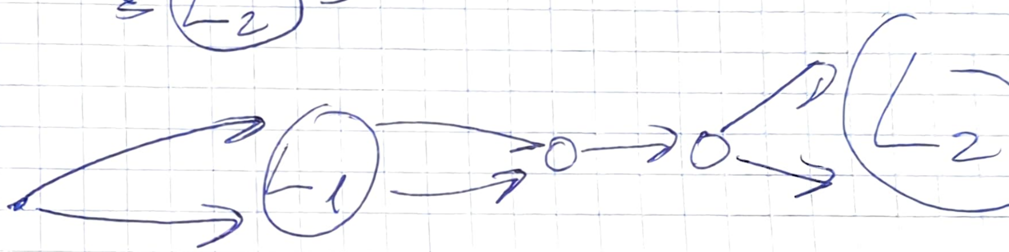




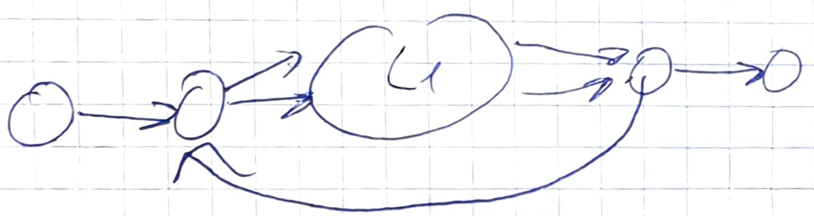


$L_1 | L_1$:

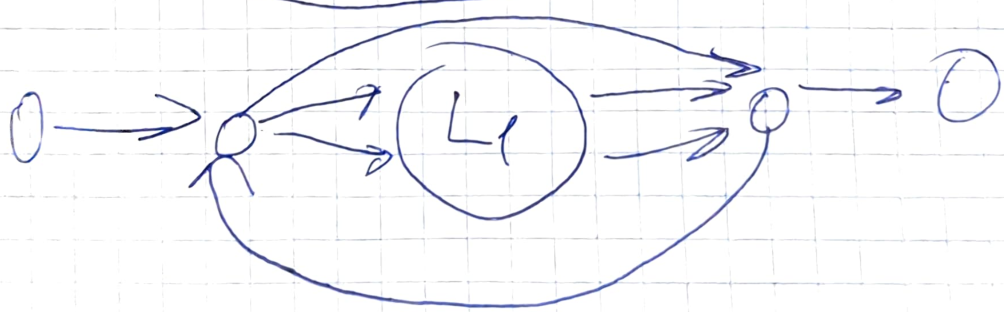
$L_1 L_2$:



L_1^+ :



L_1^* :



9.3 Анализ регулярного выражения

1. Строим граф регулярного выражения
2. В слове рассматриваем последовательно каждый символ
3. Просматриваем все пути в графе, отбрасываем тупиковые

Если хоть один путь привел в сток, слово принадлежит языку

10 Формальные грамматики

Формальная Грамматка G это совокупность следущих объектов

1. Терминальный (основной) алфавит T
2. Нетерминальный (вспомогательный алфавит) N $T \cap N = \emptyset, T \cup N = V$
3. Начальный символ $I \in N$
4. Конечное множество Π правил вида $\alpha \to \beta, \alpha \in V^+, \beta \in V^*$ α содержит хотя бы один нетерминальный символ

10.1 Вывод слов в формальных грамматиках

Вывод слова в грамматике – последовательность слов $\omega_1, \dots, \omega_k$, $\omega_1 = I$ начальное слово ω_k состоит только из терминальных символов

10.2 Пример

1. Терминальны алфавит $\{0, 1\}$
2. Нетерминальный алфавит $\{A\}$
3. $I = A$
4. $\{A \to AA, A \to 01, 1A \to A1\}$

$$A \to AA \to A01 \to AA01 \to 01A01 \to 0A101 \to 001101.$$

10.3 Классы грамматик

1. K_0 грамматики общего вида $\alpha \to \beta, \alpha \in V^+, b \in V^*$
2. K_1 контекстно-зависимые грамматики $\alpha A \beta \to \alpha \gamma \beta, \alpha, \beta \in V^*, A \in N, \gamma \in V^+$
3. K_2 контекстно-свободные грамматики $A \to \gamma, A \in N, \gamma \in V^+$
4. K_3 автоматные(регуляные) грамматики $A \to \gamma B, \vee A \to \gamma, A, B \in N, \gamma \in T^+$

10.4 Дерево вывода

Рассматриваем только контекстно-свобоодные грамматики

$$T = \{i, d, (,), +, -, *\}.$$

$$N = \{E\}.$$

1. $E \to E + E$
2. $E \to E - E$
3. $E \to E * E$
4. $E \to (E)$
5. $E \to i$
6. $E \to d$

Вывод слова $i + d * i + d$

1. E
2. $E + E$
3. $E + E + E$

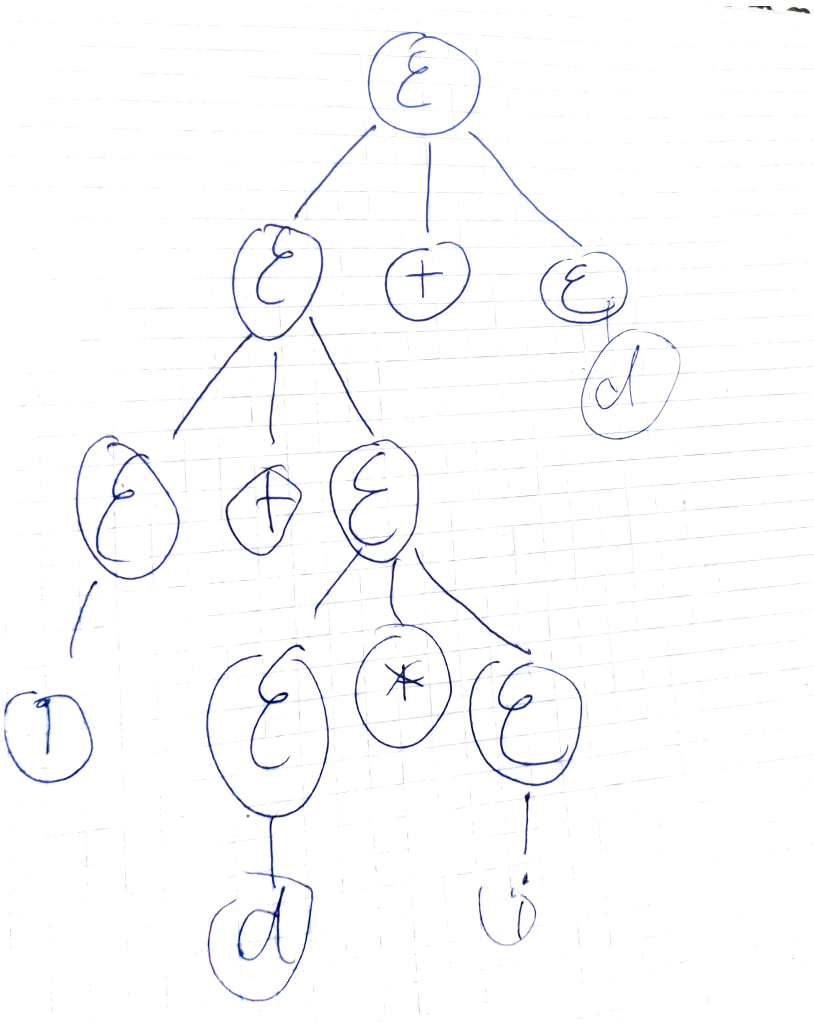
4. $E + E * E + E$

5. $i + E * E + E$

6. $i + d * E + E$

7. $i + d * i + E$

8. $i + d * i + d$



дерево вывода не единственное.

Определим однозначную грамматику для этого языка

$$N = \{E, T, P\}.$$

1. $E \to E + T$

2. $E \to E - T$

3. $E \to T$

4. $T \to T * P$

5. $T \to P$

6. $P \to (E)$

7. $P \to i$

8. $P \to d$

слово $i + d * i + d$

1. E

2. $E + T$

3. $E + T + T$

4. $E + T * P + T$

5. $T + T * P + T$

6. $P + T * P + T$

7. $P + P * P + T$

8. $P + P * P + P$

9. $i + P * P + P$

10. $i + d * P + P$

11. $i + d * i + P$

12. $i + d * i + d$

10.5 Автоматные грамматики

10.5.1 Пример

$$T = \{a, d\}.$$

$$N = \{D, H\}.$$

1. $D \to aH$
2. $D \to a$
3. $H \to aH$
4. $H \to dH$
5. $H \to a$
6. $H \to d$

слово $aadad$

1. D
2. aH
3. aaH
4. $aadH$
5. $aadaH$
6. $aadad$

Дерево автоматной грамматики можно представить в виде графа.

11 Автоматы

Определение 5 (Детерминированный конечный автомат)

$$A =< T, N, I, K, F > .$$

1. *T термнальный альфавит, алфавит анализируемого слова*
2. *N нетерминальный алфавит или алфавит состояний*
3. *$I \in N$ начальное состояние автомата*
4. *$K \subset N$ множество заключительных состояний*
5. *F детерминированная функция переходов $F : T \times N \to N$*

Слово $\alpha = \alpha_0\alpha_1 \dots \alpha_k$ считается допущенным автоматом A, если в последовательности состояний $Q_0Q_1 \dots Q_kQ_{k+1}, Q_0 = I, Q_{i+1} = F(a_1, Q_i)Q_{k+1} \in K$ иначе недопущенно

Определение 6 (Недетерминированный конечнй автомат) *тоже самое, но F : $T \times N \to 2^N$*

Слово $\alpha = \alpha_0\alpha_1 \dots \alpha_k$ считается допущенным автоматом D, если в последовательности состояний $Q_0Q_1 \dots Q_kQ_{k+1}, Q_0 = I, Q_{i+1} \in F(a_1, Q_i)Q_{k+1} \in K$ иначе недопущенно

11.1 Пример

1. $T = \{d, s\}$
2. $N = \{I, H, E\}$
3. I
4. $K = \{H\}$
5. F задаем с помощью таблицы

	s	d
I	H	E
E	E	E

12 Быстрое преобразование Фурье

Пусть задан многолчлен с комплексными коээфициентами степени меньше n

$$A(x) = a_0 + a_1 x + a_2 x^2 + \dots a_{n-1} x^{n-1}.$$

$O(n^2)$ время работы умножения многочленов. Надо быстро получить значения в точках.

$$A(x) = \sum_{i=0}^{n-1} a_i x^i.$$

$$B(x) = \sum_{i=0}^{n-1} b_i x^i.$$

13 Лямбда исчисление

$$\lambda x.e.$$

$$e_1 e_2.$$

$+2$ – функция, которую можно применить к другому числу.

$$\lambda x.(\lambda y. + xy).$$

$$+\ 2\ 6 \to 7.$$

пример δ редукции.

$$(\lambda x.e)a \to e_x^a.$$

13.1 Примеры

1. $(\lambda x.x)2 \to 2$
2. $(\lambda x.x)(\lambda x.x) \to \lambda x.x$
3. $(\lambda x.xx)(\lambda x.xx) \to (\lambda x.xx)(\lambda x.xx)$ Зациклилось все нахуй

У нас два вопроса. Может ли после при применениее редукции в разном порядке, получить новую норальную форму. Нет.

13.2 Порядки редукции

1. Апликативный, когда стараемся вычислить самые внутренние части, ели есть внешний редекс и внутренний редекс (внутри которого нет ничего), стараемся применить к второму.

2. Нормальный порядок редукции, при котором мы применяем редукции к самому левому из самых внешних.

Ленивые вычисления

$$(\lambda x. + xx)(*3\ 4).$$

$$+(*3\ 4)(*3\ 4).$$

$$+12\ 12.$$

$$24.$$

13.3 Как рекурсивную функцию предоставить в лямбда выражение

13.3.1 Факториал

$$if\ e_1\ e_2\ e_3.$$

$$eq_0.$$

$$f = \lambda n.if\ (eq_0 n)\ 1\ (*\ n\ (f\ (-n\ 1))).$$

$$FB = \lambda f.\lambda n.if\ (eq_0 n) 1 (*\ n(\ f(-\ n\ 1))).$$

Найти функцию, которая ищет неподвижную точку, такую функцию назовем комбинато неподвижной точки, Y-комбинатор

$$YF = F(Y\ F).$$

$$Y\ FB = fact.$$

Прилумано огромное количество таких комбинаторов

$$Y = \lambda h.(\lambda x.h(x\ x))(\lambda x.h(xx)).$$

$$YF = (\lambda x.F(x,x))(\lambda x.F(x,x)) = F((\lambda x.F(x,x))(\lambda x.F(xx))) = F(Y\ F).$$

$$Y(FB)2 = FB(YFB)2.$$

$$\lambda n.if(eq_0 n)1(*n((Y\ FB)(n-1)))(2).$$

$$*2(Y\ FB)1.$$

$$*21(YFB)0.$$

$$*2\ 1\ 1.$$

13.4 Чистое лямбда исчисление

13.4.1 Логика

1. TRUE $\lambda x.\lambda y.x$
2. FALSE $\lambda x.\lambda y.y$
3. IF $\lambda p.\lambda x.\lambda y.p\ x\ y$
4. AND $\lambda a.\lambda b.$IF$a\ b$ FALSE
5. OR $\lambda a.\lambda b.$IF $a\ b\ TRUE$
6. NOT $\lambda p.p$FALSE TRUE

13.5 Списки

1. nil пустой список
2. cons получает хрень и список выдает список, где голова хрень, хвост список

3. car – голова

4. cdr - хвост

5. null – поясняет пустой или нет

1. CONS $\lambda a.\lambda b.\lambda p.p\ a\ a$

2. CAR $\lambda a.\lambda b$

14 SKI-исчисление

$$I\ e = e.$$

$$K\ a\ b = a.$$

$$S\ f\ g\ x = f\ x\ (g\ x).$$

$$I = S\ K\ K.$$

$$S\ K\ K\ x = K\ x\ (K\ x) = x.$$

15 Функция абстрагирования

$$abs(x, E).$$

$$[x]E\ \lambda x.E.$$

$$[x]e_1\ e_2 = S[x]e_1[x]e_2.$$