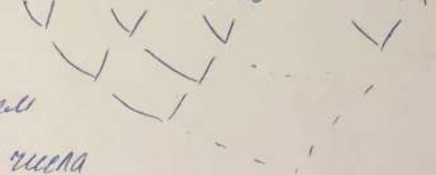
07.12.20. Дискр. матем. д/з от семинара 14.

(1) Посчитать $x_1 \oplus \dots \oplus x_n$ быстрее, чем за $\underline{O}(n \log n)$.

Ответ: можно за $\underline{O}(n)$.

Решение: будем складывать не по очереди $(x_1 \oplus x_2) \oplus x_3 \dots$,
а вот так: $x_1\ x_2\ x_3\ x_4\ x_5\ x_6 \dots x_{n-1}\ x_n$



То есть на 1-м шаге мы складываем $\frac{n}{2}$ раз 2 1-разрядных числа

На 2-м: $\frac{n}{4}$ раз 2 2-разрядных числа

На k-м шаге: $\frac{n}{2^k}$ раз два k-разрядных числа (именно k-разрядных, т.к. в этом числе max сумма $2^k \Rightarrow k$ разрядов)

А сложить два k-разрядных числа мы можем за $5k-12$ – на семинаре научились.

$$\Rightarrow \text{Сложность} = \sum_{k=1}^{\log n} \frac{n}{2^k} \cdot \underbrace{O(k)}_{\le c \cdot k} = n \cdot c \cdot \sum_{k=1}^{\log n} \frac{k}{2^k} \le n \cdot c \cdot \underbrace{\sum_{k=1}^{\infty} \frac{k}{2^k}}_{=2} = 2c \cdot n = \underline{O}(n).$$

Посчитаем $\sum_{k=0}^{\infty} \frac{k}{2^k} \cdot x^{k-1} = S$.

$$\int S dx = \sum_{k=0}^{\infty} \frac{x^k}{2^k} = \sum_{k=0}^{\infty} \left(\frac{x}{2}\right)^k = \frac{1}{1-\frac{x}{2}} = \frac{2}{2-x} + C_0.$$

$$\Rightarrow \left(\frac{2}{2-x}\right)' = 2 \cdot -1\left(\frac{-1}{(2-x)^2}\right) = \frac{2}{(2-x)^2} \Rightarrow \text{при } x=1: \text{ будет } 2. \text{ ч.т.д.}$$

(2) $F = \{ f \in P_2(n) : \|f\| = \frac{2^n}{n} \}$

$\max\limits_{f \in F} L(f) = ?$

Решение: Для начала представим нашу ф-цию в виде

$$f(x) = \bigvee_{\substack{\text{по } \frac{2^n}{n} \text{ наборам,} \\ \text{где } f=1}} x_1^{\sigma_1} \dots x_n^{\sigma_n}$$

Такое представление даёт нам сложность $\underbrace{\frac{2^n}{n}}_{\text{наборов}} \cdot \underbrace{n}_{\text{на вычисление конъюнкции}} = 2^n$

А мы хотим меньше.

Заметим, что мы хотим реализовать $\frac{2^n}{n}$ ф-ций вида $x_1^{\sigma_1} \dots x_n^{\sigma_n}$.
Разобьём это мн-во ф-ций на два: $x_{i_1} \dots x_{i_k}$ и $\overline{x_{j_1} \vee \dots \vee x_{j_l}}$, т.е. отдельно положительные степени, отдельно отрицательные. Их потом ещё нужно склеить, это будет стоить ещё $\frac{2^n}{n}$ операций. Решим, как сделать для 1-го мн-ва. Для 2-го – так же.

Ну рассм. наше мн-во ф-ций как матрицу $\frac{2^n}{n} \times n$.

Если x_i входит в j-ю конъюнкцию, ставим на ij-е место 1, иначе 0.

Разбиваем матрицу на блоки по $\lceil \log_2 n \rceil$ строк. Тогда куски - типа универсальной матрицы. А если их было $k \times 2^k - 1$, то их её реализовали за $2(2^k - k - 1)$.

Т.е. наши куски - за $\sim n$ операций. А всего кусков $\frac{2^n}{n \log n}$

$\Rightarrow$ всё т-е мин-во за $\frac{2^n}{n \cdot \log n} \cdot n = \frac{2^n}{\log n}$ операций.

Ответ: $\underline{O}\left(\frac{2^n}{\log n}\right)$

③ $F = \{f \in P_2(n) : L(f) = 2^{n/2}\}$

$\max L(f) = ?$

$f \in F$.

Решение: Делаем всё то же самое,

получаем: кусок за $\sim n$ операций, всю матрицу за $\frac{2^{n/2}}{\log n} \cdot n$ операций.

Склейка даст ещё $2^{n/2}$ операций. И ещё 2-й кусок: $\frac{2^{n/2}}{\log n} \cdot n$.

Ответ $\underline{O}\left(2^{n/2}\left(1 + \frac{2 \cdot n}{\log n}\right)\right) = \underline{O}\left(2 \cdot 2^{n/2} \cdot \frac{n}{\log n}\right)$

④ $\forall f \in P_2(n)$ $\exists$ схема с усл. остановкой в базисе $\{\&, \vee, \bar{0}, \bar{1}\}$, вычисляющая любую f.

Решение: ~~Разложим~~ Разложим функцию в СДНФ:

$$f = \bigvee x_1^{\sigma_1} \dots x_n^{\sigma_n}$$

причём некоторые $-x_i$, а некоторые $\bar{x}_i$.

переменное

У нас в базисе нет отрицания, надо его как-то сделать, используя усл. остановку.

1 способ Вспомним (см. стр 223),

что если разложить f по 1-й переменной, то будет $f(x_1, x_2, x_3) = x_1 \cdot f_1(x_2; x_3) \vee \bar{x}_1 f_2(x_2; x_3)$,

и программа P для неё такая:

$$\begin{aligned} P_1 &: z = f_1(x_2; x_3) \\ P_2 &: \text{stop}(x_1) \qquad (*) \\ P_3 &: z = f_2(x_2; x_3). \end{aligned}$$

Отсюда верно, как надо сделать для $f(x_1 \dots x_n)$:

$$f = x_1(x_2(\dots) \vee \bar{x}_2(\dots)) \vee \bar{x}_1(x_2(\dots) \vee \bar{x}_2(\dots))$$

И начиная с самых внутренних скобочек, т.е. с переменной x_n и $\bar{x}_n$, применяем конструкцию $(*)$.

2 способ: отсутствие отрицания дает нам такую проблему, что мы можем сделать $x_{i1} \dots x_{ik}$, но не можем $x_{i1} \dots x_{ik}\, \bar{x}_{j1} \dots \bar{x}_{je}$.. то есть не можем различить $11\dots11\dots1$ и $11\dots10\dots0$.

Научимся их различать так:

вот мы знаем ф-цию f и знаем ее значения на всех наборах. Сначала положим $z = f(1\dots1)$.

а потом сделаем $stop(x_1 \dots x_n)$. Если stop выполнится, значит, вообще нулей среди x_i нет, и ответ $f(1\dots1)$ — правильный.

Если stop не выполнился, то значит, хотя бы одно $x_i = 0$.

Тогда берем и последовательно для $n-1$ набора, в каждом из которых $n-1$ единица — делаем то же самое

$z = f(1\dots10)$ — единичного, он последний

$stop(x_1 \dots x_{n-1})$ — x_n — нет.

Если stop выполнится, значит, $x_1 = \dots = x_{n-1} = 1$, и мы могли скленить только значения на $1\dots11$ и $11\dots10$, но этого не произойдет, т.к. набор $1\dots1$ мог отделить раньше.

И так продолжаем дальше: на наборах с $(n-2)$-мя единицами ... на наборах с 0 единиц. ч.т.д.

⑤ $F = \{L$ — линейный оператор $L: \mathbb{B}^n \to \mathbb{R}^n\}$

$\max T(L) = ?$
$L \in F$.

Ответ: $\Theta\left(\frac{n^2}{\log n}\right)$

Решение: по лемме 11.4: $L \le \frac{n(m+\log_2 n)}{\log_2 n}\left(1+\underline{O}\left(\frac{\log\log n}{\log_2 n}\right)\right) \underset{m=n}{=} \frac{n^2}{\log n}$.
(для линейных ф-ций)

$\Rightarrow T \le L \le \frac{n^2}{\log n}$.

~~Дальше, по теореме 11.4:~~ (стр. 198)

Дальше, хотелось бы по теореме 11.4 получить нижнюю оценку для обычной сложности, но она не работает,

т.к. $|F| = 2^{n^2}$

$\log|F| = n^2$ и $n+n = 2n \ne \bar{o}\left(\frac{n^2}{2n}\right)$

$\log\log|F| = 2n$

Докажем сами, что $L \geqslant c \cdot \frac{n^2}{\log n}$.

Допустим, что $L < C \cdot \frac{n^2}{\log n}$ (константу C выберем позже)

Мы знаем, что операторов разных – их 2^{n^2}. ~~(в логарифме отлично $\sim n^2$)~~.

Причем программ должно быть больше, т.к одна программа реализует 1 оператор, но разные программы – могут реализовать один и тот же оператор.

Посчитаем, сколько разных программ, у которых $\leq \frac{C \cdot n^2}{\log n}$ команд.

Команды – 3-х типов

2-местные ф-ции – 16 разных штук

еще n выходов

$\Rightarrow$ (см. ф-ла (2.7) на стр. 233): число программ $= N < 4(L+2n))^{3L}$

↑ т.к еще выход не один, а их n.

Проверим, что $2^{n^2} < 4(L+2n))^{3L}$

Логарифмируем: $n^2 < \log(4(L+2n))^{3L}) \sim 3L\log(L+2n) \sim 3L\log L \lesssim 3C \cdot \frac{n^2}{\log n} \cdot 2\log n = 6C \cdot n^2$?

$L \leq C \cdot \frac{n^2}{\log n}$

$\Rightarrow$ если $C := \frac{1}{6}$, то число программ окажется меньше числа операторов, а это противоречие.

$\Rightarrow \exists$ программа: $L \geq \frac{1}{6} \cdot \frac{n^2}{\log n}$.

$\Rightarrow L(\max_{f \in F} L(f)) = \theta\left(\frac{n^2}{\log n}\right)$

А поскольку (см. стр. 235): $2 \lesssim \frac{L(f)}{T(f)} \lesssim 8$, т.е L и T отличаются в константу раз, то и $T(\max_{f \in F} L(f)) = \theta\left(\frac{n^2}{\log n}\right)$.

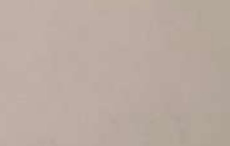
<u>Ответ</u>: $T = \theta\left(\frac{n^2}{\log n}\right)$

(6.) $\forall P$: $H(P_n) \leq C(P_n) \leq H(P_n) + 1$.

Док-ть, что $\forall n$ и $\forall \varepsilon > 0$: $\begin{cases} \exists P_n : C(P_n) \geq H(P_n) + 1 - \varepsilon \\ \exists \hat{P}_n : C(\hat{P}_n) \leq H(P_n) + \varepsilon \end{cases}$, т.е асимпт. н-во неулучшаемо.

<u>Решение</u>: а) найдем такое $P_n = \{p_1 \dots p_n\}$, $p_i > 0$, что $C(P_n) = H(P_n)$ – т.е без всякого ε даже.

Ну берем $P = \{p_1 \dots p_{n-1}; p_n\}$, где $p_i = \begin{cases} \frac{1}{2^i}; & i = 1 \dots n-2 \\ \frac{1}{2^{n-1}}; & i = n-1, n. \end{cases}$



Его оптимальным кодом (см. алгоритм Хаффмана) будет ~~код~~ полный префиксный код с $l_i = \begin{cases} i; & i = 1 \dots n-2 \\ n-1; & i = n-1, n \end{cases}$

Тогда $C(P)=\sum_{i=1}^{n} l_i p_i$ ← т.к. $-\log(p_i) = -\log\left(\frac{1}{2^i}\right) = i$, $\forall i = 1 \dots n-2$,

$\parallel$

$H(P) = -\sum_{i=1}^{n} p_i \log p_i$ ← а для 2-х последних: $l_{n-1}p_{n-1} + l_n p_n = 2\cdot(n-1)\cdot\frac{1}{2^{n-1}} =$

$= -p_{n-1}\log p_{n-1} - p_n \log p_n = -2\cdot\frac{1}{2^{n-1}}\cdot(-(n-1))$.

И вообще, можно взять любое бинарное насыщенное дерево,

и полагаем $p_i := \frac{1}{2^{l_i}}$

$\Rightarrow C(P) = \sum_{i=1}^{n} l_i p_i = \sum_{i=1}^{n} \frac{l_i}{2^{l_i}}$

$H(P) = -\sum_{i=1}^{n} p_i \log p_i = -\sum_{i=1}^{n} \frac{1}{2^{l_i}}\cdot(-l_i) = \sum_{i=1}^{n}\frac{l_i}{2^{l_i}}$ они равны!

б) покажем, что $\forall n$ и $\forall \varepsilon > 0$ $\exists P = \{p_1 \dots p_n\}, p_i > 0 : C(P) \geq H(P) + 1 - \varepsilon$.

Возьмём такое распр.: $\left(\frac{1}{2^m}; \frac{1}{2^m} \dots 1 - \frac{n-1}{2^m}\right)$

(т.е. при $m \to \infty$: одна вер-ть $\to 1$, а остальные – к нулю).

тогда энтропия $\to 0$.

А $C(P)$ – оно всегда ≥ 1, т.к. длина ≥ 1.

$\Rightarrow C(P)$ и $H(P)$ становятся сколь угодно близки при $m \to +\infty$.

Ну можно честно посчитать:

$C_m(P) = \left(1 - \frac{n-1}{2^m}\right)\cdot 1 + \frac{1}{2^m}\sum_{k=2}^{n-1}$ числа

↑ по теореме 13.4 стр. 244 (ну или алгоритм Хаффмана) — и число С

$H_m(P) = -\frac{n-1}{2^m}\cdot\log\left(\frac{1}{2^m}\right) - \left(1 - \frac{n-1}{2^m}\right)\log_2\left(1 - \frac{n-1}{2^m}\right) = \frac{n-1}{2^m}\cdot m - \left(1 - \frac{n-1}{2^m}\right)\cdot\log_2\left(1 - \frac{n-1}{2^m}\right)$

$\Rightarrow$ при $m \to +\infty$: $C_m(P) \to 1$.

$H_m(P) \to 0 + 0 = 0$

Причём по теореме 13.5: $C \leq H + 1$.

$\Rightarrow \begin{cases} C_m - H_m - 1 \leq 0 \\ C_m - H_m - 1 \to 0 \end{cases} \Rightarrow \forall \varepsilon > 0\ \exists m: C_m - H_m - 1 \geq -\varepsilon$

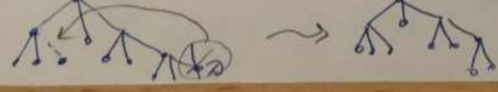
$\Rightarrow C_m \geq H_m + 1 - \varepsilon$. Чтд.

7. q-ичный алгоритм Хаффмана.

У нас в каждой вершине не 2 потомка, а q.

Ясно, что оптим. корм можно искать среди префиксных (т.к. если код оптимальный, то он удовл. аналогу н-ва Крафта-Макмиллана, которое доказывается так же, как для q=2, и раз он ему удовл, то можно сделать код с такими же длинами кодовых слов, но уже префиксный).

Дальше, если в дереве есть ненасыщенные вершины, то делаем так:



Длина кодовых слов не увеличится ⇒ стоимость кода не возрастет.
⇒ все вершины, кроме одной - нагруженные
Дальше мы сейчас докажем аналог теоремы 13.4 (стр. 244), и получим

Алгоритм: 1) ищем остаток r_0 от деления $r-1$ на $q-1$.

- Если он = 0, то просто собираем по q самых маленьких вер-ней, объединяем в вершину и т.д. ... собираем, пока единицу не получим
- Если $r_0 \neq 0$, то на 1 шаге объединяем $q' = r_0 + 1$ самых маленьких вероятностей, а дальше всегда по q вероятностей.

2) построили дерево. В листьях - записаны a_i. И код a_i - это на пути от корня до a_i, и записываем все встретившиеся символы. Все.

Почему это оптимальный код? Докажем теорему

Теорема Кодируем: $A = \{a_1 \dots a_r\} \to B = \{b_1 \dots b_q\}$; $\Pi = (p_1 \dots p_r)$; $p_1 \geq \dots \geq p_r$.
r_0 = остаток от деления $r-1$ на $q-1$.
Если $r_0 \neq 0$, то $q' := r_0 + 1$.
Если $r_0 = 0$, то $q' := q$.
} собираем самые маленькие вер-ли в одну вершину.
Пусть $\Pi' := (p_1' \dots p_{r-q'+1}')$, где $p_i' = p_i$; $i = 1 \dots r-q'$
$A' := \{a_1 \dots a_{r-q'+1}\}$ $\quad p'_{r-q'+1} = \sum_{i=r-q'+1}^{r} p_i$
Пусть Σ' - схема кодирования из A' в B; $\Sigma' : \{h'(a_i) = B_i'; i = 1 \dots r-q'+1\}$.
Определим схему Σ из A в B: $\{h(a_i) = B_i; i = 1 \dots r\}$: $B_i = B_i'$; $i = 1 \dots r-q'$; $B_{r-q'+j} = B'_{r-q'+1} B_j$; $j = 1 \dots q'$.
Док-ть, что Σ - оптим. схема из A в B с $\Pi \Leftrightarrow \Sigma'$ - оптим из A' в B с Π'.

Док-во: пусть S - стоимость схемы Σ; S' - стоимость схемы Σ'!
Из построения: $S - S' = p'_{r-q'+1}$.

1) Если Σ - не оптим, то и Σ' - не оптим. от противного.
пусть $\exists\, \Sigma_0 : A \to B$ с Π, причем $S_0 < S$.
можно сделать
Из доказанного, что у оптим. схемы самые маленькие вершины собираются в вершину, т.е. q' вершин с вер-ями p_j; $j = r-q'+1 \dots r$ - имеют общую вершину. Удалим у дерева эти q' вершин вместе с ребрами и ~~стрелки~~ объявим коровой вершиной их предка с вер-ю $p'_{r-q'+1}$, тогда получим дерево для $A' \to B$ с Π' и стоимостью $S_0' = S_0 - p'_{r-q'+1} < S'$ $\Rightarrow$ Σ' - не оптим.

2) Если Σ' - не оптим, то и Σ - не оптим.
от противного. пусть $\exists$ схема $\Sigma_0' : A \to B$ с Π', причем $S_0' < S'$. Рассм. соотв. схеме Σ_0' коровое дерево, и в этом дереве к вершине с вер-ю $p'_{r-q'+1}$ пристроим q' новых вершин на ребрах, и припишем им вер-ли p_j; $j = r-q'+1 \dots r$, а саму эту вершину из числа коровых исключим. В результате получим дерево из A в B с Π, причем его стоимость $S_0 = S_0' + p'_{r-q'+1} < S$.
$\Rightarrow$ нашлась схема со стоимостью < чем Σ.
$\Rightarrow$ Σ - не оптим.

$\Rightarrow$ Σ и Σ' - либо обе оптимальные, либо неоптимальные.