⑦ $f(n)$ - зависит от n перем.

Б - базис из всех не более чем 4-местных ф-ций.

$L_Б(f) \le ?$

<u>Решение:</u> У нас $n = 22+20 = 42$.

Будем действовать, как в теореме Лупанова,

а параметры k и l: в конце перебор по $k \in [1; 42]$, $m \in [1; 2^{n-k}]$ сделаем и выберем те, которые минимум доставят.

Сначала вспомним, за сколько операций мы можем вычислить все конъюнкции от $x_1 \dots x_k$ (см лемма 11.1 на стр. 191)

Делим все переменные пополам, т.е вычисляем все конъюнкции от $x_1 \dots x_{\frac{k}{2}}$, за $\le \underbrace{\frac{k}{2}}_{\text{операций на одну конъюнкцию}} \cdot \underbrace{2^{k/2}}_{\text{таких конъюнкций}}$, потом столько же на все конъюнкции от $x_{\frac{k}{2}+1} \dots x_k$. И потом еще нужно перемножить все куски - это $2^{\frac{k}{2}} \cdot 2^{\frac{k}{2}} = 2^k$ опер.

$\Rightarrow$ всего $2 \cdot \frac{k}{2} \cdot 2^{k/2} + 2^k = k \cdot 2^{k/2} + 2^k$.

Теперь по методу Лупанова сначала разложим ф-цию f по первым k переменным, а потом каждую из полученных ф-ций разобьем на $2^{\frac{n-k}{m}}$ кусочков ($\underbrace{\boxed{\quad\quad|\quad}}_{\frac{2^{n-k}}{m}}$), где m - выберем перебором потом.

$$\Rightarrow f(x_1 \dots x_n) = \bigvee_{\sigma_1 \dots \sigma_k} f(\sigma_1 \dots \sigma_k, x_{k+1} \dots x_n)\, x_1^{\sigma_1} \dots x_k^{\sigma_k} = \bigvee_{\sigma_1 \dots \sigma_k} f_{\sigma_1 \dots \sigma_k}(x_{k+1} \dots x_n)\, x_1^{\sigma_1} \dots x_k^{\sigma_k} =$$

$$= \bigvee_{\sigma_1 \dots \sigma_k} x_1^{\sigma_1} \dots x_k^{\sigma_k} \cdot \left(\bigvee_{i=1}^{\frac{2^{n-k}}{m}} f_{\sigma, i}(x_{k+1} \dots x_n) \right)$$

Схема S_1 - вычисляет все элем. конъюнкции вида $x_1^{\sigma_1} \dots x_k^{\sigma_k}$.

На это надо $\le k \cdot 2^{k/2} + 2^k$ операций

Схема S_2 - вычисляет все элем. конъюнкции от $x_{k+1} \dots x_n$.

На это надо $\le (n-k) \cdot 2^{\frac{n-k}{2}} + 2^{n-k}$ операций

Схема S_3 собирает вертикальную полосу дизъюнкцией из m элем. конъюнкций степени $n-k$ (которые вычислила S_2)

На это надо $\underbrace{2^m \cdot m}_{\text{слагаемое одной полоски}} \cdot \underbrace{\frac{2^{n-k+1}}{m}}_{\text{кол-во полосок}} = 2^{n-k+m+1}$

Схема S_4 собирает слагаемые - строчки таблицы, из $\le \frac{2^{n-k}}{m}$ кусков

На это надо $\le \underbrace{2^k}_{\text{строк}} \cdot \underbrace{\left\lceil \frac{2^{n-k}}{m} \right\rceil}_{\text{для одной строки}} \le \frac{2^n}{m} + 2^k$ ← за счет целой части

Схема S_5 умножает ф-ции из схемы S_3 на элем. конъюнкции из схемы S_1.

Это 2^k конъюнкций и 2^k дизъюнкций $\Rightarrow L(S_5) \le 2 \cdot 2^k$.

$$\Rightarrow L(S) \leq k\cdot 2^{k/2} + 2^k + (n-k)\cdot 2^{\frac{n-k}{2}} + 2^{n-k} + 2^{n-k+m+1} + \frac{2^n}{m} + 2^k + 2\cdot 2^k =$$

$$= 4\cdot 2^k + 2^{n-k} + k\cdot 2^{k/2} + (n-k)\cdot 2^{\frac{n-k}{2}} + \frac{2^n}{m} + 2^{n-k+m+1}.$$

У нас $n=42$

А $k \in [1;42]$, $m \in [1; 2^{n-k}]$ – нужно перебрать и выбрать, при каких значениях параметров $L(S)$ получится меньше всего.

пишем программу на Python: (или на си, потому что в питоне 2^{n-k} не влезло во float)

```
n=42
k0=1
m0=1
otv= 4*2**k0+2**(n-k0)+k0*2**(k0/2)+(n-k0)*2**((n-k0)/2)+(2**n)/m0 +2**(n-k0+m0+1)
for k in range(1,n+1):
    for m in range(1, 2**(n-k)):
        otv_tek = 4*2**k+2**(n-k)+k*2**(k/2) +(n-k)*2**((n-k)/2)+(2**n)/m +
            + 2**(n-k+m+1)
        if(otv_tek < otv):
            print(k,m,otv_tek)
            otv=otv_tek.
```

программа выдала $k=32$

$m=23$

$otv = 225\,581\,250\,390.26086.$

$otv \approx \frac{1}{4}\cdot 10^{12}$

Зам. А $\frac{2^n}{n} = \frac{2^{42}}{42} \approx \frac{4\cdot(10^3)^4}{42} = \frac{4\cdot 10^{12}}{42} \approx 10^{11}$. Ну почти одинаково

$2^{10}\approx 10^3$

Зам. Если $n=20$, то $k=13$

$m=7$

$otv = 216\,716.39307195874$

$otv \approx 2\cdot 10^5$

А $\frac{2^n}{n} = \frac{2^{10}}{20} \approx \frac{10^3}{20} = \frac{1}{2}\cdot 10^2$ } очень большая разница.

Значит, $n=42$ – это уже достаточно большое n, а $n=20$ – ещё не достаточно большое

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>


int main(){

int n=20;
int m0=1;
int k0=1;
double otv=4*pow(2,k0)+pow(2,n-k0) + k0*pow(2,k0/2)+ (n-k0)*pow(2,(n-k0)/2)+pow(2,n)/m0
+pow(2,n-k0+m0+1);
printf("n=%d otv=%le\n", n, otv);
double otv_tek;
for (int k=1; k<=n; k=k+1){
    for (int m=1; m<=pow(2,n-k); m=m+1){
        //printf("k=%d,m=%d\n",k,m);
        otv_tek=4*pow(2,k)+pow(2,n-k) + k*pow(2,k/2)+ (n-k)*pow(2,(n-k)/2)+pow(2,n)/m
+pow(2,n-k+m+1);
        if (otv_tek<otv){otv=otv_tek; printf("%d %d %le\n", k,m,otv);}
    }

}
printf("otv=%le\n",otv);

}
```

```
3 1 1710131.142704822
4 1 1380432.0
4 2 1118288.0
5 2 822071.5743110038
6 2 673840.0
6 3 630149.3333333333
7 2 599783.8216433873
7 3 490557.15497672063
```

lhost

c 2c... | Я конверта... | integer -... | Docume... | Untitled...

test_v_sphery — -bash — 80×24

```
n=20 otv=3.679753e+06
2 1 2.368532e+06
3 1 1.708326e+06
4 1 1.380432e+06
4 2 1.118288e+06
5 2 8.212680e+05
6 2 6.738400e+05
6 3 6.301493e+05
7 2 5.994160e+05
7 3 4.901893e+05
8 3 4.210773e+05
8 4 3.992320e+05
9 3 3.868853e+05
9 4 3.322720e+05
10 4 3.006720e+05
10 5 2.810112e+05
11 5 2.516832e+05
11 6 2.494987e+05
12 5 2.436352e+05
12 6 2.250667e+05
13 6 2.249307e+05
13 7 2.163486e+05
otv=2.163486e+05
(base) MacBook-Pro-Aleksandra:test_v_sphery aleksandra$
```

```
k0)/2)+pow(2,n)/m0

-k)/2)+pow(2,n)/m
;}
```

48 % Пн 20:28