Sestrojte hradlovou síť pro porovnání 2 n-bitových čísel v hloubce $O(\log n)$, která vrátí 1 pokud $x < y$, jinak vrátí 0.

Mějme x: 0 0 1 0 0
y: 0 1 0 0 0

Pozorování 1: Mějme binární číslo se samými nulami kromě pozice i. Pak je vždy větší než číslo ≤ 1 na všech pozicích $> i$:

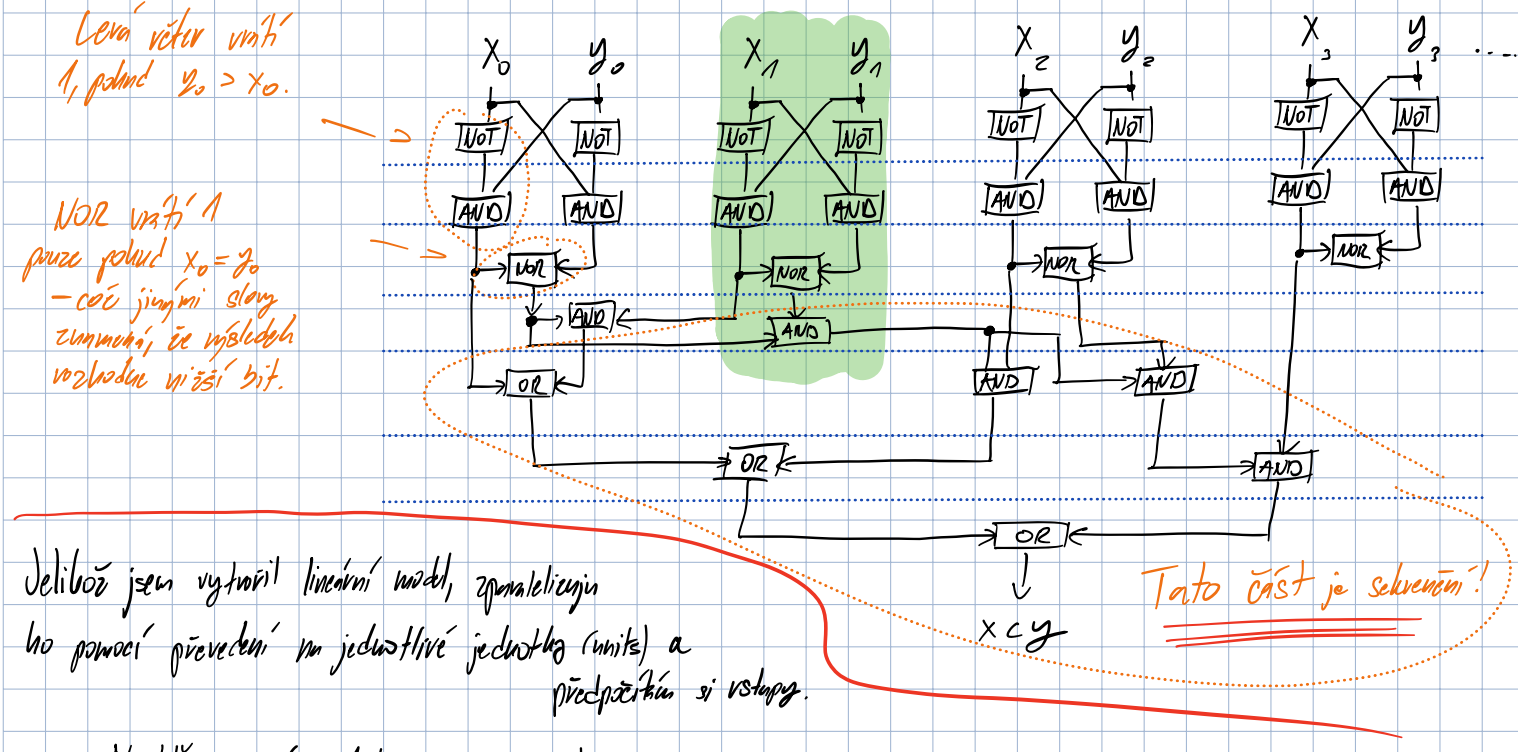
x y
0 0 0 1 1 ... < 0 0 1 0 0 ...

Pozorování 2:

Pokud porovnáním x, y a y má jako první (při průchodu zleva) 1, zatímco x stále nemá, je y určitě větší.

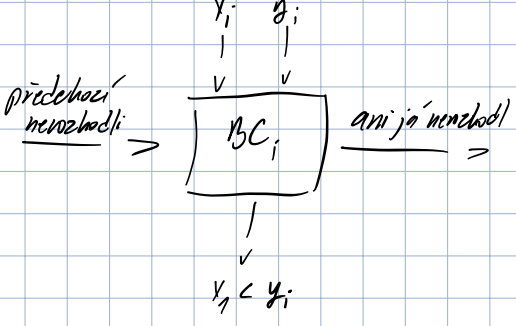
Hledáme $x < y$:

$x_0 \rightarrow MSb$, $x_{n-1} \rightarrow LSb$!

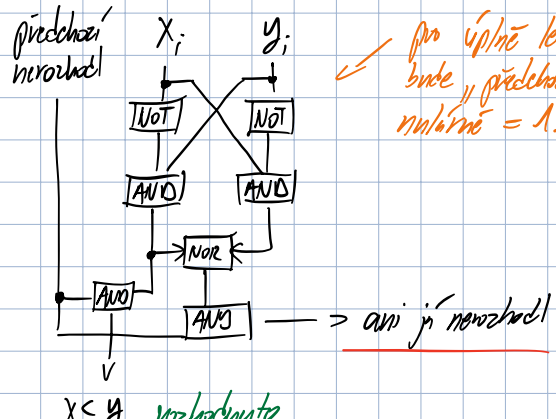


Jelikož jsem vytvořil lineární model, zparalelizuju ho pomocí převedení na jednotlivé jednotky (units) a předpočítám si vstupy.

Nechť porovnání 1-bitu je nový box (BitComparer)



≃



pro úplně levý bit bude „předchozí nerozhodl" natvrdo = 1.

Tabulka pro 1-bit bloky:

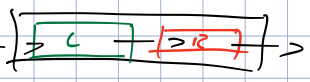
y \ x	0	1
0	<	0
1	0	<

Kde: 0 = bylo rozhodnuto ($x<y$ or $x>y$), tedy se nic napříště dál

< = nebylo nijak rozhodnuto (kopíruju přenos)

1 nikdy negeneruji, protože buď kopíruji (rozhodl jsem) nebo pohlcuji (rozhodl jsem a další bity už nerozhodnou).

Tabulka pro chování komunikačního bloku:



R \ L	0	<
0	0	0
<	0	<

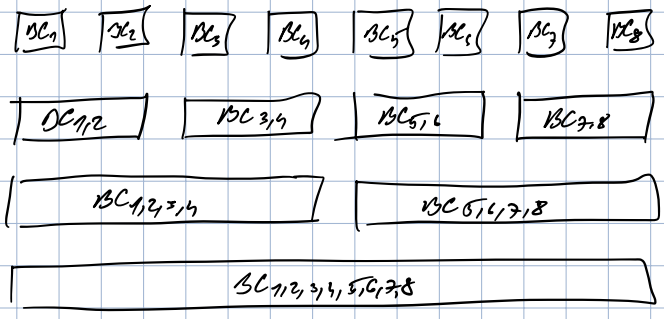
pokud bylo v levém bloku rozhodnuto, určitě se dál nic rozhodovat nebude, tudíž 0.

pokud vlevo se nerozhodlo, ale vpravo ano, celkově se bere v úvahu rozhodnutí.

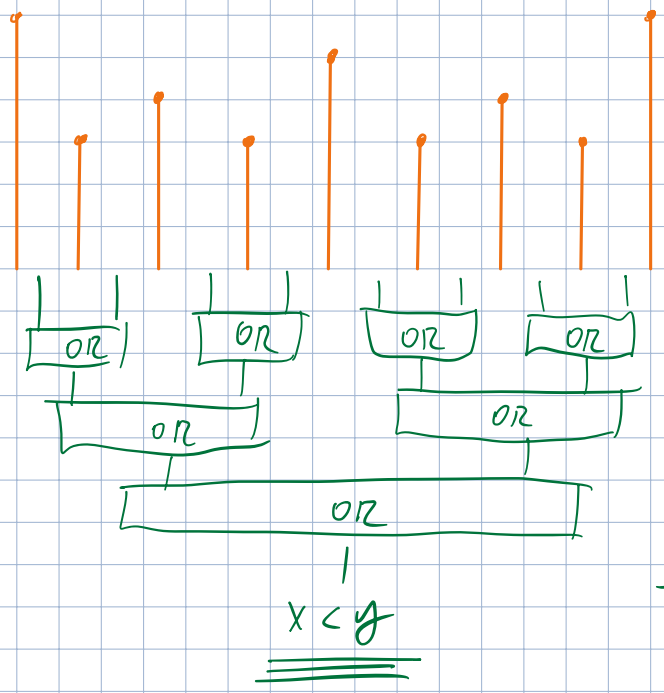
pokud ani L, R nerozhodlo, celý blok nerozhodl.

Tedy vytvořím obdobnou „unit" jako u binární sčítačky a opět budu počítat s obvyklými případy vstupu „předchozí nerozhodli", tedy si to předpočítám, stejně jako jsme to dělali na přednášce.

Tedy si vytvořím síť komunikačních bloků (units) a vypočítám chování jednotlivých bloků.



Zde bude hloubka očividně $O(\log n)$



Zde pro známe „rozhodnutí" umíme vypočítat „přenosy" mezi jednotlivými units.

Díky rozdělení výše bude hloubka opět $O(\log n)$, protože postupně můžu vyhodnotit paralelně 2, 4, 8... units.

V posledním kroku musím vzít výsledky z jednotlivých units a postupně udělat OR mezi všemi, což se dá taktéž paralelizovat mezi dvojice, čímž opět získáme $O(\log n)$ hloubku.

Celkově je tedy hloubka $3 \cdot O(\log n) \leq O(\log n)$

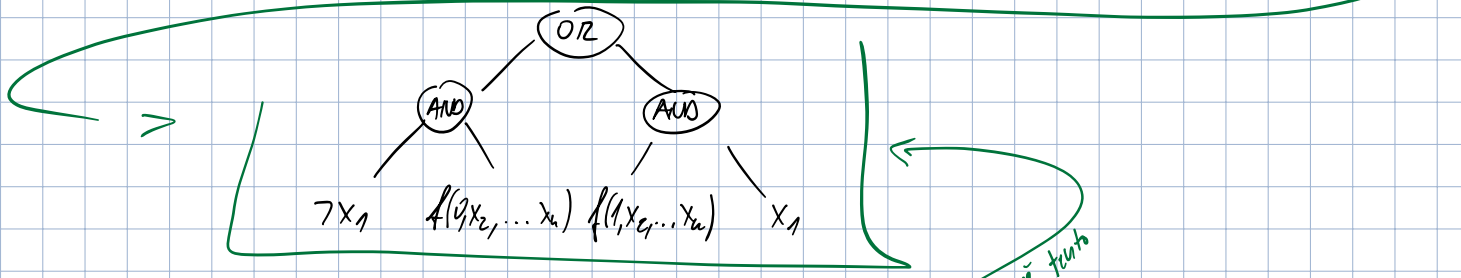
Jednotlivé units můžeme považovat za konstantní hloubky ($k=4$).

Ve výsledku jsem tak linearitu ze začátku díky předpočítání výsledků zredukoval na $O(\log n)$.

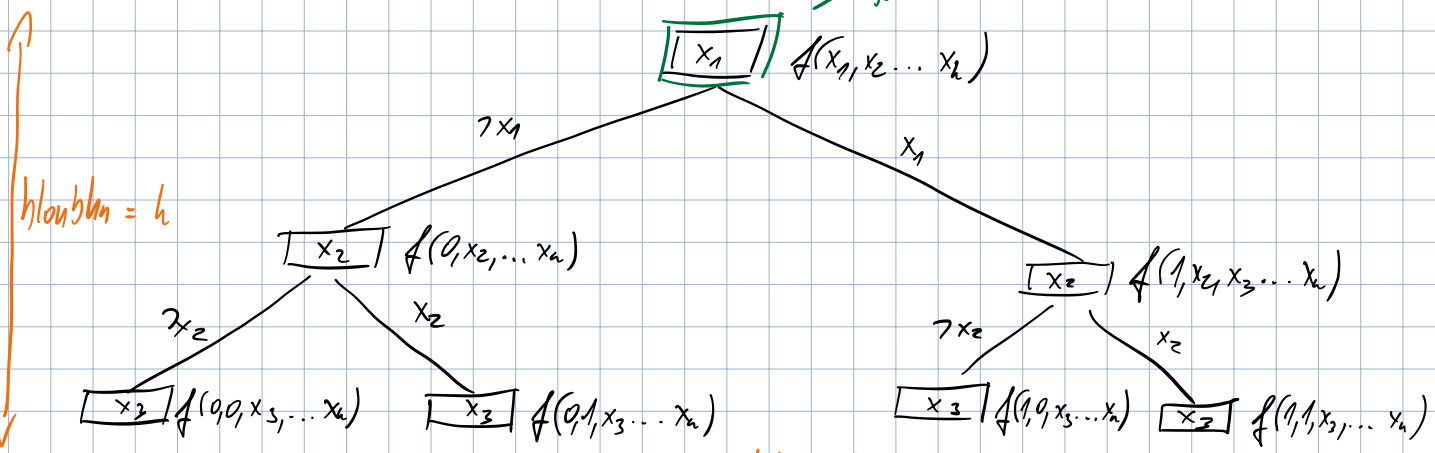
Dokažte, že lib. booleanovskou funkci s k vstupy lze spočítat booleanovským obvodem hloubky $O(k)$ s $O(2^k)$ hradly.

Pozn.: Booleanovské funkce se skládají pouze z AND, OR, NOT gatů.
– Každý z těchto gatů má výstupní aritu právě 1, vstupní právě 1.

Mějme nyní booleanovskou funkci $f: \{0,1\}^k \to \{0,1\}$ reprezentovanou jako $f(x_1, \dots, x_k)$.
Pak můžeme takovou funkci vyjádřit jako $(x_1 \wedge f(1, x_2, \dots, x_k)) \vee (\neg x_1 \wedge f(0, x_2, \dots x_k))$.
Takové formule rozhodně musí platit, jelikož defacto jen propaguje proměnnou mimo výpočet a její hodnotu ve výpočtu fixuje. Závorce takové formule jde reprezentovat takto:



Avšak takto bychom mohli rozepsat rozklad dále:



Mimo jiné vzniklne úplný binární strom

tedy postupně fixovat proměnné a větvit se podle jejich možných (0/1) ohodnocení.

Až dojdu do listů, kde budou všechny proměnné funkce zafixovány, tudíž takové funkce může nahradit nulárním hradlem odpovídající výsledku pravdivostní tabulky pro zadanou posloupnost proměnných $x_1 - x_k$.

Očividně je hloubka $= k$, tedy počtu proměnných, jelikož v každé vrstvě zafixuji právě jednu další proměnnou. Celkem jich mohu zafixovat k.

Tedy hloubka je $O(k)$

Pro počet hradel musí platit následující: $S(2) = 1$ → tedy pro funkci $f(x_1, x_2)$ s dvěma fixovanými proměnnými stačí 1 nulární hradlo s hodnotou z tabulky.

$$S(n) = 2 \cdot S(n-1) + 6$$

Odůvodnění:

2 - kvůli: V každé vrstvě se rozdělím na dvě strany, kde danou proměnnou zafixuji a zbývá mi již $n-1$ nezafixovaných proměnných

$S(n-1)$: V každé vrstvě potřebuji znát hodnotu funkce, kterou mám již o jednu méně nezafixovaných proměnných, jelikož jsem jednu na vrstvě aktuálně zafixoval.

+6: V každém boxu $\boxed{x_i}$ → popsáno výše — použiji: 1x OR, 2x AND, 1x NOT (pro negaci x_i), nulární 1, nulární 0. (pro nastavení hodnot)

Dostáváme tedy rekurenci, kdy

$$S(n) = \frac{7}{4} 2^n - 6 = O(2^n)$$

Tedy celkem hradel bylo použito $O(2^n)$

Jelikož na rekurenci není možné přímo použít Master Theorem, dovolil jsem si na výsledek rekurence použít WolframAlpha.

X

Celkově je takový postup platný, jelikož je:

- Úsporný: po k vrstvách bude mít všechny proměnné zafixované
- Úplný: nastaví (nějak) každou proměnnou
- Správný: Pro každé obdarování pak může za podmínky splnění obdarování dostat bitu $\left(x_i \overset{\text{AND}}{\diagup\diagdown} f(0,0,\dots 0, x_{i+1}, \dots x_n) \right)$ distribuovaný výsledek výše.