1. sada domácich úloh

Termín odovzdania: **streda 13. 11. 2024, 23:59**

Pravidlá pre domáce úlohy nájdete na `http://www.dcs.fmph.uniba.sk/~rajnik/udds/ulohy.html`.

(1,5 boda) Rozhodnite, či zložený výrok

$$[(a \Leftrightarrow (\neg b \Rightarrow c)) \Rightarrow (d \vee e)] \vee [(a \vee d) \Rightarrow (c \vee (b \wedge \neg e))]$$

je tautológia. Vaše tvrdenie dokážte.

Sporom ukážeme, že ide o tautológiu. Nech to tautológia nie je, teda pre nejaké ohodnotenie v premenných máme:

$\Rightarrow\ v([(a \Leftrightarrow (\neg b \Rightarrow c)) \Rightarrow (d \vee e)] \vee [(a \vee d) \Rightarrow (c \vee (b \wedge \neg e))]) = 0$

- $\Rightarrow\ v((a \Leftrightarrow (\neg b \Rightarrow c)) \Rightarrow (d \vee e)) = 0$
  - $\Rightarrow\ v(a \Leftrightarrow (\neg b \Rightarrow c)) = 1$ (1)
  - $\Rightarrow\ v(d \vee e) = 0$
    - $\Rightarrow\ v(d) = 0$
    - $\Rightarrow\ v(e) = 0$
- $\Rightarrow\ v((a \vee d) \Rightarrow (c \vee (b \wedge \neg e))) = 0$
  - $\Rightarrow\ v(a \vee d) = 1$ (2)
  - $\Rightarrow\ v(c \vee (b \wedge \neg e)) = 0$
    - $\Rightarrow\ v(c) = 0$
    - $\Rightarrow\ v(b \wedge \neg e) = 0$ (3)

Ďalej dostávame:

- $v(a) = 1$, lebo $v(a \vee d) = 1$ (2) a $v(d) = 0$
- $v(b) = 0$, lebo $v(b \wedge \neg e) = 0$ (3) a $v(w) = 0$.
- $v(a \Leftrightarrow (\neg b \Rightarrow c)) = 1 \Leftrightarrow (\neg 0 \Rightarrow 0) = 0$ – no to je spor s (1).

Komentár. Riešenie je zapísané pomerne stručne, nech vidíte, ako možno písať riešenia na skúškach. Taktiež je napísané iným štýlom ako v materiáloch k cvičeniam. Najprv sme naznačili pomocou odsadenia, čo z čoho vyplýva – tým sme spravili rozbor z toho, čo všetko vieme usúdiť z toho, že výrok je nepravdivý. Takto sme dostali ohodnotenie niektorých elementárnych výrokov a troch zložených výrokov (1), (2), (3). Na základe určeých elementárnych výrokov sme sa potom vedeli posunúť ´ďalej. Ak by sme chceli byť ešte strujčnejší, tak sa dali aj tieto zdôvodnenia nazanačiť šípkami.

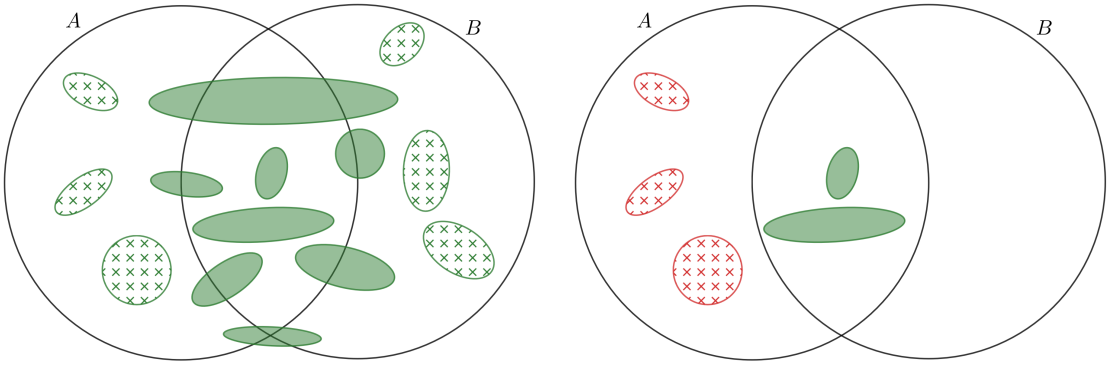
(1,5 boda) Rozhodnite a následne dokážte, či pre ľubovoľné dve množiny A, B platí:

a) $\mathcal{P}(A \cup B) - (\mathcal{P}(A - B) \cup \mathcal{P}(B - A)) \subseteq \mathcal{P}(A \cap B) - \mathcal{P}(A - B)$

b) $\mathcal{P}(A \cup B) - (\mathcal{P}(A - B) \cup \mathcal{P}(B - A)) \supseteq \mathcal{P}(A \cap B) - \mathcal{P}(A - B)$

Intuícia

Najskôr sa intuitívne zamyslime, ako vyzerajú podmnožiny, ktoré sa môžu vyskytnúť na ľavej a pravej strane.



Na ľavej strane máme na začiatku podmnožiny $A \cup B$ (označené zelenou), pričom odstránime také, ktoré sa celé zmestia do $A - B$ alebo $B - A$ (vyplnené krížikmi). Vyzerá to preto tak, že zostanú len také, ktoré majú prvky v prieniku $A \cap B$ alebo v aspoň dvoch „regiónoch“ diagramu.

Podobne vieme urobiť úvahy pre pravú stranu. Pôvodne sú v nej podmnožiny $A \cap B$ (označené zelenou), a potom odstránime také, ktoré sa celé zmestia do $A - B$ (vyplnené krížikmi). Vyzerá to tak, že sme žiadne podmnožiny neodstránili, lebo sme odstraňovali z „regiónu“, v ktorom sme pôvodne žiadne podmnožiny nemali.

Preto to vyzerá tak, že a) vo všeobecnosti nebude platiť a b) bude. Vieme teda, čo máme ísť dokazovať.

To, čo je uvedené v predošlých odsekoch, však nie je dôkaz. Dôvodom je, že vysvetlenia sú veľmi vágne naformulované, preto ľahko môžeme prehliadnuť nejaké podmnožiny. Napríklad by sme sa mohli dovtípiť, že $\mathcal{P}(A \cap B) - \mathcal{P}(A - B) = \mathcal{P}(A \cap B)$. To však neplatí, lebo každá potenčná množina obsahuje aj prázdnu množinu – a teda prázdna množina nebude prvkom ľavej strany, no bude prvkom pravej. Dôvod, prečo tvrdenia dokazujeme formálne, je, že nás to dosť často uchráni od takýchto chýb.

Podúloha a)

Keď chceme ukázať, že nejaká inklúzia neplatí vo všeobecnosti, stačí nájsť dve množiny A, B, pre ktoré neplatí. Z obrázkov v intuitívnych úvahách sa javí, že ak budeme mať nejaké prvky v $A \cap B$ aj v $A - B$, tak vzniknú nejaké podmnožiny, ktoré budú iba na ľavej strane. Skúsme si teda zobrať $A = \{1, 2\}$, $B = \{2\}$. Potom

$$\mathcal{P}(A \cup B) - (\mathcal{P}(A - B) \cup \mathcal{P}(B - A)) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\} - (\{\emptyset, \{1\}\} \cup \{\emptyset\}) = \{\{2\}, \{1, 2\}\},$$
$$\mathcal{P}(A \cap B) - \mathcal{P}(A - B) = \{\emptyset, \{2\}\} - \{\emptyset, \{1\}\} = \{\{2\}\}.$$

Vidíme, že pre tieto dve množiny $\mathcal{P}(A \cup B) - (\mathcal{P}(A - B) \cup \mathcal{P}(B - A)) \not\subseteq \mathcal{P}(A \cap B) - \mathcal{P}(A - B)$, preto táto inklúzia neplatí. Všimnime si ešte, že s druhou inklúziou zatiaľ problém nemáme.

Podúloha b)

Túto inklúziu by sme chceli dokázať. Na to by sme pre všetky množiny chceli vedieť, že ak $X \in \mathcal{P}(A \cap B) - \mathcal{P}(A - B)$, tak aj $X \in \mathcal{P}(A \cup B) - (\mathcal{P}(A - B) \cup \mathcal{P}(B - A))$. Pri takýchto úlohách je fajn si najskôr z definícií

(t. j. ekvivalentnými úpravami) prepísať tieto podmienky. Poďme teda na to.

$$
\begin{aligned}
&X \in \mathcal{P}(A \cap B) - \mathcal{P}(A - B), && \\
&X \in \mathcal{P}(A \cap B) \wedge \neg(X \in \mathcal{P}(A - B)), && \text{(definícia množinového rozdielu)} \\
&X \subseteq A \cap B \wedge \neg(X \subseteq A - B), && \text{(definícia potenčnej množiny)} \\
&(\forall x \in X : x \in A \cap B) \wedge \neg(\forall x \in X : x \in A - B), && \text{(definícia podmnožiny)} \\
&(\forall x \in X : x \in A \cap B) \wedge (\exists x \in X : \neg(x \in A - B)), && \text{(negácia kvantifikovaného výroku)} \\
&(\forall x \in X : (x \in A \wedge x \in B)) \wedge (\exists x \in X : \neg(x \in A \wedge x \notin B)), && \text{(definície množinových operácií)} \\
&(\forall x \in X : (x \in A \wedge x \in B)) \wedge (\exists x \in X : (x \notin A \vee x \in B)). && \text{(De Morganov zákon)}
\end{aligned}
$$

Všimnite si, že sme nerobili žiadne myšlienkové úvahy, išlo iba o priamočiaru aplikáciu definícií a pravidiel na zjednodušovanie negácií. Keď budeme podobne upravovať druhú podmienku (vyskúšajte si, je to dobré cvičenie), dostaneme, že

$$
\begin{gathered}
X \in \mathcal{P}(A \cup B) - (\mathcal{P}(A - B) \cup \mathcal{P}(B - A)) \\
\Leftrightarrow \\
(\forall x \in X : (x \in A \vee x \in B)) \wedge (\exists x \in X : (x \notin A \vee x \in B)) \wedge (\exists x \in X : (x \notin B \vee x \in A)).
\end{gathered}
$$

Chceme teda z predpokladu

$$
\underbrace{(\forall x \in X : (x \in A \wedge x \in B))}_{\text{(P1)}} \wedge \underbrace{(\exists x \in X : (x \notin A \vee x \in B)))}_{\text{(P2)}}
$$

dokázať záver

$$
\underbrace{(\forall x \in X : (x \in A \vee x \in B))}_{\text{(Z1)}} \wedge \underbrace{(\exists x \in X : (x \notin A \vee x \in B))}_{\text{(Z2)}} \wedge \underbrace{(\exists x \in X : (x \notin B \vee x \in A))}_{\text{(Z3)}}.
$$

Všimnime si, že (Z1) priamo vyplýva z (P1), lebo keďže všetky prvky musia patriť do A aj B, tak máme splnené pre ľubovoľný prvok obe strany disjunkcie (Z1) a na jej pravdivosť stačí splnenie jednej z nich.

Ďalej, (Z2) je to isté ako (P2), tým tento záver máme dokázaný priamo v predpoklade.

A nakoniec treba dokázať (Z3). Z (P2) vieme, že X nemôže byť prázdna. Z (P1) vieme, že všetky prvky X musia patriť do A. Preto určite máme prvok z X, ktorý patrí do A. A tým už máme pre nejaký prvok splnenú pravú stranu disjunkcie (Z3), čím je pravdivá celá.

Podarilo sa nám teda dokázať implikáciu

$$
(X \in \mathcal{P}(A \cap B) - \mathcal{P}(A - B)) \Rightarrow (X \in \mathcal{P}(A \cup B) - (\mathcal{P}(A - B) \cup \mathcal{P}(B - A))),
$$

pričom tento dôkaz bol nezávislý od voľby X, a preto máme aj

$$
\mathcal{P}(A \cap B) - \mathcal{P}(A - B) \subseteq \mathcal{P}(A \cup B) - (\mathcal{P}(A - B) \cup \mathcal{P}(B - A)),
$$

ako sme chceli.

Úloha 3

(2 body) V rade je uložených $2^n + 1$ políčok, pričom prvé a posledné políčko sú zasvietené a zvyšné políčka sú zhasnuté. Na prvom (najľavejšom) políčku sa nachádza robot, ktorému môžeme dávať nasledovné príkazy:

- `vpravo`: robot sa pohne o 1 políčko doprava (pokiaľ je na poslednom políčku, tak sa nepohne).

- `vlavo`: robot sa pohne o 1 políčko doľava (pokiaľ je na prvom políčku, tak sa nepohne).
- `zazni`: pokiaľ sa robot nachádza na políčku, ktoré je presne medzi dvomi zasvietenými políčkami, tak toto políčko zažne. V opačnom prípade sa nič nestane.

Dokážte, že pre všetky celé čísla $n \geq 0$, vieme pomocou $n \cdot 2^n$ alebo menej príkazov zažať všetky políčka.

Príklad. Pre $n = 3$ máme $2^3 + 1 = 9$ políčok, ktoré si môžeme očíslovať $0, 1, \ldots, 8$. Príkazy môžeme dávať napríklad takto:

príkaz	akcia robota
`vpravo`	Posun na políčko 1
`vpravo`	Posun na políčko 2
`vpravo`	Posun na políčko 3
`zazni`	Nič sa nestane, lebo políčko 3 sa nenachádza presne medzi políčkami 0 a 8
`vpravo`	Posunieme sa na políčko 4
`zazni`	Robot zažne políčko 4, lebo sa nachádza presne medzi políčkami 0 a 8
`vlavo`	Posun na políčko 3
`vlavo`	Posun na políčko 2
`zazni`	Robot zažne políčko 2, lebo sa nachádza presne medzi políčkami 0 a 4

Po týchto deviatich príkazoch svietia štyri políčka (0, 2, 4 a 8).

Bonus *(1 bod)* Napíšte program, ktorý zo vstupu načíta nezáporné celé číslo n a vypíše postupnosť príkazov (v každom riadku jeden príkaz), pomocou ktorej zasvietime všetky žiarovky. Vo vašom riešení aj zdôvodnite, prečo váš program funguje. (V ideálnom prípade sa stačí vhodne odvolať na riešenie úlohy 3).

To, že možno políčka rozsvietiť, ukážeme najľahšie tak, že opíšeme algoritmus pre robota. Náš dôkaz bude mať preto takú podobu. Pri algoritme budeme chcieť opakovať nejaké vzory – na to nám v matematike slúži matematická indukcia. Okrem opisu algoritmu aj ňou dokážeme, že algoritmus naozaj funguje a že vykoná požadovaný počet príkazov. Takémuto riešeniu priamočiaro zodpovedá aj program, ktorý ku koncu každého riešenia uvedieme.

Indukcia delením na polovice

Tvrdenie dokážeme matematickou indukciou.

Báza. Pre $n = 0$ máme $2^0 + 1 = 2$ políčka, ktoré obe svietia. Preto nemusíme robiť nič, čím sme úlohu vyriešili na $0 \leq 0 \cdot 2^0$ príkazov.

Indukčný krok. Uvažujme teraz celé $n \geq 0$. Predpokladajme, že pre n tvrdenie platí, teda že rad $2^n + 1$ políčok, kde prvé a posledné políčko sú zasvietené, vieme zažať na $n \cdot 2^n$ alebo menej príkazov.

Dokážeme, že tvrdenie platí aj pre $n + 1$, teda, že rad $2^{n+1} + 1$ políčok, kde prvé a posledné políčko sú zasvietené, vieme zažať na $(n+1) \cdot 2^{n+1}$ alebo menej príkazov. Políčka očíslujeme $0, 1, \ldots, 2^{n+1}$. Rozsvietime ich nasledovne:

1. Pomocou 2^n príkazov `vpravo` presunieme robota na prostredné políčko (s číslom $2^n = (0 + 2^{n+1})/2$).
2. Zažneme prostredné políčko: 1 príkaz `zazni`.
3. Rozsvietime všetkých $2^n + 1$ políčok od 0 po 2^n – to je možné vďaka indukčnému predpokladu (obe krajné svietia) a použijeme na to najviac $n \cdot 2^n$ príkazov.
4. Pomocou niekoľkých príkazov `vpravo` vrátime robota na stredné políčko. Keďže políčko 0 už svietilo, tak robot na ňom skončil, preto nám stačí $2^n - 1$ posunov `vpravo`.

5. Rozsvietime všetkých $2^n + 1$ políčok od 2^n po 2^{n+1} – to je možné vďaka indukčnému predpokladu (obe krajné svietia) a použijeme na to najviac $n \cdot 2^n$ príkazov.

Týmto sme rozsvietili všetky žiarovky a požili sme pritom najviac

$$(2^n + 1) + n \cdot 2^n + (2^n - 1) + n \cdot 2^n = 2 \cdot 2^n + 2 \cdot n \cdot 2^n = 2^{n+1} + n \cdot 2^{n+1} = (n + 1) \cdot 2^{n+1} \text{ príkazov,}$$

čo sme aj chceli dokázať. Dôkaz indukciou je tak ukončený.

Program. Pri písaní programu narazíme na istý problém – úplne technicky nie je správne začať riešiť problém pre $n + 1$ pohybmi vpravo. Prísne vzaté, nám to vraví, že keď ideme využiť indukčný predpoklad, tak ľavú polovicu začneme tiež tým, že sa hýbeme vpravo. Avšak do stredu ľavej časti sa potrebujeme pohnúť pohybom vpravo. Tento detail sa stratil v slovnom opise riešenia. Môžeme si predstaviť, že sme akoby prečíslovali ľavú časť, aby stred (teda začiatočná pozícia robota v ľavej časti), mal číslo 0. Týmto sme vymenili vľavo a vpravo. To si v písanom texte ľahko predstavíme, ale pri programovaní je to náročnejšie. Na vyriešenie tohto problému väčšinou príde vhod pridanie ďalších parametrov do našej rekurzívnej funkcie. Napr. či `bool` parametra, ktorý nám hovorí, či sa hýbeme vpravo alebo vľavo.

Uvedieme riešenie, kde pridáme dva parametre `zaciatok` a `koniec`, ktoré nám udávajú začiatok a koniec úseku, ktorý robot má rozsvietiť, pričom sa nachádza na pozícii `zaciatok`. Oproti algoritmu opísanom v indukčnom kroku nespravíme veľa zmien. Najväčšou je to, že sa nehýbeme vždy vpravo, ale smer pohybu zistíme podľa vzájomnej polohy začiatku a konca.

Okrem napísania samotného algoritmu si doňho vieme dať pomocné funkcionality, ktoré nám budú kontrolovať, že robíme to, čo máme. Priebežne si pamätáme pozíciu robota a pre každé políčko, či je zasvietené. Tak si vieme skontrolovať, že zažínané políčko je vždy medzi dvomi zasvietenými. Podobne si skontrolujeme aj, či všetky políčka na konci svietia a či sme použili správny ťahov. Takáto kontrola je užitočná (alebo aspoň ručná kontrola na malých vstupcoh) – niektorý študenti totiž odovzdali programy, ktoré ani na malých vstupoch nedávali správny výsledok. To by sa vám nemalo pri programátorských úlohách stávať.

Všimnite si v uvedenom programe jeho podobnosť s dôkazom v indukčnom kroku. Tým, že sme túto podobnosť dodržali, tak náš dôkaz z 3. úlohy aj dokazuje správnosť tohto programu. (Úplne poriadny dôkaz správnosti to nie je, ale taký nám stačil pre potreby tohto predmetu.)

```
#include <cassert>
#include <iostream>
#include <vector>

using namespace std;

// pomocne globalne premenne, v ktorych si udrziavame stav planu
int pozicia = 0;
vector<bool> svieti;
int dlzka;
int prikazy = 0;

// pohyb vpravo - okrem vypisania aktualizujeme nase pomocne premenne
void vpravo() {
    if (pozicia < dlzka) pozicia++;
    cout << "vpravo\n";
    prikazy++;
}

// pohyb vlavo
void vlavo() {
    if (pozicia > 0) pozicia--;
    cout << "vlavo\n";
    prikazy++;
}
```

```
// Funkcia, ktora pre zadane n vypise postupnosti priazov. Funguje na rekurzivnom principe opisanom v
    rieseni. Do funkcie sme pridali dva parametre: zaciatok (pociatocna pozicia robota) a koniec (koniec
    useku, ktory robot rozsvecuje).
void solve(int n, int zaciatok, int koniec) {
    if (n == 0) return; // Vtedy nic nerobime

    // stredne policko
    int stred = (zaciatok + koniec) / 2;
    assert((zaciatok + koniec) % 2 == 0); // kontrola, ci stredne policko existuje
    // 1. Presun do stredu: 2^(n - 1) prikazov vpravo, resp. vlavo
    for (int i = 0; i < 1 << (n - 1); i++) {
        if (zaciatok < koniec) vpravo();
        else vlavo();
    }
    // 2. Zazneme prostredne policko
    assert(svieti[zaciatok] and svieti[koniec]); // kontrola, ze sa nachdza medzi dvomi zasvietenymi
    cout << "zazni\n";
    svieti[stred] = true;
    prikazy++;
    // 3. Rozsvietime lavu cast, teda od stred (tu sa teraz nachadzame) po zaciatok
    solve(n - 1, stred, zaciatok);
    // Vratime sa do stredu - nevieme kolko prikazov vykoname, ale bude to najviac 2^(n - 1) - 1
    while (pozicia != stred) {
        if (zaciatok < koniec) vpravo();
        else vlavo();
    }
    // 3. Rozsvietime pravu cast, teda od stred (tu teraz sme) po koniec
    solve(n - 1, stred, koniec);
}

int main() {
    int n;
    cin >> n;
    dlzka = (1 << n) + 1;
    svieti = vector<bool>(dlzka, false);
    svieti[0] = true;
    svieti[dlzka - 1] = true;
    solve(n, 0, dlzka - 1);

    // Zavercna kontrola
    for (int i = 0; i < dlzka; i++) assert(svieti[i]);
    assert(tahy < n * (1 << n));
}
```

Technický detail: pre počítanie mocnín dvojky používame bitový posun (čo je najefektívnejší spôsob). Teda 2^n počítame ako `1 << n`.

Riešenie zovšeobecnením dokazovaného tvrdenia

Predošlé riešenie má problém, že nevieme, kde robot skončí po použití indukčného predpokladu. Tento problém sme vyriešili tým, že sme tento počet odhadli číslom $2^n - 1$. Iným riešením tohto problému je zovšeobecniť dokazované tvrdenie, aby sme vedeli, kde presne skončíme. Spôsobov zovšeobecnenia je viacero. Jeden z pomerne efektívnych je nasledovný.

Dokážeme, že pre každé $n \in \mathbb{N}^+$ platí: Nech máme políčka $0, 1, \ldots, 2^n$, pričom políčka 0 a 2^n svietia a robot začína na políčku 0. Potom robot vie s využitím najviac $n \cdot 2^n$ príkazov zažať všetky políčka. Navyše, robotovi vieme určiť, či skončí na políčku 1 alebo $2^n - 1$.

Dokonca, ak by sme chceli, môžeme dokazovať, že robot vykoná presne $(n + 3)2^{n-1} - 2$ príkazov. Toto tvrdenie možno opäť dokázať indukciou. Naznačíme len indukčný krok (za predpokladu, že tvrdenie platí pre $n - 1$) pre prípad, že robot má skončiť na políčku 1:

1. Pohneme robota do stredu: 2^{n-1} príkazov.

2. Zažneme stred: 1 príkaz.

3. Použijeme IP na pravú polovicu, pričom robotu vyberieme, nech skončí hneď naľavo od miesta, kde začal (teda na políčku $2^{n-1}+1$: $(n+2)2^{n-2}-2$ príkazov.

4. Vrátime robota do stredu: 1 príkaz

5. Použijeme IP na ľavú polovicu, pričom robotu vyberieme, nech skončí na opačnej strane úseku (teda na políčku 1), ako začal: $(n+2)2^{n-2}-2$ príkazov.

Tým všetko rozsvietime na

$$2^{n-1}+1+(n+2)2^{n-2}-2+1+(n+2)2^{n-2}-2=2^{n-1}+(n+2)2^{n-1}-2=(n+3)2^{n-1}-2 \text{ príkazov.}$$

Riešenie dvomi prechodmi

Predhovor. Na záver ukážeme ešte jedno efektívnejšie riešenie (s rádovo menším počtom príkazov) a aj založené na pomerne jednoduchej idei: Robot najprv pôjde doprava, pričom zažne každé políčko, ktoré bude môcť. Potom robot ide vľavo a zažne zvyšné políčka. Avšak nie je úplne jednoduché dokázať, že je správne. Správnosť oboch častí dokážeme matematicou indukciou. Na to treba sformulovať vhodné tvrdenie, ktoré budeme dokazovať. To bude záhrňať aj pozíciu robota, aby sme mohli tvrdenia potom spojiť dohromady.

Políčka si očíslujeme zľava doprava $2^n, 2^n-1, \ldots, 1, 0$, pričom robot začína na políčku 2^n. Počítanie ťahov si rozdelíme na počítanie príkazov `zazni` a pohybov, teda príkazov `vlavo` a `vpravo`. Vždy keď vykonáme príkaz `zazni`, tak skontrujeme, či sa svetlo rozsvieti – teda či ho vôbec možno rozsvietiť a tiež, či ho nerozsvecujeme podruhýkrát. Takto zaručíme, že príkazov `zazni` vykonáme presne 2^n-1 (toľko, koľko je zhasnutých políčok). Vo zvyšku riešenia tak budeme počítať len počet pohybov.

Tvrdenie 1. *Pre každé $n \in \mathbb{N}$ platí: Nech máme políčka $2^n, \ldots, 1, 0$, z ktorých krajné svietia a robot začína na políčku 2^n. Potom robot vie rozsvietiť políčka, ktoré sú mocniny dvoch, na 2^n-1 pohybov, pričom skončí na políčku 1.*

Dôkaz. Indukciou. Pre $n=0$ sme už hotoví a nič netreba robiť. Tvrdenie teraz dokážeme pre $n \geq 1$ za predpokladu, že platí pre $n-1$. Robotom pohneme 2^{n-1}-krát `vpravo`, čím ho dostaneme na políčko 2^{n-1}, ktoré zažneme – je totiž medzi zasvietenými 2^n a 0 (a také políčko existuje, keďže $n \geq 1$). Ostali nám políčka $2^{n-1}, \ldots, 1, 0$, z ktorých krajné svietia. Podľa indukčného predpokladu ich vieme zvyšné políčka rozsvietiť na $2^{n-1}-1$ pohybov (a aj skončiť na políčku 1). Teda spolu máme $2^{n-1}+2^{n-1}-1=2^n-1$ pohybov. □

Tvrdenie 2. *Pre každé $k \in \mathbb{N}^+$ platí: Pre všetky $n \in \mathbb{N}$ platí: Nech máme políčka $2^n, \ldots, 1, 0$, z ktorých svietia všetky mocniny dvoch a políčko 0. Potom ak sa robot nachádza na políčku 2^n-k a všetky políčka s číslom 2^n-k alebo nižším sú rozsvietené, tak robot vie zažať všetky políčka na $k-1$ pohybov.*

Dôkaz. Pre $k=1$ sa robot nachádza na políčku 2^n-1, ktoré podľa predpokladu tvrdenia svieti a taktiež svietia aj všetky políčka s nižším číslom. Vyššie číslo má len políčko 2^n, ktoré tiež svieti. Nie je teda potrebné robiť nič.

Ďalej dokážeme platnosť tvrdenia pre $k>1$ za predpokladu, že platí pre $k-1$. Pozíciu robota označme $p=2^n-k$. Pre $k>2^k$ je predpoklad vety nepravdivý, preto automaticky platí. Ďalej uvažujeme len $k \leq 2^k$. Robota posunieme príkazom `vlavo` na políčko $p+1$. Ak políčko $p+1$ nesvieti, tak ukážeme, že ho robot vie zažať.

Nech ℓ je najmenšie také číslo, pre ktoré platí $2^\ell > p+1$ (teda 2^ℓ je najbližšia mocnina dvoch naľavo od robota). Preto $p+1 \geq 2^{\ell-1}$. Vzdialenosť medzi políčkami 2^ℓ a $p+1$ je $2^\ell-(p+1) \leq 2^\ell-2^{\ell-1}=2^{\ell-1} \leq p+1$. Keďže $p+1 \geq 2^{\ell-1}$, tak v rade políčok vieme nájsť políčko napravo, ktoré bude v rovnakej vzdialenosti – teda ide o políčko $p+1-2^{\ell-1}$. Podľa predpokladu tvrdenia sú obe z políčok 2^ℓ a $p+1-2^{\ell-1}$ zasvietené, a preto vieme zasvietiť aj políčko $p+1$.

Teraz máme robota na políčku $p+1 = 2^n - (k-1)$ a všetky políčka napravo vrátane políčka $p+1$ svietia. Preto podľa indukčného predpokladu robot vie rozsvietiť všetky políčka na $k-2$ pohybov. Spolu s jedným pohybom vľavo, ktorým sme robota presunuli na políčko $p+1$, tak dostávame $k-1$ pohybov, ako sme chceli. □

Teraz už len spojíme tieto tvrdenia do úplného dôkazu pre $n \geq 1$ – pre $n = 0$ nám zjavne stačí $0 \leq 0 \cdot 2^0$ príkazov. Najprv podľa tvrdenia 1 rozsvietime všetky mocniny dvoch s vykonaním $2^n - 1$ pohybov a skončíme s robotom na políčku 1. Políčko $1 = 2^n - (2^n - 1)$ teraz svieti a tiež aj všetky políčka napravo od neho (menovite políčko 0). Máme teda splnené predpoklady tvrdenia 2, a preto vieme všetky políčka rozsvietiť na $2^n - 1 - 1 = 2^n - 2$ pohybov. Teraz spočítame všetky pohyby a pripočítame k nim $2^n - 1$ príkazov `zazni`, čím dostaneme celkovo

$$(2^n - 1) + (2^n - 2) + (2^n - 1) = 3 \cdot 2^n - 4 \text{ pohybov.}$$

Na záver už len dokážeme, že platí

$$3 \cdot 2^n - 4 \leq n \cdot 2^n.$$

To je totiž ekvivalentné nerovnosti $-4 \leq (n-3) \cdot 2^n$. Ručne overíme, že platí pre $n \in \{1, 2\}$. Pre $n \geq 3$ je $(n-3) \cdot 2^n \geq 0$, čo je viac ako -4.

Poznámka k tvrdeniu 2. V tomto riešení sme uviedli tvrdenie veľmi poriadne. Idea však nie je až taká zložitá. Počas pohybu vľavo si udržiavame, že všetky políčka napravo od robota svietia. Keď sa nám toto podarí zachovať, tak po príchode na ľavý kraj bude všetko rozsvietené. A v zásade presne aj to robíme v dôkaze – predpokladáme, že už všetko vľavo je zasvietené a ukážeme, že po pohybe vľavo vieme zasvietiť aj toto nové políčko. Indukcia nám tu slúži len na formálne vyjadrenie opaokovania tohto procesu.

Program. Toto riešenie zodpovedá nasledovnému programu.

```
#include <cassert>
#include <iostream>
#include <vector>

using namespace std;

void doprava(int n) {
    if (n > 0) {
        for (int i = 0; i < 1 << (n - 1); i++) {
            cout << "vpravo\n";
        }
        cout << "zazni\n";
        doprava(n - 1);
    }
}

void dolava(int n, int k) {
    if (k > 1) {
        cout << "vlavo\n";
        int pozicia = (1 << n) - k + 1;
        if ((pozicia & (pozicia - 1)) != 0) { // ak pozicia nie je mocnina dvoch
            cout << "zazni\n";
        }
        dolava(n, k - 1);
    }
}

int main() {
    int n;
    cin >> n;
    doprava(n);
    dolava(n, (1 << n) - 1);
}
```

Z programátorského hľadiska ide o veľmi zložito napísané riešenie. Rekurzívne funkcie sa dajú nahradiť jednoduchými for cyklami. Taktiež použitie parametru k vo funkcii `dolava` je veľmi ťažkopádne. Tento program však nemá za cieľ dodržiavať dobré programátorské zásady. Je napísaný, aby ilustroval fungovanie nášho dôkazu. Potom otestovaním tohto programu si vieme skontrolovať a aj lepšie predstaviť správnosť nášho matematického dôkazu.

V tomto programe sme nepísali pomocné príkazy na kontrolu. Avšak možno ich tam dopísať na podobný štýl ako v predošlom riešení. Alebo si môžeme napísať samostatný prgoram, ktorý nám bude kontrolovať výstup tohto programu.