Obsah

1. Východiská

V tejto kapitole sa budeme venovať teoretickým základom potrebným pre túto prácu. Začneme s definíciami relevantných pojmov a nasledovať bude vysvetlenie Exact cover problému, Algoritmu X, technike Dancing links a použitia Dancing links pri Algoritme X. Na konci kapitoly si ešte uvedieme ako riešiť 3D drevené hlavolamy pomocou redukcie do Exact cover problému a Algoritmu X.

1.1 NP-úplný problém

NP predstavuje skratku pre nedeterministický polynomiálny čas. Trieda NP problémov je charakteristická tým, že správnosť riešení týchto problémov je dokázateľná v polynomiálnom čase. NP-úplné problémy [4, s. 452] sú najnáročnejšie z pomedzi triedy NP, nie sú zatiaľ riešiteľné v polynomiálnom čase. Ak by sa však niekedy našlo riešenie jedného NP-úplného problému, ktoré má polynomiálnu zložitosť, budú vyriešené všetky tieto problémy. Ide o jedny z najviac riešených problémov súčasnej informatiky. Ak o probléme je preukázané, že je NP-úplný, tak jeho riešenie bude mať minimálne exponenciálnu zložitosť. Klasickým príkladom takéhoto problému je Exact cover problém.

1.2 Spájaný zoznam

1.2.1 Jednosmerne spájaný zoznam

Ide o najjednoduchší spomedzi všetkých druhov spájaných zoznamov. Jednosmerný spájaný zoznam [5] je definovaný pomocou objektu Node, ktorý ma atribúty názov (name) a smerník na nasledujúci objekt typu Node (next). Ak máme referenciu na prvý vrchol spájaného zoznamu, tak pomocou atribútu next ho vieme celý prejsť. Ak však nejaký vrchol (Node) má nejaký predchádzajúci prvok, tak ku tomuto sa už dostať nedá, keďže referencie sú iba jednosmerné.

1.2.2 Dvojsmerne spájaný zoznam

Tento druh spájaného zoznamu je vylepšením jednosmerného. Definovaný je [5] pomocou objektu Node, ktorý na rozdiel od prechádzajúceho typu má ešte jeden smerník, nazvaný prev, odkazujúci na predchádzajúci vrchol spájaného zoznamu. Toto poskytuje veľkú

výhodu, pretože si stačí pamätať referenciu na ľubovoľný vrchol zoznamu a pomocou dvojsmerných smerníkov sa dá dostať ku všetkým ostatným vrcholom.

1.2.3 Cyklický spájaný zoznam

V prvých dvoch typoch spájaných zoznamov mal zoznam vždy začiatok a koniec. Cyklický spájaný zoznam [5] nemá ani jedno z toho. Dá sa spraviť z ľubovoľného necyklického spájaného zoznamu tým, že ako nasledovník posledného vrcholu sa dá prvý a ak ide o dvojsmerný zoznam, tak ako predchádzajúci prvok prvého prvku sa dá posledný. Týmto vznikne cyklus. Takto sa aj v jednosmerne spájanom zozname vieme dostať ku všetkým vrcholom, bez ohľadu na to, na ktorý máme referenciu.

1.3 Exact cover problém

Majme množinu prvkov S, Exact cover, alebo úplné pokrytie množiny S, je taká kolekcia S_1, S_2... S_n podmnožín množiny S, že zjednotenie množín S_1 až S_n tvorí S a ich prienik je prázdna množina.

Knuth tento problém ilustruje pomocou binárnej matice [1, s. 64].

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Obr. 1: binárna matica A

Na obrázku 1 možno vidieť binárnu maticu A, ktorá ma sedem stĺpcov, ktoré pomenujeme A až G a má šesť riadkov, ktoré pomenujeme 1 až 6. Stĺpce predstavujú prvky (items- prvky množiny S), ktoré treba pokryť a riadky možnosti (options). Každá možnosť je nejaká podmnožina všetkých prvkov (množiny S), ak je v riadku jednotka, daná možnosť obsahuje daný prvok, ak tam je nula, neobsahuje ho. Riešením podľa Knutha [1, s. 64] je taká kolekcia disjunktných riadkov matice, ktorá bude mať v každom stĺpci presne jednu jednotku. Po

krátkom preskúmaní je jasné, že jediným riešením je zjednotenie riadkov 1,4 a 5, ktoré tvorí celú množinu S.

1 = {C, E}

4 = {A, D, F}

5 = {B, G}

1 ∪ 4 ∪ 5 = {A, B, C, D, E, F, G} = S

Exact cover problém [2, s. 2] je NP-úplný už pri výskyte aspoň troch jednotiek v každom riadku matice. Toto naznačuje, že riešenia takýchto problémov už aj pri malom rozmere, vedia byť náročne na výpočet.

1.4 Algoritmus X

Je to nedeterministický algoritmus vytvorený Donaldom Knuthom na nájdenie všetkých riešení Exact cover problému definovaného binárnou maticou A [2, s. 3]. Podstatou algoritmu je postupné skúšanie možností metódou pokus-omyl. Pseudokód algoritmu vyzerá nasledovne [2, s. 4]:

Ak A je prázdne, problém je vyriešený, program sa úspešne ukončí.
V opačnom prípade sa deterministicky vyberie stĺpec c.
Nedeterministicky sa vyberie riadok r, taký, že A[r, c] = 1.
r sa priradí do čiastočného riešenia.
Pre každé j také, že A[r, j] = 1,
    zmaž stĺpec j z matice A
    pre každé i také, že A[i, j] = 1
        zmaž stĺpec i z matice A
opakuj rekurzívne algoritmus na redukovanej matici A.

Nedeterministická voľba r [2, s. 4] znamená, že sa algoritmus sa naklonuje do nezávislých subalgoritmov, každý subalgoritmus zdedí súčasnú verziu matice A, ale redukuje ju podľa vlastného výberu r. Subalgoritmy vytvoria vyhľadávací strom, s originálnym problémom v koreni a s úrovňou k obsahujúcou každý k-tý subalgoritmus korešpondujúci s k vybranými

riadkami. Backtracking je proces prechádzania tohto stromu do hĺbky v postupnosti preorder.

1.5 Dancing links

Dancing links je technológia vymyslená Donaldom Knuthom založená na jednoduchom princípe mazania a opätovného vracania vrcholov v dvojsmernom cyklickom spájanom zozname. Táto technika ma veľké využitie pri backtrackingu, kde treba jednoducho odrobiť veľa zmien a veľmi efektívne zrýchľuje aj Algoritmus X [3].

Majme dvojsmerný cyklický spájaný zoznam, kde má každý vrchol X svoj predchádzajúci vrchol LLINK(X) a nasledujúci vrchol RLINK(X). Vrchol X sa dá jednoducho vymazať zo zoznamu tak, že nastavíme [1, s. 63]:

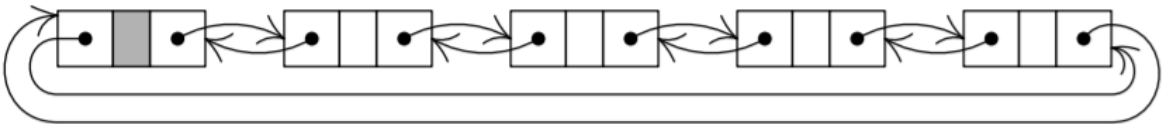
$$\text{RLINK}(\text{LLINK}(X)) = \text{RLINK}(X), \quad \text{LLINK}(\text{RLINK}(X)) = \text{LLINK}(X)$$

Zvonku sa potom javí, že vrchol X v zozname nie je. Smerníky prvku X ale stále ukazujú na RLINK(X) a LLINK(X). Síce by normálne bolo zvykom aj tieto vnútorne smerníky vynulovať, ale pre Dancing links techniku je veľmi podstatné, že tieto smerníky zostanú nedotknuté, ako sa ukáže pri opätovnom vracaní prvku X do zoznamu. Vymazanie vrcholu sa da teda odrobiť ďalšou jednoduchou dvojicou operácii [1, s. 63]:

$$\text{RLINK}(\text{LLINK}(X)) = X, \qquad \text{LLINK}(\text{RLINK}(X)) = X$$

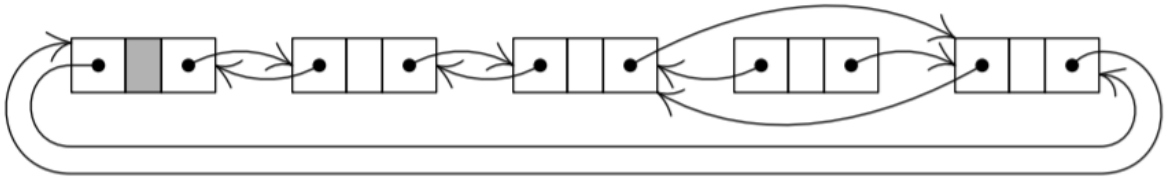
Stačí si teda pamätať iba smerník na X na to aby mohol byť opätovne vrátení do zoznamu na miesto, kde pôvodne bol.

Predvedieme si ako táto technika funguje na malom ilustračnom príklade. Uvažujme 4-prvkový spájaný zoznam s hlavičkou (špeciálnym vrcholom zoznamu, ktorý zvykne byť zapamätaný v premennej, ale inak nemá žiadny iný účel) z obrázku 2 [1, s. 63]:



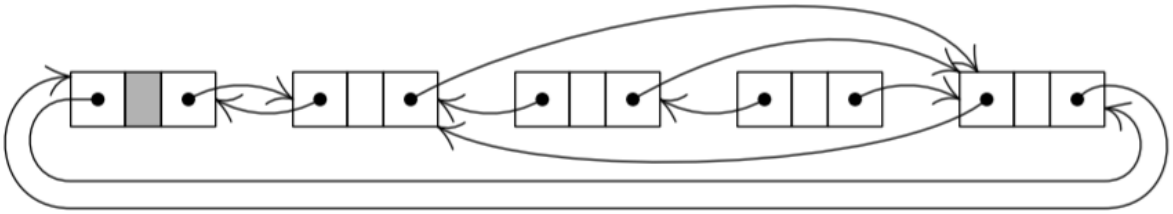
Obr. 2: dvojstranný cyklický spájaný zoznam

Obrázok 3 potom zobrazuje ako budú vyzerať smerníky po vymazaní tretieho vrcholu



Obr. 3: zoznam po vymazaní 3. prvku

Po vymazaní ešte vrcholu 2 bude zoznam vyzerať ako na obrázku 4.



Obr. 4: zoznam po vymazaní 3. a 2. prvku

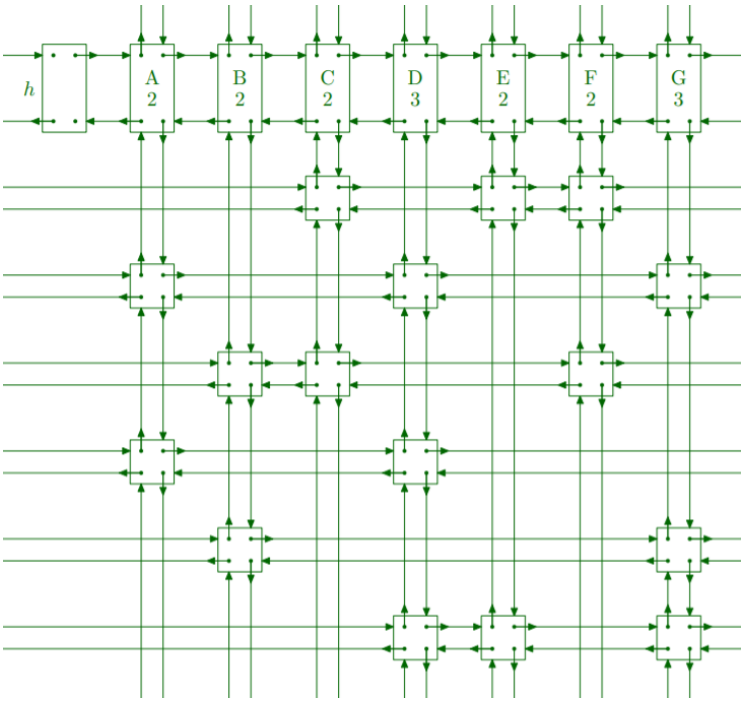
1.6 Exact cover problém pomocou 4-smerného cyklického spájaného zoznamu

Na to aby sme mohli použiť využiť Dancing links pri riešení Exact cover problému pomocou Algoritmu X, je najprv potrebné aby sme binárnu maticu prerobili do 4-smerného cyklického spájaného zoznamu.

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Obr. 5: binárna matica

Obrázok 6 zobrazuje ako by sa matica z obrázku 5 prerobila do takejto dátovej štruktúry [2, s. 6]. Je to vlastne štruktúra horizontálnych a vertikálnych cyklických spájaných zoznamov. Prvý horizontálny spájaný zoznam predstavuje prvky množiny S, ktoré sa budú pokrývať. Prvý prvok tohto zoznamu je hlavička (header) zoznamu, referencia na túto hlavičku bude zapamätaná v premennej. Každý prvok X tohoto prvého horizontálneho spájaného zoznamu má RLINK(X), LLINK(X), ULINK(X), DLINK(X), LEN(X) a NAME(X), hlavička má len prvé dva. Tieto atribúty predstavujú ľavý, pravý, horný a dolný vrchol prislúchajúce ku X. Posledné dve predstavuje počet možností (options), ktoré pripadajú ku danému prvku (itemu) a názov prvku. Všetky vertikálne cyklické spájané zoznamy spájajú prvky a možnosti, ktoré obsahujú daný prvok. Zvyšné horizontálne cyklické spájané zoznamy predstavujú jednotlivé možnosti.



Obr. 6: 4-smerný cyklický spájaný zoznam

Riadky binárnej matice predstavujúce možnosti sa transformujú do horizontálnych cyklických spájaných zoznamov tak, že z jednotkových prvkov sa stanú vrcholy zoznamu a nulové sa ignorujú. Vrcholy predstavujúce možnosti majú atribúty RLINK(X), LLINK(X), ULINK(X), DLINK(X) a TOP(x). TOP predstavuje referenciu na vrchol z prvého horizontálneho zoznamu, ku ktorému vrchol z danej možnosti prilieha.

1.7 Algoritmus X s použitím Dancing Links

Majme Exact cover problém transformovaný do 4-smerného cyklického spájaného zoznamu ako bolo ukázane v podkapitole 1.6. Aby sme mohli použiť Algoritmus X s pomocou Dancing links na tento problém uvedieme si najprv pomocné funkcie pracujúce s touto dátovou štruktúrou [1, s. 63, 64]:

cover(i) (i predstavuje item) =

Nastav p = DLINK(i). (p, l a r sú lokálne premenné.)

Dokiaľ p != i, hide(i), následne nastav p = DLINK(p) a opakuj.

Nastav l = LLINK(i), r = RLINK(i),

RLINK(l) = r, LLINK(r) = l

hide(p) =

Nastav q = RLINK(p) a opakuj, kým q != p:

Nastav u = ULINK(q), d = DLINK(d), x = TOP(q).

Nastav DLINK(u) = d, ULINK(d) = u

LEN(x) = LEN(x) – 1, q = RLINK(q)

Ďalej nasledujú 2 funkcie [1, s. 64] ktoré pomocou technológie Dancing link odrobia zmeny, ktoré prvé 2 funkcie vykonajú. Môžeme si všimnúť, že odrábajú zmeny presne v opačnom poradí ako boli spravené, pretože iba takto sa dátová štruktúra vráti do predchádzajúcej verzie. Pseudokód týchto funkcii vyzerá nasledovne:

uncover(i) =

Nastav l = LLINK(i), r = RLINK(i),

RLINK(l) = i, LLINK(r) = i.

Nastav p = ULINK(i).

Dokiaľ p != i, unhide(i), potom nastav p = ULINK(p) a opakuj

unhide(p) =

Nastav q = LLINK(p) a opakuj, kým q != p:

Nastav u = ULINK(q), d = DLINK(d), x = TOP(q).

Nastav DLINK(u) = q, ULINK(d) = q

LEN(x) = LEN(x) + 1, q = LLINK(q)

Samotný Algoritmus X (v tejto implementácii ide už o deterministický algoritmus) by sa dal napísať takto [1, s. 64]:

X1. [Inicializácia] Nastav problém v pamäti, tak ako obrázku 5. Nastav l = 0

X2. [Vstúp do úrovne l] Ak RLINK(header) = header (teda už boli pokryté všetky prvky), navštív riešenie špecifikované $x_0x_1x_2...x_{l-1}$ a choď do X8

X3. [Vyber i] v tomto čase ešte musia byť prvky i_1, ..., i_t pokryté, kde i_i = RLINK(header), i_{j+1} = RLINK(i_j) a RLINK(i_t) = header. Vyber jeden z týchto prvkov a nazvi ho i. (O tom ako vybrať prvok i sa pojednáva v podkapitole 1.8.)

X4. [Pokry i] Pokry prvok i pomocou funkcie cover(i) a nastav x_l = DLINK(i)

X5. [vyskúšaj x_l] Ak x_l = i, choď do X7 (vyskúšali sme všetky možnosti). V opačnom prípade nastav p = RLINK(x_l) a rob nasledovné dokiaľ p != x_l:

Nastav j = TOP(p).

cover(j), nastav p = RLINK(p) (toto zakryje všetky prvky != i v možnosti ktorá obsahuje x_l).

Nastav l = l + 1 a vráť sa do X2.

X6. [Vyskúšaj opäť] Nastav p = LLINK(x_l) a rob nasledujúce dokiaľ p != x_l:

Nastav j = TOP(p).

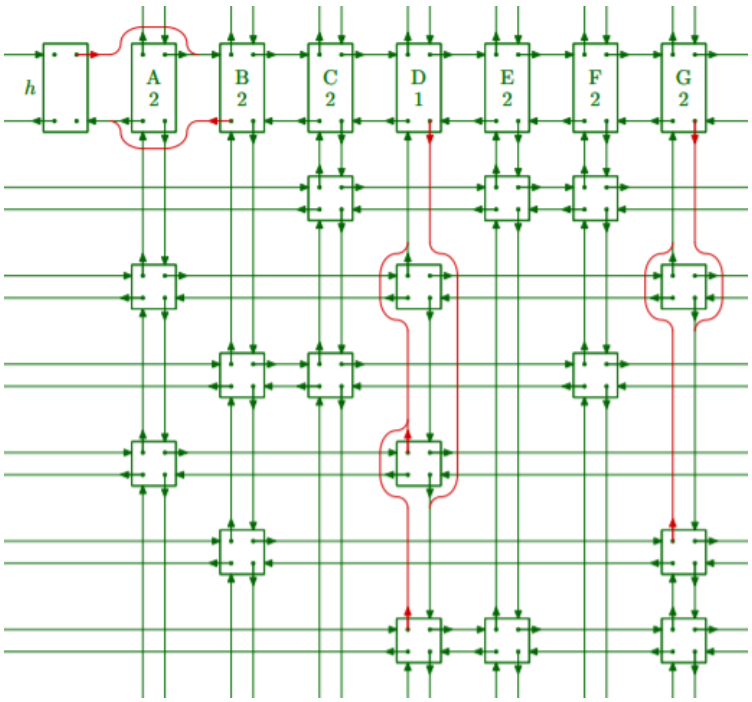
uncover(j) a nastav p = LLINK(p). (Toto odokryje všetky prvky != i v možnosti, ktorá obsahuje x_l, pomocou reverznej operácii ku X5)

Nastav i = TOP(x_l), x_l = DLINK(x_l) a vráť sa do X5.

X7. [Backstrackuj] Odkry prvok i pomocou funkcie uncover(i).

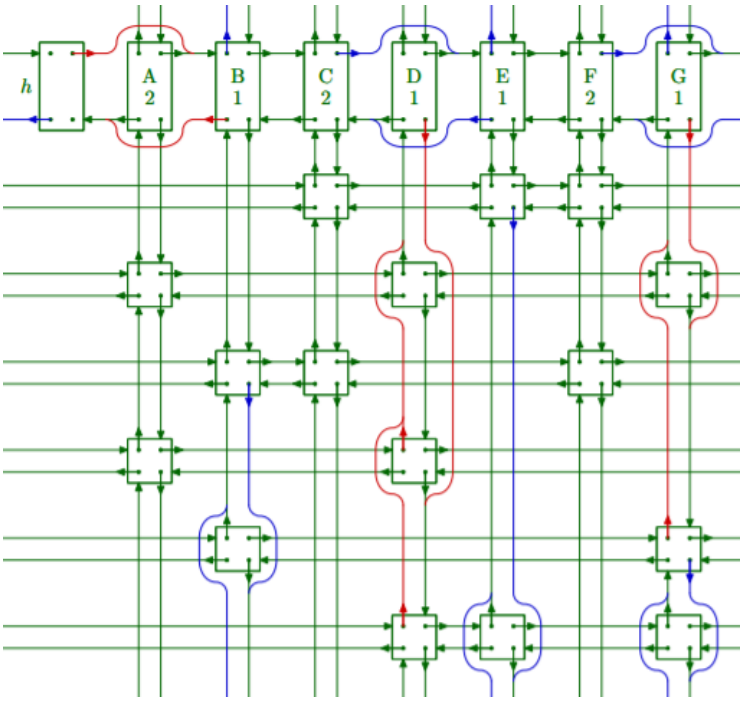
X8. [Opusti úroveň l] Ukonči ak l = 0. V opačnom prípade nastav l = l – 1 a vráť sa do X6.

Algoritmus si môžeme predviesť na príklade z obrázku 6. Predpokladajme, že na úrovni 0 sa vyberie prvok A (na poradí výberu prvkov v konečnom dôsledku nezáleží, ale pre ilustráciu problému sme si zvolili prvok A) do premennej i a v kroku X4 sa zavolá cover(A). Dátová štruktúra po týchto krokoch bude vyzerať ako na obrázku 7.



Obr. 7: Spájané zoznamy po pokrytí prvku A

Je vidno, že prvok A je pokrytý a rovnako aj všetky vrcholy, ktoré pripadali do jednotlivých možností obsahujúcich prvok A okrem vrcholov pripadajúcich ku samotnému vrcholu A. Tieto sú ale pokryté tým, že vrchol za je pokrytý.



Obr. 8: Spájané zoznamy po pokrytí prvkov A, D a G

Do x_0 sa vyberie možnosť ’A, D, G’. Po kroku X5 sa zakryjú aj prvky D a G a spájané zoznamy budú mať podobu z obrázku 8. Potom algoritmus vkročí do druhej úrovne.

V tej zostávajú už iba 2 možnosti: ’C, E, F a ’B, C, F’. A keďže tieto možnosti nie sú disjunktné, nech si vyberie ktorýkoľvek prvok na pokrytie, nakoniec zistí, že v tejto vetve sa riešenie nenájde, keďže nebude možné pokryť všetky prvky.

Algoritmus sa pomocou X6, X7 a X8 vráti na úroveň 0, čím odrobí všetky doteraz spravené zmeny na dátovej štruktúre, a do x_0 sa teraz vyberie možnosť ’A, D’. V kroku X5 sa pokryje aj prvok D a ďalej sa vstúpi na úroveň 1. Zostanú 3 možnosti: ’C, E, F’, ’B, C, F’ a ’B, G’.

Na druhej úrovní budeme predpokladať, že sa algoritmus rozhodne pokryť prvok C (opäť nezáleží na tom, že to bude práve C, mohol by to byť aj ktorýkoľvek iný z ostávajúcich nepokrytých prvkov), vyberie teda jedinú možnosť ’C, E, F’ prislúchajúcu k prvku C a pokryje aj E a F. Ostane teda úž iba jediná nepokrytá možnosť ’B, G’ a iba 2 nepokryté prvky: B a G.

Na tretej úrovni sa algoritmus pokúsi pokryť jeden z týchto dvoch prvkov a nezostane už žiaden nepokrytí prvok. Na začiatku ďalšej úrovne algoritmus zhodnotí, že bolo nájdené riešenie pozostávajúce z možností:

A D

C E F

B G

Vidíme, že všetko sú to disjunktné možnosti a ich zjednotenie je množina A až G, teda je to nepochybne riešenie. Toto je zároveň aj jediné riešenie tohto problému.

1.8 S heuristika

Jednoduchšia verzia Algoritmu X [2, s. 6] v kroku X3 vyberie prvok i, ktorý treba pokryť jednoducho:

i = RLINK(header)

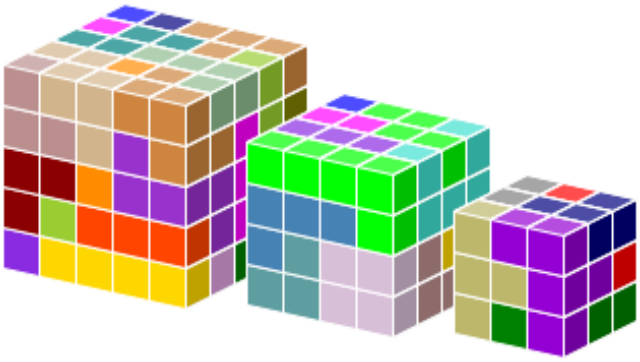
Toto riešenie funguje dobre, pretože treba pokryť vždy všetky prvky a je aj veľmi jednoduché na naprogramovanie.

Knuth ale navrhuje [2, s. 4] použitie inej technológie výberu prvku na pokrytie. Konkrétne spomína, že algoritmus si vyberie vždy prvok, ktorý sa vyskytuje v najmenšom počte možností. Tento spôsob výberu nazýva S heuristika. V kroku X3 algoritmu X treba teda najprv zistiť, ktorý z nepokrytých prvkov ma najmenej možností. Na tento trik efektívne slúži atribút LEN(X), ktorý pripadá ku každému vrcholu X predstavujúcemu prvok množiny S, keďže Algoritmus X tak ako sme ho napísali túto hodnotu neustále mení tak, aby zodpovedala súčasnému stavu spájaného zoznamu.

Knuth ďalej pojednáva [2, s. 9, 10] o efektivite týchto dvoch spôsobov výberu prvku. Uvádza, že pri menších problémoch sa ukazuje lepší prvý postup, zatiaľ čo pri väčších ten druhý. Skúšal použiť aj kombináciu týchto dvoch prístupov, ale ukázalo sa, že je všeobecne lepšie zachovať S heuristiku vždy. Z tohto dôvodu budeme aj v tomto riešení používať vždy S heuristiku.

1.9 Redukcia riešenia 3D drevených hlavolamov do Exact cover problému.

Na 3D hlavolamy sa dá pozerať ako na takzvané polykocky (polycubes), útvary zložené z kociek o rozmere 1x1x1, spojených tvárou v tvár. Takýto hlavolam je zložený z viacerých menších polykociek rôznych tvarov (ako je vyobrazené na obrázku 9). Na každý z týchto útvarov z ktorých je zložený hlavolam sa dá pozerať ako na jeho podmnožinu. Jednotlivé dieliky sú disjunktné a ich zjednotením je celý hlavolam. Z tohto je už zrejme ako sa drevené hlavolamy redukujú na Exact cover problém.



Obr. 9: 3D hlavolamy

Ak by sme chceli nejaký ľubovoľný 3D hlavolam pretransformovať aby bol riešiteľný Algoritmom X, tak jednotlivé 1x1x1 kocky z ktorých je hlavolam tvorený by sa stali prvkami (items), ktoré bude algoritmus pokrývať.

Zistenie všetkých možností, ktorými sa dajú prvky pokryť je viacfázový proces. Majme dostupné tvary, ktoré sú podmnožiny hlavolamu, ktorými sa bude hlavolam pokrývať, všetky možnosti pokrytia prvkov [1, s. 250] sa získajú nasledovne:

- Pre každý dielik zostrojíme všetky jeho rotácie a symetrie tým, že ho otáčame okolo osy x, y a z. V 3D priestore ich je presne 24, pričom v závislosti od tvaru špecifického dieliku, môžu byť niektoré z týchto rotácii totožné, teda celkový počet bude trošku menší.
- Pre každú rotáciu každého dielika vyskúšame akými všetkými spôsobmi sa dá umiestniť do hlavolamu. Toto sa dosiahne prechodom dieliku cez x, y a z súradnice hlavolamu. Každé takéto posunutie rotácie dieliku po hlavolame,

kde žiadna kocka dieliku nepretŕča z hlavolamu v žiadnej osy je považovaná za možnosť pokrytia hlavolamu.

Po vykonaní týchto operácii máme špecifikované aj prvky aj možnosti a teda sa dá spustiť Algoritmus X, ktorý nájde všetky spôsoby, ktorými sa pomocou jednotlivých dielikov dá hlavolam pokryť.

1.10 Podobné práce

Práca Solving Sudoku efficiently with Dancing Links [3] hľadala riešenia hry sudoku pomocou Algoritmu X a technológie Dancing links. Vychádza z teoretických podkladov od Donalda Knutha a je postavená hlavne na zistení výhodnosti použitia Dancing links pri riešení Exact cover problémov a hľadaní efektívnej redukcie Sudoku do Exact cover problému. Taktiež motivuje softvérových vývojárov aby hľadali už existujúce algoritmy na riešenie problémov, ako je v ich prípade Algoritmus X, namiesto Brute force riešení problémov.

Použitá literatúra

[1] Donald E. Knuth. *The Art of Computer Programming, Volume 4*. Addison-Wesley, 1968.

[2] Donald E. Knuth. *Dancing Links*. In Millenian Perspectives in Computer Science, strany 187-214, 2000.

[3] Mattias Harrysson, Hjalmar Laestander. *Solving Sudoku efficiently with Dancing Links.* Stockholm, 2014.

[4] Éva Tardos, Jon Kleinberg. *Algorithm Design*. Addison-Wesley, 2005.

[5] Andrej Blaho. *Spájané zoznamy:* http://input.sk/python2016/27.html [Online, 8.2.2021]