Report – ročníkový projekt
Rekonfiguračný graf stable-tree dekompozícii grafov

Vincent Hlaváč

Ciele:

1. Implementovať program, ktorý pre zadaný kubický graf G s 4k+2 vrcholmi vytvorí rekonfiguračný graf jeho stable-tree dekompozícii
2. Oboznámiť sa so stable-tree dekompozíciami kubických grafov
3. Potenciálne rozšíriť program o testovanie vlastností výsledného grafu, optimalizovať zložitosť programu

Na implementáciu riešenia problému som sa rozhodol použiť programovací jazyk java.

Počas zimného semestra sa podarilo:

1. Zoznámiť so stable-tree dokompozíciami 4k+2-vrcholových kubických grafov.

2. Naprogramovať reprezentáciu neorientovaného grafu pomocou adjacency list (listu susednosti) – AdjacencyListGraph.java. Možnosť s ním pracovať pomocou univerzálneho grafového rozhrania – Graph.java.

3. Naprogramovať reader, ktorý zo súboru prečíta graf a vráti jeho objektovú reprezentáciu – AdjacencyListGraphReader.java. Možnosť s ním pracovať pomocou univerzálneho rozhrania pre grafové readere – GraphReader.java. V konštruktore mu možno špecifikovať kontrolu – GraphTypeControl.java, ktorú urobí na prečítanom grafe, ktorý vracia, len ak ju spĺňa. Implementoval som kontrolu pre 4k+2-vrcholový kubický graf – Cubic4KPlus2Control.java. Podarilo sa otestovať korektnosť readera.

4. Naprogramovať algoritmus generujúci všetky stable-tree dekompozície daného 4k+2-vrcholového kubického grafu – StableTreeDecomposition.java. Podarilo sa otestovať jeho korektnosť na základe porovnaní s očakávanými výstupmi (počty dekompozícii a počty výskytov jednotlivých vrcholov v ich stable setoch). Pre implementáciu algoritmu som naprogramoval repezentáciu komponentu grafu – Component.java.

5. Naprogramovať generátor konkrétnych grafov, ktorý vytvorý rozsiahlejšie grafy pre testovanie pri akom veľkom vstupe narazíme na exponenciálnu zložitosť daného backtrackového algoritmu. Konkrétne generujeme 4k+2-vrcholové kružnice s hranami medzi protiľahlými vrcholmi – Circle4KPlus2Generator.java a 2k+1-boké hranoly – Prism2KPlus1SidedGenerator.java. Možnosť s nimi pracovať pomocou rozhrania pre grafové generátory – GraphGenerator.java.

Počas letného semestra sa podarilo:

1. Naprogramovať reprezentáciu rekonfiguračného grafu – AdjListReconfGraph.java. Možnosť s ním pracovať pomocou univerzálneho rozhrania pre rekonfiguračné grafy – ReconfGraph.java. Rozširuje triedu AdjacencyListGraph a je možnosť v ňom využiť rôzne typy vrcholov, ktoré implementujú univerzálne rozhranie na prácu s vrcholmi – Vertex.java. Pre naše potreby som naprogramoval vrchol reprezentujúci stable-tree dekompozíciu – StableTreeDecompVertex.java.

2. Naprogramovať algoritmus, ktorý z vygenerovaných stable-tree dekompozícii vytvorí rekonfiguračný graf. Podarilo sa otestovať jeho korektnosť na základe porovnaní s očakávanými výstupmi.

3. Naprogramovať algoritmy pre zistenie niektorých vlastností grafov (počet komponentov, bipartitnosť) v podobe statických metód poskytovaných triedou pre grafové algoritmy – GraphAlgorithms.java. Tie umožnia skúmať vytvorený rekonfiguračný graf.

4. Otestovať časovú zložitosť algoritmu na generovanie stable-tree dekompozícii.

Všetky triedy sú súčasťou balíka graphs, ktorý tvorí akoby knižnicu jazyka java a možno ich používať vo vlastnom programe. Graph je možné vytvoriť len pomocou GraphReader, aby bolo zaručené, že vytvorený graf je korektný.

Metódy poskytované StableTreeDecomposition:

1. public List<List<Integer>> getDecomposition() - vracia vygenerované stable-tree dekompozície v podobe List<List<Integer>>, kde integre reprezentujú vrcholy zo stable setu dekompozície.

2. public ReconfGraph getReconfigurationGraph() - vracia rekonfiguračný graf vygenerovaných stable-tree dekompozícii v podobe ReconfGraph. Daný graf generuje pri prvom volaní metódy.

Ďalej som naprogramoval samostatné programy na testovanie a skúšobné použitie tried balíka graphs:

1. GraphGenerationTest – generuje súbory s grafmi na základe vstupných parametrov.

2. STDGenerationTest – načíta zo súboru určeného vo vstupných parametroch graf a vygeneruje jeho stable-tree dekompozície, ktoré uloží do súboru v provizórnom formáte výpisu popisujúcom vrcholy stable setu.

3. RGPropertiesTest – načíta zo súboru určeného vo vstupných parametroch graf, vygeneruje jeho stable-tree dekompozície a z nich rekonfiguračný graf, ktorý otestuje na súvislosť a bipartitnosť. Výsledky uloží do súboru vrátane indexov vrcholov v jednotlivých komponentoch, ak nie je súvislý, respektíve v jednotlivých častiach (parts), ak je bipartitný.

   *Indexy vrcholov vo výstupe z RGPropertiesTest sa zhodujú s usporiadaním stable-tree dekompozícii vo výstupe z STDGenerationTest.

   *Príloha example.7z obsahuje výstupy týchto programov pri testovaní.

   *Popis vstupných parametrov je v komentári v kóde.

Súborová reprezentácia grafu:

Na prvom riadku je číslo n reprezentujúce počet vrcholov grafu. Ďalších n riadkov prislúcha postupne všetkým vrcholom s indexami od 0 po n-1. Na riadku prislúchajúcemu vrcholu s indexom i sú indexy tých vrcholov do ktorých ide z i hrana a zároveň sú vačšie než i. Takto popisujeme každú hranu len raz. Súbor nesmie obsahovať žiadne prebytočné neblankové znaky.

Výsledky testovania generovania stable-tree dekompozícii niektorých grafov:

1. 2k+1-boké hranoly – generované pomocou Prism2KPlus1SidedGenerator.java
   a) k=4 (18 vrcholov) – 144 stable-tree dekompozícii, čas 18ms
   b) k=5 (22 vrcholov) – 352 stable-tree dekompozícii, čas 38ms
   c) k=10 (42 vrcholov) – 21504 stable-tree dekompozícii, čas 1987ms
   d) k=15 (62 vrcholov) – 1015808 stable-tree dekompozícii, čas 267473ms cca 4,5min
2. 4k+2-vrcholové kružnice – generované pomocou Circle4KPlus2Generator.java
   a) k=4 (18 vrcholov) – 144 stable-tree dekompozícii, čas 19ms
   b) k=5 (22 vrcholov) – 352 stable-tree dekompozícii, čas 38ms
   c) k=10 (42 vrcholov) – 21504 stable-tree dekompozícii, čas 2143ms
   d) k=15 (62 vrcholov) – 1015808 stable-tree dekompozícii, čas 272590ms cca 4,6min

      *Súbory s grafmi aj výslednými stable-tree dekompozíciami sú v prílohe example.7z

Výsledky testovania generovania a vlastností rekonfiguračných grafov pre niektoré grafy:

1. 2k+1-boké hranoly – generované pomocou Prism2KPlus1SidedGenerator.java
   a) k=4 (18 vrcholov) – 144 vrcholov, čas 18ms, súvislý, bipartitný
   b) k=5 (22 vrcholov) – 352 vrcholov, čas 37ms, súvislý, bipartitný
   c) k=10 (42 vrcholov) – 21504 vrcholov, čas 9754ms, súvislý, bipartitný
2. 4k+2-vrcholové kružnice – generované pomocou Circle4KPlus2Generator.java
   a) k=1 (6 vrcholov) – 6 vrcholov, čas 2ms, nesúvislý, 2 komponenty, nie je bipartitný
   b) k=4 (18 vrcholov) – 144 vrcholov, čas 14ms, súvislý, nie je bipartitný
   c) k=5 (22 vrcholov) – 352 vrcholov, čas 41ms, súvislý, nie je bipartitný
   d) k=10 (42 vrcholov) – 21504 vrcholov, čas 10230ms, súvislý, nie je bipartitný

   *Súbory s výsledkami testovania vlastností rekonfiguračných grafov sú v prílohe example.7z