УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Петар П. Петровић

МАСТЕР РАД ИЗ МАТЕМАТИКЕ ИЛИ РАЧУНАРСТВА ЧИЈИ ЈЕ НАСЛОВ ЈАКО ДУГАЧАК

мастер рад

Београд, 2016.

Ментор:

др Мика Микић, редован професор
Универзитет у Београду, Математички факултет

Чланови комисије:

др Ана Анић, ванредни професор
University of Disneyland, Недођија

др Лаза Лазић, доцент
Универзитет у Београду, Математички факултет

Датум одбране: 15. јануар 2016.

Мами, тати и деди

Наслов мастер рада: Мастер рад из математике или рачунарства чији је наслов јако дугачак

Резиме: Klasa np problema je jedna veliki skup problema u programiranju. Mnoge stvari se izucavaju na np kompletnim problemima, pa je tako tema ovog master rada nastala. Prvobitna ideja je rada je da se automatizuje proces svodjenja jednog problema na drugi, ali kako je to zahtevalo mnogo posla.

Rad se svodi na transforamciju nekih problema na SAT problem u jeziku URSA

prvo poglavlje govori sta je jezik ursa, postavlja teoriju np koplenosti. Drugo poglavlje se koncentirse na implementaciju samih algoritama prevodjenja. U zakljucku je obradjeno do cega smo dosli kao i moguca dalja istrazivanja

Кључне речи: НП-Комплетност, Теорија израчунљивости, језик УРСА, , Недетерминизам, Сертификати

Садржај

1 Увод **1**

1.1 Примери коришћења класичних LaTeX елемената 1

2 Разрада **3**

2.1 НП-Комплетност 3

2.2 Свођење НП-комплетних проблема 6

3 Закључак **8**

Библиографија **9**

Глава 1

Увод

ovo je drugi panagram

1.1 Примери коришћења класичних LaTeX елемената

Ово је реченица у којој се јавља цитат [3]. Још један цитат [2]. Испробавамо наводнике: „Рекао је да му се јавимо сутра”. У табели 1.1 која следи приказани су резултати експеримента. Ovo je primer rečenice ispisane latiničkim pismom u okviru ćiriličkog dokumenta. У овој реченици се налази једна реч написана латиницом. Иза ове реченице следи фуснота.[1] Сајт математичког факултета је `http://www.matf.bg.ac.rs`.

Ovo je malo duži blok teksta ispisan latiničkim pismom u okviru ćiriličkog dokumenta. Fijuče vetar u šiblju, ledi pasaže i kuće iza njih i gunđa u odžacima.

Ево и један пример математичке формуле: $e^{i\pi}+1 = 0$. На слици 1.1 приказан је један графикон.

[1] Ово је фуснота.

Табела 1.1: Резултати

1	2	3
4	5	6
7	8	8

Слика 1.1: Графикон

$$\int_a^b f(x)\ \mathrm{d}x\ =_{def}\ \lim_{\max \Delta x_k \to 0} \sum_{k=1}^{n} f(x_k^*)\Delta x_k$$

Више детаља биће дато у глави **??** на страни **??**.

У тезу можемо убацити и програмски кôд.

```
Ovo je doslovni tekst.
```

Овај C програм се може превести помоћу преводиоца GCC [1].

Можемо правити и набрајања:

1. Анализа 1
2. Линеарна алгебра
3. Аналитичка геометрија
4. Основи програмирања

Treci panagram

Глава 2

Разрада

2.1 НП-Комплетност

Постоје проблеми за које не потоји елегантно решење. Можда није уложено довољно напора да се оно нађе, али се са доста сигурности може претпоставит да постоје проблеми који немају ефиксно решење.

За алгоритме који се могу решити у полиномијалном времену (O(P(n)) кажемо да су ефикасни. Класу свих проблема који се могу решити ефикасним алгоритмом означавамо са П (полиномијално време). Једна потклаcа проблема који нису П су НП-комплетни проблеми. Ти проблеми припадају истој класи јер су међусобно строго еквивалентини => ефикасан алгоритам за неки НП-комплетан проблем постоји акко за сваки НП-Комплетан проблем постоји ефикасан алгоритам. За сада, такав алгоритам није пронађен, и верује се да не постоји. Овакивих проблема има на стотине, можда и хиљаде, због чега је ова област занимљива.

Битан концепт за НП-Комплетност је редукција - свођење једног проблема на други. Да би се лакше дефинисали, уводимо ограничење посматрања проблема одлучивања. Ако умемо да решимо проблем одлучивања, обично можемо да решимо и полазни проблем.

Дефиниција: Нека су L1 и L2 два језика, подскупова редом скупова улаза U1 и U2. Кажемо да је Л1 полиномијално сводљив на Л2, ако постоји алгоритам полиномијалне врменске сложености, који дати улаз у1 (припада) У1 преводи у улаз у2 (припада) У2, тако да у1 (припада)Л1 акко у2 (припада)Л2.

Ако знамо алгоритам за препознавање Л2, онда га можемо суперпонирати са алгоритмом редукције и тако добити алгоритам за решавање Л1. Алгоритам

редукције АР, алгоритам препознавања Л2 је АЛ2. Препознавање улаза у1 (припада)У1 може се применом АР трансформисати у улаз у2 (припада) У2, и применом АЛ2 установити да ли у2 (прпада) Л2, а тиме и да ли у1 (припада)Л1.

Теорема: Ако је језик Л1 полиномијално сводљив на језик Л2, и постоји алгоритам полиномијалне временске сложености за препознавање Л2, онда постоји алгоритам полиномијалне временске сложености за препознавање Л1

Доказ: Релација полиномијалне сводљивости није полиномијална сводљивост Л1 на Л2 не повлачи полиномијалну сводљивост Л2 на Л1. Ово произилази из чињенице да дефиниција сводљивости захтева да се произвољан улаз за Л1 може трнсформисати на Л2, али не и обрнуто. Могуће је да улаз за Л2, добијен на овај начин, представља само мали део свих могућих улаза. Ако је Л1 полиномијално сводљив на Л2, сматрамо да је Л2 **тежи**

Два језика Л1 и Л2 су полиномијално еквивалентна ако је сваки од њих полиномијално сводљив на други. Релација полиномијалне сводљивости је транзитивна.

Теорија НП-Комплетности је део шире области која се зове сложеност израчунавања.

Формална дефиниција класе језика П: П = Л | постоји полиномијални програм М за ДТМ, за који је Лм = Л

Један од најпознатијих НП-Комплетних проблема је проблем трговачког путника. „НАВЕСТИ ПРОБЛЕМ" - МОЖЕ ДА БУДЕ ЈЕДАН ЗА СВОЂЕЊЕ КАСНИЈЕ. За овај проблем нема познатог алгоритма полиномијалне сложености, али ако имамо неки улаз (низ чворова), лако можемо да проверимо да ли је то Хамилтонов циклус и да израчунамо његову дужину => временска сложеност полиномоијална = полиномијална проверљивост.

Провера за полиномијално време није исто што и решавање за поликомијслно време.

НП се неформално дефинише помоћу недетерминистичког алгоритма. Тај алг се састоји од фазе погађања и фазе провере.

Sa дати улаз у у првој фази се изводи просто погађање структуре С. Затим се у и С заједно предају као улаз за фазу провере, која се изводи на детерминистички начин, па се зевршава одговором да, не или се извршава бесконачно дуго. Недетерминистички алгоритам „решава" проблем одлучивања П, ако су за произвољни улаз у (припада) Уп за овај проблем испуњена два услова: - Ако у (припада) Лп онда постоји таква структура С, чије би погађање за улаз

у довело до тога да се фаза провере са улазом (у, С) заврши одговором „да". - Ако у (не припада) Лп, онда не постоји таква структура С, чије би погађање за улаз у обезбедило завршавање фазе провере са улазом (у, С) одговорм „да".

Каже се да недетерминистички алгоритам који решава проблем одлучивања П ради за „полиномијално време", ако постоји полином п такав да за сваки улаз у (припада) Уп постоји такво погађање С, да се фаза провере са улазом (у, С) завршава са одговором „да" за време п(|у|).

Класа НП, дефинисана неформално, то је класа свих проблема одлучивања који при разумном кодирању могу бити решени недетирмистичким (Н) алгоритмом за полиномијално(П) време. Формално: НП)= Л | постоји НДТМ програм М са полиномијалним временом извршавања, за који је Лм = Л Постоје проблеми који се не могу решити ефикасно ни помоћу недетерминистичког алгоритма. Класа проблема за које постоји недетерминистички алгоритам полиномијалне временске сложености зове се НП.

Проблем односа П и НП је познат као проблем П=НП. **Дефиниција** Проблем Х је НП-тежак проблем ако је сваки проблем из класе НП полиномијално сводљив на Х. **Дефиниција** Проблем Х је НП-комплетан проблем ако припада класи НП и ако је НП-тежак.

Проблем Х је НП-комплетан ако (1) Х припада класи НП, и (2) постоји НП-комплетан проблем Y који је полиномијално сводљив на Х

Проблем задовољивости (CAT)

Посебно ћемо описати прблем задовољивости израза јер је он први који за који је доказано да је НП-Комплетан. Кукова теорема. (ставити цитат ка теореми, линк...)

Нека је С Булов израз у конјуктивној нормалној форм (КНФ). С је конјукција више клауза - дисјункција група литерала, симбола променљивих или њихових негација. С = (x+y+|z)(|x+y+z)(|x+|y+|z) где је сабирање дисјункција(или), а множење конјукција (и), а свака променљива има вредност 0 или 1. Свака булова функција се може представити изразом у КНФ. Булов израз је задовољив, ако постоји такво додељивање нула и јединица променљивим, да израз има вредност 1. Проблем САТ се састоји у утврђивању да ли је задати израз задовољив - није неопходно пронаћи одговарајуће вредности.

Проблем САТ је у класи НП јер се за (недетерминистички) изабране вредности променљивих за полиномијално време (од величине улаза - укупне дужине

формуле) може проверити да ли је израз тачан. Проблем САТ је НП-тежак јер се извршавање програма НДТМ за алгоритам који решава произвољан изабрани проблем из класе НП може описати Буловим изразом.

URSA језик

Свођење алгоритама ћемо радити у језику урса. То је језик који је настао на Математичком факултету за сврху свођења проблема на проблем САТ(можда је и САТ солвер, прочитати изучити). (Линк ка раду и сајту)

2.2 Свођење НП-комплетних проблема

Да би се показало да је неки проблем НП-Комплетан, морамо да покажемо да припада и класи НП и класи НП-тешких проблема.

Проблем припада НП-класи ако може да се у полиномијалном времену провери тачност, односно за дати сертификат, можемо у полиномијалном времену проверимо да ли је то решење проблема.

Да бисмо доказали да неки проблем припада класи НП-тешких проблема, потребно је да неки познати НП-тежак проблем сведемо на овај проблем у полиномијалном времену.

Узајамно свођење проблема САТ и Клике

Проблем клике: клика је подграф графа такав да сви чворови тог подграфа су међусобно повезани, што значи да је тај подграф заправо комплетан граф. Проблем проналаска максималне клике је проблем налажења таквог подграфа са највећим могућим бројем чворова. Наш проблем одлучивања ће бити проналазак клике величине К.

Да би се показало да клика припада класи НП проблема задајемо сертификат: скуп од С чворова који се састоје у клики и С је подграф графа Г. Да би смо проверили да ли је С величине К треба нам време О(1). Затим треба још да проверимо да ли су сви чворови међусобно повезани за шта нам треба јос О(к*к) времена. Тако да провера да ли к чворова из С су комплетни захтева време од О(н*н) где је н број чворова графа.

Да бисмо показали да је клика НП-тежак проблем, свешћемо проблем САТ на њу. То радимо тако што кренемо од неке инстанце сат проблема, и кажемо

да ако је CAT формула задовољива, онда ће и у добијеном графу постојати клика бар величине броју клаузула

За други смер свођења имамо 4 ограничења: -свака позиција у клики мора бити попуњена -Један чвор не може бити на више од једне позиције у клики -свака позиција мора бити попуњена различитим чворовима -Чворови морају бити међусобно повезани

овим поступком добијамо доста клаутула и више решења због пермутација. Тачан број је дат формулом: број клика у графу * faktorijal величина клике

Свођење проблема Клике на проблем Покривача грана

Hamiltonov ciklus <=> TSP: Problem Hamiltonovog ciklusa se može svesti na TSP tako što se težine grana postave na 1 za postojeće veze i na veliku vrednost za nepostojeće veze. Ako postoji TSP rešenje sa težinom jednakom broju čvorova, postoji i Hamiltonov ciklus.

TSP <=> Problem raspoređivanja: TSP se može svesti na problem raspoređivanja definisanjem zadataka kao poseta čvorovima i resursa kao puteva između čvorova, gde težine grana predstavljaju troškove ili vremena putovanja.

Problem raspoređivanja <=> SAT: Problem raspoređivanja se može svesti na SAT tako što se za svaki zadatak definišu promenljive koje predstavljaju raspored resursa i vremenskih okvira, a ograničenja se predstavljaju kao klauze.

Klika <=> Maximum Independent Set <=> SAT: Maksimalni nezavisni skup se može svesti na SAT tako što se svaka podskupina čvorova predstavlja promenljivama, a ograničenja međusobne nepovezanosti se predstavljaju kao klauze.

Глава 3

Закључак

sesti panagram
sedmi panagram

Библиографија

[1] Free Software Foundation. GNU gcc, 2013. on-line at: `http://gcc.gnu.org/`.

[2] Yuri Gurevich and Saharon Shelah. Expected computation time for Hamiltonian path problem. *SIAM Journal on Computing*, 16:486–502, 1987.

[3] Petar Petrović and Mika Mikić. Naučni rad. In Miloje Milojević, editor, *Konferencija iz matematike i računarstva*, 2015.

Биографија аутора

Вук Стефановић Караџић (*Тршић, 26. октобар/6. новембар 1787. — Беч, 7. фебруар 1864.*) био је српски филолог, реформатор српског језика, сакупљач народних умотворина и писац првог речника српског језика. Вук је најзначајнија личност српске књижевности прве половине XIX века. Стекао је и неколико почасних доктората. Учествовао је у Првом српском устанку као писар и чиновник у Неготинској крајини, а након слома устанка преселио се у Беч, 1813. године. Ту је упознао Јернеја Копитара, цензора словенских књига, на чији је подстицај кренуо у прикупљање српских народних песама, реформу ћирилице и борбу за увођење народног језика у српску књижевност. Вуковим реформама у српски језик је уведен фонетски правопис, а српски језик је потиснуо славеносрпски језик који је у то време био језик образованих људи. Тако се као најважније године Вукове реформе истичу 1818., 1836., 1839., 1847. и 1852.