Poročilo

Ime: Tamara Pogačar

Datum: 29. 1. 2021

O seminarski nalogi

V moji seminarski nalogi si ogledamo nekaj algoritmov v labirintih. Seminarska naloga se začne s predsatavitvijo smiselne `delitve labirintov`, čemur sledi predstavitev enega izmed `načinov zapisa labirinta v obliki grafa` (tj. kako pridemo iz 2D labirinta do pripadajočega grafa). Za boljše razumevanje si algoritem ogledamo na dveh primerih - en primer za labirint brez krožnih poti, drug pa za labirint s krožnimi potmi. V nadaljevanju pobližje spoznamo načine `zapisa grafa` v računalniku predstavljivem zapisu - natančneje si ogledamo enega izmed njih in sicer tabelo sosednosti.

Po kratki pedstavitvi labirintov sledi ogled konkretnih algoritmov. Najprej si ogledamo `pregled v širino` in `pregled v globino`. Ugotovimo, da sta si algoritma nadvse podobna, saj se razlikujeta zgolj v uporabljeni podatkovni strukturi za shranjevanje vozlišč, ki jih srečamo tekom samega pregleda. Za lažjo predstavo uporabe obravnavanih algritmov si ogledamo `aplikacijo preiskovalnih algoritmov na konkretnih labirintih`. Na tem mestu si za boljše razumevanje prenosa algoritma iz grafa na konkreten labirint ogledamo vse korake postopeka. V nadaljevanju si pri posameznih primerih ogledamo samo še "pomembne" dele, kjer pride do pomembnih sprememb in pripadajočih razmislekov.

Ogledamo si algoritem za `iskanje najkrajše poti v labirintu`. Najprej si ogledamo algoritem za iskanje `najmanjšega števila hodnikov`, ki jih moramo prehoditi od vhoda do izhoda. Nato pa si ogledamo še algoritem za iskanje `najkrajše poti`, kjer je najkrajka pot tista, z najmanj prehojenimi dolžinskimi enotami. Ker si pri kompleksnejših labirintih ne želimo iskati najkrajšo pot v konkretnem labirintu, si ogledamo, kako dan labirint ustrezno predstavimo z oteženim grafom. Nato si ogledamo delovanje algoritma na pripadajočem grafu, pri čemer opazimo, da lahko na podoben način dobilmo tudi najdaljšo pot v labirintu.

O kratkem vprašanju

Ogledamo si algoritme za ugotavljanje ujemanja nekega nadomestnega vzorca z danim nizom. Na začetku si za motivacijo ogledamo nekaj primerov ujemanj in neujemanj nizov, ki nas seznanijo z danim problemom in dajo občutek, kako se želimo lotiti danega problema.

Misli strnemo v poglavju `Kako se lotimo problema?`, kateremu sledita primera reševanja. Tu opazimo, da bi za sam algoritem potrebovali štiri kazalce, katerih delovanje predstavimo v poglavju `Kazalci`. Seveda tudi temu poglavju sledijo primeri za boljše razumevanje obravnavanega. Tekom teh primerov ugotovimo, da se nekata primerjanja ponavljajo. Opazimo, da vselej ugotavljamo ujemanje podnizov, katerih začetek se "premika" od začetkov danih nizov proti njunima koncema. Torej dobimo rešitev celotnega problema ujemanja iz rešitev ujemanj ustreznih (manjših) podnizov. Zato vemo, da se postopka ugotavljanja enakosti lahko lotimo s pomočjo dinamičnega progrmiranja.

Problem predstavimo z `matriko`, kjer hitro opazimo, da nekateri elementi ne vplivajo na končni rezultat ujemanja danih nizov. Zato si ogledamo algoritem z uporabo `tabele tabel`, ki je optimalnejša. Nato si

ogledamo še primerjavo matrike s tabelo tabel, ali se dana niza lahko ujemata na več načinov, zakaj gre pri tem problemu za dinamično programiranje ter za konec še časovno in prostorsko zahtevnost za tiste algoritme, katerih zahtevnosti so najmanjše.

Težave in izzivi

Na prve težave sem naletela, ko v predlagani literaturi podatki niso bili pravilni (časovna zahtevnost algoritma za določanje ujemanja nadomestnega vzorca). Precej zahtevno je bilo tudi raziskovati področja, ki jih pred tem še nismo uspeli obravnavati v sklopu predavanj. Ugotovila sem, da bi proti koncu šolskega leta lahko naredila primernejše predstavitve, saj bi lahko bolje ocenila znanje poslušalcev, njihovo znanje pa bi bilo občutno večje kot v danem trenutku. Večina mojih problemov je namreč izvirala iz dejstva, da smo snov mojih predstavitev pričeli obravnavati zgolj kakšen teden pred samo predstavitvijo, večino pa smo žal zaključili po predstavitvi. Na primer sama seminarska naloga je sprva temeljila predvsem na obravnavi pregleda v globino in širino. Kakšen teden pred samo prestavitvijo, smo na najino žalost to snov obravnavali tekom predavanj. Zato sva želela predstavitev prilagoditi, a zaradi pomanjkanja časa večje spremembe niso bile mogoče. Torej sem imela ogromno zadnjih popravkov, ki pa so se žal izkazali za ne najboljše, saj v tako kratkem času žal izdelek ni mogel biti dodelan.

Kaj sem se naučila?

Naučila sem se, da ni dovolj navesti vir črpanja napačnih informacij. Potrebno je raziskati področje in najti pravilno informacijo. Poleg tega sem se naučila tudi, da se je zelo pomembno zavedati, komu se predava, da se lahko temu ustrezno prilagodi samo predstavitev. Predvsem pa je zelo pomembna pametna izbira para. Naučila sem se tudi, da včasih podrobna predstavitev postopka ni tako zelo pomembna kot poudarjanje (in pomoče ponovitev) pomembnih delov za razumevanje celote.

Kako sva z Matejem delovala v paru?

Menim, da pretekli semester delo v paru ni bilo najboljše. Veliko časa sva z Matejem namreč potrebovala z "razumevanje" drug drugega. Sprva sem večino gradiv zbrala sama, nato sva skupaj določila konkretne teme za predstavitev, ki so se nama zdele poučne za najine kolege. Potlej sem spisala večji del predstavitve, zgolj zaključek (iskanje najkrajše poti) je bil še nedodelan. Nato sva se dogovorila, da Matej sestavi prosojnice in si dodelila dele, ki sva jih nato predstavila med samo predstavitvijo. Potem sva vsak nad svojim delom izvedla zadnje popravke (Matej se je na tem mestu odločil spremeniti zadnji del), imela sva generalko (kjer sva ocenila potrebovani čas za predstavitev) in na koncu svojo seminarsko nalogo predstavila kolegom.

Kako je "zloženo" gradivo

Na mojem GitHub repozitoriju **`OddajaSN_R1`** se poleg datotek `LICENSE` in `README.md` nahajajo štiri mape:

1. `gradiva_predstavitve`, ki vsebuje:
    - datoteko `prosojnice_kratko_vprasanje.html`. To je datoteka s prosojnicami, uporabljenimi na predstavitvi mojega kratkega vprašanja.
    - datoteko `prosojnice_kratko_vprasanje.md`, ki je izvorna datoteka datoteke uporabljene pri sami predstavitvi.

    - datoteko `prosojnice_SN.pptx`, ki vsebuje prosojnice uporabljene pri predstavitvi seminarske naloge.
2. `kratko_vprašanje`, ki vsebuje:
    - sliko `index.jpg`,
    - sliko `niza.jpg`,
    - datoteko `opis.md`, ki je izvorna datoteka datoteke `opis.pdf`,
    - datoteko `opis.pdf`, ki vsebuje podroben opis vsebine kratkega vprašanja.
3. `porocilo`, ki vsebuje:
    - sliko `niza.jpg`,
    - datoteko `porocilo.md`, ki je izvorna datoteka datoteke `porocilo.pdf`.
    - datoteko `porocilo.pdf`, ki vsebuje poročilo o moji seminarski nalogi (tj. seminarska naloga in kratko vprašanje).
4. `SN`, ki vsebuje:
    - mapo `dodatne_slike`, ki vsebuje nekaj slik uporabljenjih pri seminarski nalogi,
    - mapo `slike`, ki vsebuje preostanek slik uporabljenih pri seminarski nalogi,
    - datoteko `alogoritmi_v_labirintih.md`, ki je izvorna datoteka datoteke `alogoritmi_v_labirintih.pdf`,
    - datoteko `alogoritmi_v_labirintih.pdf`, ki vsebuje podroben opis same seminarske naloge.

Vsi viri

Kratko vprašanje:

Dynamic Programming | Wildcard Pattern Matching | Linear Time and Constant Space (1. 11. 2020) Pridobljeno s https://www.geeksforgeeks.org/dynamic-programming-wildcard-pattern-matching-linear-time-constant-space/

Dynamic Programming | Wildcard Pattern Matching | Linear Time and Constant Space (15. 12. 2020) Pridobljeno s https://www.geeksforgeeks.org/dynamic-programming-wildcard-pattern-matching-linear-time-constant-space/

Coding Interview Question | Wildcard Matching | Dynamic Programming with Optimization (7. 11. 2020) Pridobljeno s https://youtu.be/7SHV_QfVROE

Wildcard Pattern Matching (Dynamic Programming) (15. 12. 2020) Pridobljeno s https://iq.opengenus.org/wildcard-pattern-matching-dp/

Wildcard Pattern Matching (15. 12. 2020) Pridobljeno s https://prabhusiddarth.wordpress.com/2018/03/25/wildcard-pattern-matching/

Dinamično programiranje (15. 12. 2020) Pridobljeno s https://sl.wikipedia.org/wiki/Dinami%C4%8Dno_programiranje

Seminarska naloga:

Depth First Search Algorithm In Python (Multiple Examples) (8. 11. 2020) Pridobljeno s: https://likegeeks.com/depth-first-search-in-

python/#:~:text=%20Depth%20First%20Search%20algorithm%20in%20Python%20%28Multiple,to%20implement%20the%20DFS%20algorithm%20in...%20More%20

Shortest path through a maze (14. 11. 2020) Pridobljeno s: https://aakritty.wordpress.com/2014/03/02/shortest-path-through-a-maze/

Maze (10. 11. 2020) Prisobljeno s https://en.wikipedia.org/wiki/Maze

Search A Maze For Any Path - Depth First Search Fundamentals (Similar To "The Maze" on Leetcode) (7. 11. 2020) Pridobljeno s: https://youtu.be/W9F8fDQj7Ok

Depth First Search (20. 11. 2020) Pridobljeno s: https://www.hackerearth.com/practice/algorithms/graphs/depth-first-search/tutorial/

Breadth First Search (20. 11. 2020) Pridobljeno s: https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/

Longest path in a directed Acyclic graph | Dynamic Programming (20. 11. 2020) Pridobljeno s: https://www.geeksforgeeks.org/longest-path-in-a-directed-acyclic-graph-dynamic-programming/

Longest path in a Directed Acyclic graph | Dynamic Programming | GeeksforGeeks (20. 11. 2020) Pridobljeno s: https://youtu.be/YxF-x3imVFA