Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dejan Petrović

Spletni pajki

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Matjaž Branko Jurič

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

[1] V dogovorju z mentorjem lahko kandidat diplomsko delo s pripadajočo izvorno kodo izda tudi pod katero izmed alternativnih licenc, ki ponuja določen del pravic vsem: npr. Creative Commons, GNU GPL. V tem primeru na to mesto vstavite opis licence, na primer tekst [**?**]

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00139/2013
Datum: 08.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DEJAN PETROVIĆ**

Naslov: **SPLETNI PAJKI**
WEB CRAWLERS

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Opredelite vlogo spletnih pajkov in razložite strukturo rekurzivnih in iterativnih pajkov. Podrobno predstavite arhitekturo in funkcionalnosti pajkov ter načine zajemanja podatkov. Implementirajte lastnega pajka, ki bo prikazal uporabo najboljših konceptov.

Mentor:

prof. dr. Matjaž B. Jurič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Dejan Petrović, z vpisno številko **63090082**, sem avtor diplomskega dela z naslovom:

Spletni pajki

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Matjaža Branka Juriča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 11. januarja 2013 Podpis avtorja:

Zahvala družini, prijateljem in punci, ki so mi stali ob strani.

Moji Alji.

Kazalo

Povzetek

Spletni pajek je avtomatiziran program oziroma skripta, ki samostojno preiskuje spletne strani na internetu. Ob tem skuša določiti in izvleči želene podatke iz spletnih strani. Podatki se potem shranijo v podatkovno bazo in se kasneje uporabljajo v različne namene. Nekateri pajki prenašajo spletne strani in jih shranjujejo v ogromne repozitorije, drugi pa iščejo bolj specifične podatke, na primer elektronske naslove ali telefonske številke. Najbolj znana in najpomembnejša uporaba spletnih pajkov je preiskovanje strani za namene spletnih iskalnikov, kot je recimo Google.

Cilj diplomske naloge je preučiti delovanje obstoječih spletnih pajkov in implementirati svojo različico pajka. Znotraj dela so opisani različni tipi spletnih pajkov, kaj je njihova naloga, potek preiskovanja strani, kje se po navadi začne proces preiskovanja in kako se izbirajo strani, ki jih bo pajek še preiskal. Nato sledi, kako pajek določi vsebino strani, katere podatke shrani ter kako in kam jih shranjuje. V nadaljevanju so opisane razlike med različnimi spletnimi pajki in njihova uporaba. Sledi primer implementacije delujočega spletnega pajka, ki začne preiskovanje na izbrani spletni strani in shranjuje odkrite podatke v podatkovno bazo.

Abstract

A web spider is an automated program or a script that independently crawls websites on the internet. At the same time its job is to pinpoint and extract desired data from websites. The data is then saved in a database and is later used for different purposes. Some spiders download websites which are then saved into large repositories, while others search for more specific data, such as emails or phone numbers. The most well known and the most important application of web crawlers is crawling websites for the purpose of search engines such as Google.

The aim of the thesis is to examine the performance of existing web spiders and implement our own version of the spider. In this thesis, we describe different types of spiders, what their goal is, the course of web crawling, where the crawling process is usually started and how to choose the pages that the spider will crawl. Followed by how the spider determines the content of a page, where the data is stored and in what way it's stored. Later we describe the differences between various web spiders and their use. Finally, we present an example implementation of a functioning Web crawler that starts crawling on selected web pages and stores the information found in a database.

Uvod

Informacije in podatki postajajo dandanes vse bolj pomembni in dragoceni, na internetu pa je na voljo praktično neomejena količina podatkov, razpršenih po nešteto različnih spletnih straneh. Zaradi tolikšnega obsega in dinamične narave interneta je potreben program, ki lahko v kratkem času obišče ogromno spletnih strani, pri tem pa išče podatke, ki nas zanimajo. Prav to je naloga spletnih pajkov. Kot nam pove ime, splet ali v angleškem jeziku web oziroma mreža spominja na pajkovo mrežo, saj se spletne strani med seboj povezujejo s povezavami in je mogoče po njih "potovati". Prav zato se taki programi imenujejo spletni pajki, saj sledijo povezavam in gredo tako od ene spletne strani do druge. Ker je na spletu na voljo veliko različnih podatkov in drugih vsebin, ki nas zanimajo, obstajajo različni tipi pajkov, ki nam omogočajo njihovo zbiranje. Čeprav nobena od nalog, ki jih trenutno imajo spletni pajki ni zanemarljiva, pa je ena izmed njih še posebej pomembna. Imenuje se spletno indeksiranje in neposredno omogoča delovanje spletnih iskalnikov, med drugimi tudi delovanje Googla.

V sodobnem času si je praktično nemogoče predstavljati življenje brez interneta. Vse več ljudi se obrne k spletnim iskalnikom za pomoč pri pridobivanju želenih informacij. Od vremenskih napovedi, rezultatov športnih dogodkov do nakupovanja preko spleta, branja novic in člankov. To nam omogočajo spletni iskalniki. Njihova naloga je prikazati seznam spletnih strani, ki najbolj ustrezajo uporabnikovi zahtevi. Da lahko spletni iskalniki uporabniku prikažejo relevantne rezultate, je potrebno vsaj okvirno poznati vsebino spletnih strani. Na tem mestu nastopijo spletni pajki in indeksira-

nje. Indeksiranje označuje zbiranje, obdelovanje in shranjevanje podatkov s spletnih strani na način, ki omogoča hitro in natančno iskanje informacij. Za učinkovito delovanje spletnih iskalnikov je potrebno preiskati čim več strani. S tem zagotovimo, da so uporabniku prikazane tiste strani, ki so zanj res pomembne.

Delo spletnih pajkov se nikoli ne konča, kljub temu, da internet vsak dan preiskuje ogromno število pajkov. Prvi vzrok za to je v količini strani na spletu. Že samo spletni iskalnik Google ima indeksiranih kar 48 milijard spletnih strani [16], pri tem je potrebno vedeti, da je ogromen del interneta še vedno neodkrit. Spletni pajek deluje tako, da začne svoje preiskovanje na naslovu določene spletne strani. Po navadi so to zelo znane in obširne strani, saj vsebujejo veliko število povezav na druge strani. Spletni iskalnik, na primer Google, torej pride do določene spletne strani tako, da sledi povezavam do nje z drugih spletnih strani. Če na neko neznano spletno stran ne kaže nobena druga stran, ki jo je preiskal pajek, bo posledično ta spletna stran ostala neodkrita. Nekatere enostavno niso dovolj znane, pomembne ali zanimive, da bi katera izmed drugih strani kazala nanje, medtem ko druge želijo ostati skrite.

Drugi vzrok za to, da se delo spletnih pajkov pravzaprav nikoli ne konča, je stalno spreminjanje vsebin spletnih strani. Zato da uporabnik dobi najnovejše informacije o temi, ki ga zanima, je pomembno da spletni iskalnik redno ažurira svoje indeksirane podatke. Ker se vsebina spletnih strani zelo pogosto spreminja, kar je še posebej očitno recimo pri straneh z novicami, saj se spreminjajo vsakodnevno, je pomembno, da spletni pajek tako stran preišče še enkrat. S tem se ažurirajo podatki, ki so bili shranjeni pri prejšnjem obhodu pajka.

Spletni pajki so torej ključni del spletnih iskalnikov. Neprestano preiskujejo ogromno količino spletnih strani, ob tem pa shranjujejo pomembne podatke v podatkovno bazo.

Poglavje 1

Vloga spletnih pajkov

Spletni pajki se uporabljajo v različne namene. V splošnem gre povsod za preiskovanje spletnih strani in njihovo shranjevanje oziroma shranjevanje njihovih pomembnih delov. Vseeno pa se njihova značilna uporaba deli na različna področja.

1.1 Spletni iskalniki

Daleč najpomembnejša in najvidnejša uporaba spletnih pajkov je preiskovanje strani za potrebe spletnih iskalnikov. Vsi popularni spletni iskalniki imajo implementirano svojo različico spletnega pajka, saj je nujno potreben za njihovo učinkovito delovanje. Izmed pajkov je najbolj znan Googlov, imenovan Googlebot. Tudi ostali iskalniki imajo svojega, na primer Bing-ov Bingbot, Yahoojev Slurp, MSN-jev Msnbot. Vsi omenjeni pajki se med seboj bolj ali manj razlikujejo, ponujajo različne funkcionalnosti (na primer Bing-ovo upoštevanje uporabnikove lokacije), nekateri so bolj drugi manj kompleksni, vendar je osrednja funkcionalnost pri vseh omenjenih pajkih enaka: shranjevanje in indeksiranje podatkov, pridobljenih s preiskovanjem spletnih strani.

1.2 Spletno arhiviranje

Precej manj vidna vloga spletnih pajkov je spletno arhiviranje. Gre za shranjevanje celotnih spletnih strani, ki so trenutno na internetu, po navadi v namene raziskovanja in ohranjanja kulture. Kopije spletnih strani se shranjujejo v arhive, dostopne raziskovalcem in zgodovinarjem, nekateri pa so tudi prosto dostopni na internetu. Trenutno največji arhiv shranjenih spletnih strani, ustanovljen leta 1996, je "Internet archive", katerega cilj je trajno shraniti čim večji del svetovnega spleta. Zaradi stalnega shranjevanja podatkov velikost arhiva hitro in vztrajno raste, od oktobra 2012 je velikost internetnega arhiva presegla 10 petabajtov (1000 terabajtov).

1.3 Podatkovno rudarjenje

Manj znan, a vse bolj popularen način uporabe spletnih pajkov, je za namene spletnega rudarjenja podatkov. Podatkovno rudarjenje je proces interpretacije podatkov oziroma iskanje informacij znotraj velike količine podatkov. Je eden izmed analitičnih načinov iskanja korelacij med ogromnimi količinami neorganiziranih podatkov, po navadi s pomočjo posebnih algoritmov.

1.4 Zbiranje specifičnih podatkov

Vse bolj popularni so spletni pajki, ki iščejo in shranjujejo točno določene podatke, na primer za delovanje kakšne aplikacije. Na primer, program NetAttache shranjuje dele strani, ki jih pogosto obiskujemo, na lokalni disk, kar omogoča hitrejše brskanje po internetu. Nekateri pajki omogočajo iskanje besedila, ki ga določi uporabnik, spet drugi iskanje elektronskih naslovov, telefonskih številk in podobno.

Poglavje 2

Struktura pajka

Spletni pajek je lahko implementiran na več načinov, z različnimi lastnostmi in funkcionalnostmi, vendar se v grobem po načinu delovanja delijo na dve skupini. Prva je realizacija pajka s pomočjo rekurzije, torej vsakič, ko pajek odkrije kakšen nov URL, se takoj usmeri nanj in se vrne na začetno stran, šele ko obišče vse druge povezave. Druga možnost pa je implementiranje pajka na iterativen način s pomočjo seznama povezav, kamor se shranijo vse povezave, ki jih spletni pajek odkrije med preiskovanjem strani in jih kasneje tudi obišče.

1. Začetek

   Algoritem se začne s seznamom URL-jev. Na tem seznamu so vsi naslovi strani, ki jih mora pajek še preiskati. Na začetku izvajanja je seznam najprej prazen, vanj se doda začetni URL, ki ga določi uporabnik ali pa je pridobljen na kakšen drug način. S seznama se izbere URL (trenutno je samo eden) strani, se označi kot obdelan in na izbrani URL se pošlje zahtevek HTTP.

2. Prenos strani

   Po poslanem zahtevku čakamo na odgovor strežnika. Da bi se izognili predolgemu čakanju na odziv strežnika lahko določimo časovno

omejitev (ang. timeout). Če se ta čas izteče, preden dobimo odgovor, se stran preskoči. Poleg tega lahko za večjo učinkovitost omejimo velikost prenesenega odgovora strežnika (po navadi 10-20 kb) [10], da se čas prenosa čim bolj skrajša, s čimer se poveča prepustnost pajka. Ko dobimo strežnikov odgovor (ang. HTTP response), ga je potrebno razčleniti (ang. parse), da dobimo statusno kodo in možne preusmeritve. Pomemben je tudi "last-modified"atribut znotraj strežnikovega odgovora, s čimer pridobimo starost dokumenta. Pri pajkih, ki služijo spletnim iskalnikom, se običajno strani prenašajo v repozitorij datotek na disku strežnika. V najosnovnejši obliki se vsaka stran shrani v svojo datoteko, za kar je potrebno zagotoviti edinstvena imena datotek. To lahko dosežemo s *hash* funkcijo, ki nam zagotovi enako dolge nize pri vsakem naslovu strani pri čim manjši verjetnosti kolizij. Ena izmed možnosti je MD5, *hash* funkcija, večinoma uporabljana v namene varovanja, ki iz vsakega niza pridobi 128-bitno vrednost. Tako z uporabo algoritma MD5 iz naslova http://www.google.com/ dobimo niz dolžine 128 znakov, "ff90821feeb2b02a33a6f9fc8e5f3fcd". S tem zagotovimo fiksne dolžine imen datotek za vsak vhodni naslov. Če je potrebno, je potem dobljeni repozitorij možno uporabiti tudi za preverjanje, vendar le v primeru, da je bila stran že preiskana [10].

3. Obdelava strani

   Po prenosu strani se iz dokumenta izlušči vsebina, ki nas zanima. Tu se pokaže največ razlik med pajki, saj ima vsak svoj specifičen namen. Lahko nas zanimajo vse slike znotraj strani, ali pa želimo najti vse elektronske naslove. Pri pajkih, kot je Googlebot, torej pri spletnih iskalnikih, je pomembna predvsem analiza teksta, ključne besede v besedilu, kolikokrat se pojavijo ... V vsakem primeru se na tem mestu poiščejo vse povezave znotraj dokumenta in če jih tam še ni, dodajo na seznam čakajočih povezav. Več o tem v poglavju 4.

4. Nov obhod zanke ali zaključek

Ko se konča obdelava strani, se zanka ponovno izvede. Zahteva se nov URL s seznama čakajočih povezav. V primeru, da dobimo veljaven URL, se ponovno pošlje zahtevek HTTP, čemur sledi prenos in obdelava strani. Če pa je seznam povezav že prazen, pride pri zahtevi za URL do napake, saj ni več razpoložljivih naslovov. Izvajanje programa se konča.

2.1 Rekurzivni pajek

Za razlago rekurzivnega načina implementiranja spletnega pajka je najprej potrebno vedeti, kaj rekurzija pravzaprav je. Rekurzija je programerska tehnika, pri kateri program, oziroma metoda znotraj programa, kliče samo sebe. V nekaterih primerih je omenjen način dobra izbira, sej je implementacija precej enostavnejša od iterativne implementacije. Primerna je predvsem takrat, ko delamo na bolj nezahtevnih pajkih, kjer je delovanje bolj enostavno, in ne preiskujemo velikega števila ogromnih spletnih strani. Težava je v tem, da se ves program shranjuje na sklad vsakič, ko se metoda na novo kliče. Posledično, če je neka stran prevelika in je potrebno slediti velikemu številu povezav, pride do prevelikega števila klicev metode, zaradi česar se spomin zapolni in program se ustavi. Še ena težava pri rekurzivnem načinu se pojavi, če želimo uporabiti paralelnost, torej več niti znotraj programa. Težava je v tem, da ima vsaka nit svoj sklad, kar pomeni, da rekurzija ne bi pravilno delovala. Zato v primeru, da želimo implementirati bolj obsežen program, ki nima težav z velikostjo in številom strani, uporabimo iterativni način implementacije [8].

2.2 Iterativni pajek

Implementacija pajka na iterativen način odpravi težave rekurzivne izvedbe pajka. Ker nismo omejeni na rekurzivni sklad, lahko nekatere dele programa izvajamo paralelno – v več nitih, kar drastično pohitri delovanje spletnega

pajka. Vsako spletno stran, ki jo mora pajek preiskati, lahko sedaj obdelujemo ločeno v svoji niti, torej lahko istočasno obdelujemo toliko strani, kolikor imamo na voljo niti. Iterativni pajek deluje z uporabo več seznamov, imenovanih vrste (ang. queues), v katerih hranimo povezave na druge strani, ki jih mora pajek še obiskati. Za optimalno delovanje pajka je potrebno implementirati več kot le en seznam povezav [8]. V tem primeru bodo uporabljeni štirje seznami:

- seznam čakajočih povezav,
- seznam izvajajočih povezav,
- seznam napak,
- seznam končanih povezav.

2.2.1 Seznam čakajočih povezav

Prva in najpomembnejša vrsta za nemoteno delovanje pajka. Je seznam vseh povezav, ki jih mora program še obiskati in preiskati. Sem pajek shranjuje vse povezave, ki jih je do sedaj odkril pri preiskovanju spletnih strani. Ker je na povprečni strani med 20 in 40 povezav, to pomeni, da se ta seznam zelo hitro povečuje in da je delo, ki ga mora pajek še opraviti, praktično neomejeno. Ko program prebere naslednji URL iz vrste čakajočih povezav, se URL prestavi v drug seznam – izvajajoče povezave. Ko je ta seznam povezav prazen, je delo končano in izvajanje pajka se konča.

2.2.2 Seznam izvajajočih povezav

Seznam izvajajočih povezav vsebuje vse naslove strani, ki jih pajek trenutno preiskuje. Če gre za pajka, ki je implementiran brez uporabe več niti, je ta naloga trivialna, saj je v seznamu vedno le ena povezava. V nasprotnem primeru pa je ta seznam zelo pomemben, da ne pride do večkratnega preiskovanja istih strani, kar lahko poleg izgubljenega časa privede tudi do težav za lastnike teh strani.

2.2.3 Seznam napak

Tu se hranijo vse povezave tistih strani, pri katerih je ob prenosu prišlo do napake iz različnih vzrokov. Od napak na strežniku do začasne nedostopnosti strani, pomanjkanja pravic za dostop in podobno. Seznam ima dve glavni vlogi. Tako kot pri vseh ostalih straneh in seznamih, je tudi pri straneh z napako potrebno shraniti naslov strani, da se lahko izognemo večkratnim preiskovanjem.

Druga vloga je pomembna le pri nekaterih pajkih, saj je odvisno od implementacije. Vzemimo za primer neko spletno stran, ki je na internetu dostopna 99 % časa. Ravno, ko jo obišče spletni pajek, pa stran začasno ni dostopna, torej je pajek na koncu ne preišče, le naslov se doda v seznam napak. Zato so nekateri pajki zasnovani tako, da se kasneje vrnejo na tako stran, v primeru, da je že dostopna.

2.2.4 Seznam končanih povezav

Seznam se uporablja za spremljanje napredka pajka in za preprečevanje večkratnih preiskovanj istih spletnih strani. Ko je določena stran obdelana, se njen naslov premakne sem iz seznama izvajajočih povezav. Vseeno pa dejstvo, da je naslov neke spletne strani shranjen v tem seznamu, še ne pomeni, da pajek te strani ne bo več obiskal. Razlog je v konstantnem spreminjanju strani in njihove vsebine, še posebej na primer kakšne informativne strani z novicami. Pajek mora stran torej večkrat obiskati, tako da so podatki čim bolj ažurirani.

Z uporabo omenjenih seznamov povezav lahko učinkovito realiziramo iterativno različico spletnega pajka. Vendar samo uporaba seznamov še ne pomeni, da se bo pajek dobro obnesel pri večjih količinah spletnih strani. Še vedno obstaja enaka težava kot pri rekurzivni obliki programa. Za shranjevanje seznamov spletnih strani se uporablja delovni spomin računalnika, kar pomeni, da bo tudi pri iterativni izvedbi pajka prišlo do pomanjkanja prostega pomnilnika in se bo s tem delovanje programa ustavilo. V povprečju

lahko na nekem računalniku pajek, ki ima seznam povezav shranjen v spominu, normalno deluje pri okoli 10.000 straneh. Pri večjem številu strani pa se pri delovanju pajka že pojavljajo težave. Težavo lahko rešimo s pomočjo baze SQL (ang. Structured Query Language). SQL je poseben programski jezik, zasnovan za urejanje podatkov, hranjenih v relacijski podatkovni bazi.

2.2.5 Uporaba podatkovne baze SQL

Za implementacijo visoko zmogljivega spletnega pajka, ki lahko preiskuje veliko večje količine naslovov spletnih strani, uporabimo bazo SQL za shranjevanje seznamov povezav. Z uporabo baze občutno povečamo količino spletnih strani, ki jih pajek preiskuje. Če upoštevamo, da lahko po potrebi uporabimo tudi več baz SQL, je število strani praktično neomejeno. Sezname oziroma vrste, ki jih potrebujemo za shranjevanje povezav, lahko realiziramo na dva načina.

Pri prvem načinu za vsak seznam (čakanje, izvajanje, končani, napake) ustvarimo svojo tabelo. Poleg primarnega ključa in naslova spletne strani shranimo tudi datum zadnjega preiskovanja strani, v primeru, da želimo večkratno preiskovanje za ažuriranje vsebine spreminjajočih strani. Vsakič, ko se stanje obdelave strani spremeni, na primer iz "čakanja"na "izvajanje", se izbriše vrstica v tabeli čakajočih strani z ukazom SQL DELETE FROM in se vstavi v tabelo izvajajočih z ukazom INSERT INTO. Tako se vrstica z naslovom neke spletne strani premika po seznamih in na koncu pristane v seznamu končanih, če se uspešno preišče, oziroma v seznamu napak, če med preiskovanjem pride do kakšne napake.

Težava v takem načinu uporabe baze SQL je v potratnosti časa, saj lahko program izboljšamo tako, da namesto dveh ukazov, INSERT in DELETE, uporabimo le UPDATE. Tako realiziramo drugi, bolj učinkovit način uporabe seznamov. Implementiramo ga tako, da uporabimo le eno tabelo z vsemi povezavami, ki imajo dodan atribut *status*, kamor se shrani eno izmed naslednjih stanj: W – *waiting*, R – *running*, C – *completed*, E – *error*. Namesto da pri vsaki menjavi stanja brišemo vrstico s trenutnega seznama in jo do-

damo na naslednjega, lahko sedaj le spremenimo atribut *status*, na primer iz *running* na *completed*, torej iz izvajajočega v končano stanje. S tem pri vsaki spremembi stanja privarčujemo en ukaz SQL. Na ta način je realiziran tudi pajek v tem diplomskem delu.

Poglavje 3

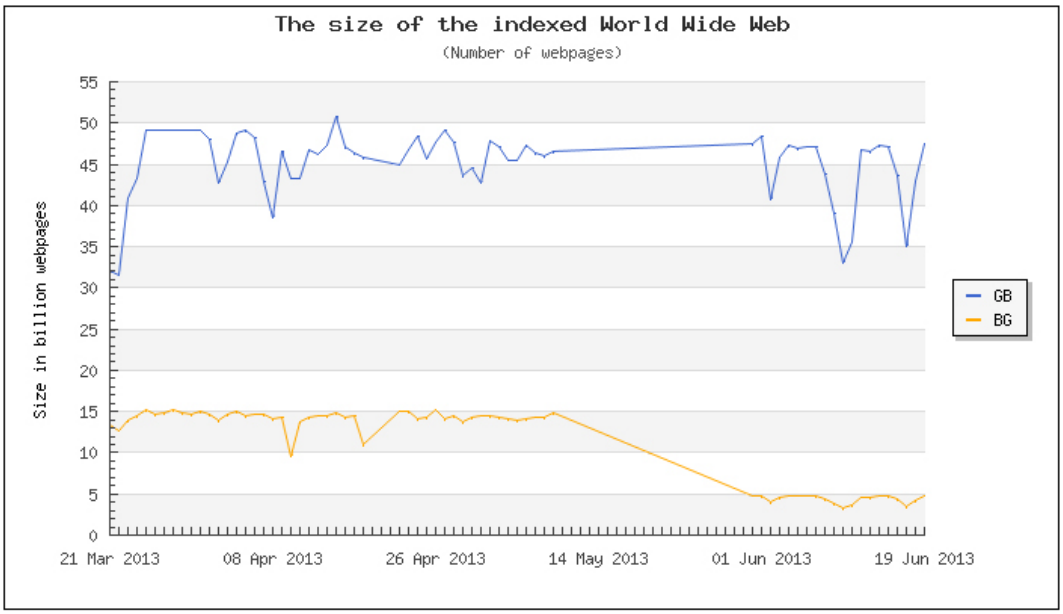
Arhitektura pajka

Naloga spletnih pajkov je v večini primerov zelo podobna, razlikujejo se predvsem po obsegu njihovega dela. Konceptualno je algoritem spletnega pajka zelo enostaven [1]:

- Izberi URL izmed množice naslovov.
- Prenesi vse spletne strani, povezane s tem naslovom.
- Izlušči vse povezave znotraj teh strani.
- Dodaj vse strani, ki še niso bile preiskane v začetno množico povezav.

Preprostega in delujočega spletnega pajka je dejansko možno napisati v le nekaj vrsticah kode, vendar implementacija učinkovitega spletnega pajka, ki je primeren za preiskovanje večjega števila spletnih strani, predstavlja velik inženirski izziv. Izkaže se, da je največja težava pri pajku skalabilnost. Skalabilnost (tudi razširljivost) označuje sposobnost programa ali sistema za obvladovanje vse večjih količin dela oziroma pripravljenost programa na razširitev, da lahko potem sprejme večjo količino dela [1].

Pomembnost skalabilnosti se kaže predvsem pri tistih pajkih, ki skrbijo za delovanje spletnih iskalnikov. Tako zaradi ogromne količine spletnih strani, ki vsakodnevno še vedno strmo narašča, kot tudi zaradi konstantnega spreminjanja njihove vsebine. Za primer lahko vzamemo Googlov Googlebot, ki ima



Slika 3.1: Velikost indeksiranega spleta pri Google-u in Bing-u [16].

trenutno največje število indeksiranih spletnih strani v svoji podatkovni bazi. Poleg tega, da je potrebno preiskati in indeksirati vse nove strani, ki so bile oddane Googlu s strani lastnikov, ali pa so bile odkrite med preiskovanjem drugih strani, je potrebno tudi zagotoviti, da so vse strani, ki jih indeksira, ažurirane. Po zadnji oceni Google trenutno indeksira kar 48 milijard spletnih strani.

Googlebot mora torej preiskovati nove strani, vsak dan se po ocenah pojavi vsaj ena milijarda novih strani, polega tega pa mora še vzdrževati vsebino vseh že indeksiranih strani. Zagotavljanje ažuriranosti vseh spletnih strani v indeksu je praktično nemogoče, vendar želimo, da je vsebina čim bolj sveža. Osnovni spletni pajek je možno realizirati precej enostavno in hitro, če pa potrebujemo visoko zmogljivega pajka, ki je sposoben obdelave ogromnih količin spletnih strani v kar najkrajšem času, je potrebno uporabiti ustrezno arhitekturo.

3.1 Porazdeljenost

3.1.1 Enotni pajek

Enotni pajek (ang. single web crawler) je najbolj enostavna zasnova spletnega pajka. Vsa logika se izvaja na enem računalniku, z izjemo uporabe baze SQL na oddaljenem strežniku v primeru, da se uporablja SQL za shranjevanje seznamov in prenesene vsebine. Ker je najbolj enostaven, je tudi najbolj neučinkovit, vsaj kar se tiče hitrosti preiskovanja strani. Noben del programa se ne izvaja paralelno, tok izvajanja se dogaja v eni niti. Uporaben je predvsem pri zelo specifičnih nalogah, ki so omejene na manjše število strani, torej recimo za namene indeksiranja spletnega iskalnika ni priporočljiv. Količina spletnih strani, ki jih lahko preišče v določenem času, je odločno premajhna. Uporabimo ga lahko na primer za preverjanje delovanja povezav na spletni strani.

Prednosti

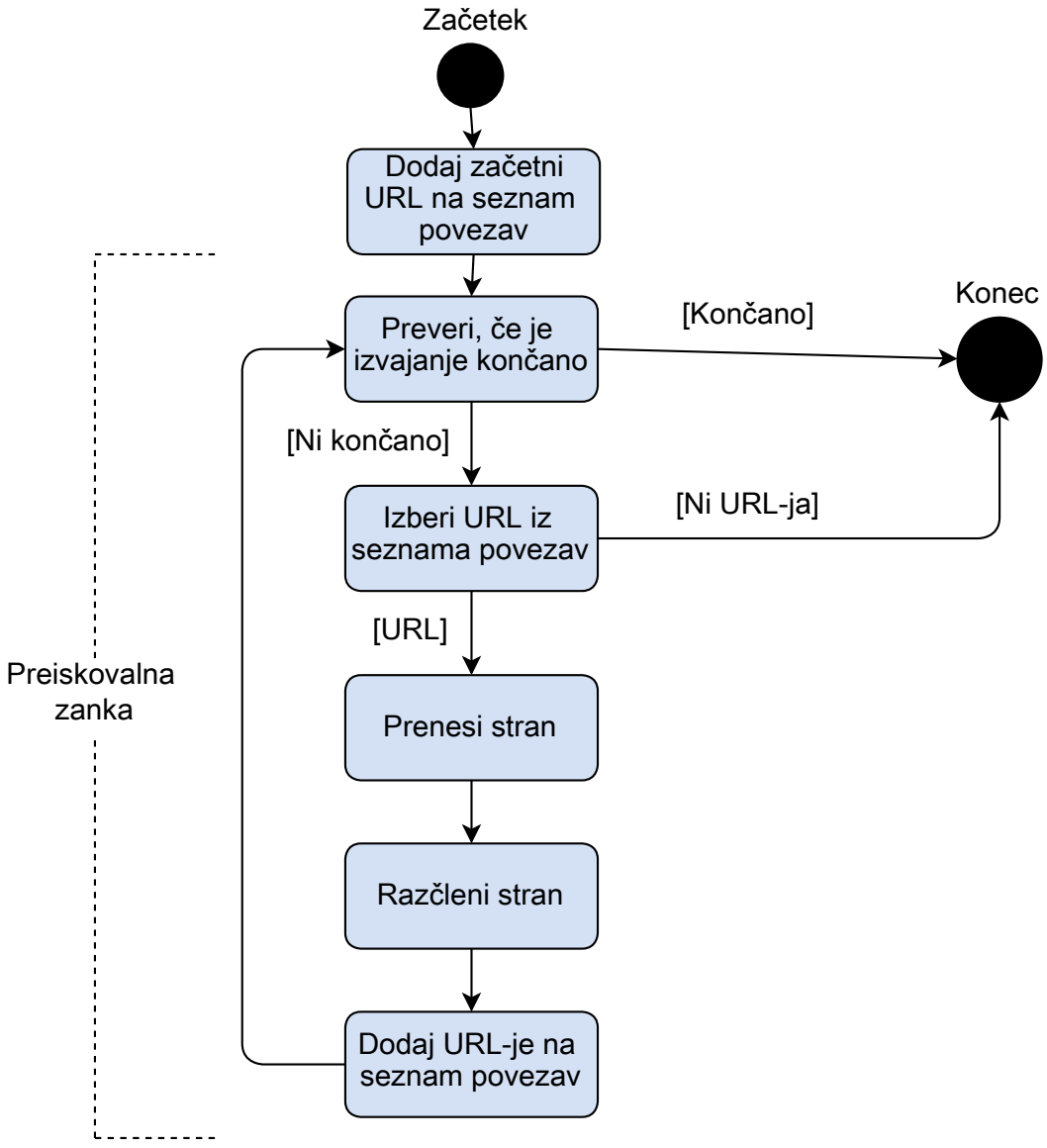
- Enostaven za implementacijo.
- Ni težav pri izbiranju naslednjega naslova strani.
- Ni potrebna komunikacija z ostalimi nitmi ali pajki.

Slabosti

- Slaba prepustnost (ang. throughput), količina prenesenih podatkov na enoto časa je zelo nizka.
- Ni skalabilen.

3.1.2 Večnitni pajek

Pri enotnem pajku je veliko prostora za izboljšanje delovanja programa. Tako enotni kot tudi večnitni pajek (ang. multithreaded web crawler) tečeta na



Slika 3.2: Diagram enostavnega pajka.

enem računalniku. Če opazujemo zasedenost procesorja in omrežja, lahko vidimo, da je velik delež časa vsaj eden od njiju v mirovanju, po navadi, ko čakata eden na drugega. Vzemimo za primer pajka v izvajanju, ki prenese določeno spletno stran in jo začne obdelovati. V tem času vse delo opravlja procesor, omrežje pa ostaja neuporabljeno. Želimo implementacijo pajka, ki bo bolje uporabila neizkoriščene vire računalnika. To lahko dosežemo z večnitnim pajkom. Kot že namiguje ime samo, gre za tip pajka, ki pri preiskovanju spletnih strani uporablja več niti, kar pomeni, da se nekateri deli pajka izvajajo vzporedno.

Pri večnitni realizaciji pajka določenih delov ni mogoče izvajati paralelno. Predvsem tistih odsekov programa, do katerih dostopajo vse niti. To so po navadi parametri, ki jih potrebuje funkcija znotraj posamezne niti. Posledično so to največkrat funkcije, s katerimi nastavimo oziroma pridobimo kakšen parameter (*get* in *set* funkcije). Omenjene funkcije je sicer mogoče izvajati vzporedno, vendar bi prišlo do nepravilnosti pri delovanju programa. Za primer poglejmo, kaj se zgodi pri spletnem pajku, če uporabljamo večnitnost in dovolimo, da se tudi metode za nastavljanje in pridobivanje parametrov izvajajo paralelno.

Recimo, da imamo metodo imenovano *getURL*. Ta metoda vrne naslov naslednje spletne strani, ki jo je potrebno preiskati. Vsaka posamezna nit znotraj programa najprej potrebuje naslov (URL) strani, zato kliče metodo *getURL*, s tem pridobi niz črk, ki kaže na neko stran. Ta naslov se nato izbriše s seznama čakajočih in se vstavi v seznam izvajajočih povezav (oziroma njen status se spremeni iz "waiting"v "running"). Nato se pošlje zahteva, s tem se odpre stran, ki se nato shrani in obdela. Ko je delo končano, se ponovno kliče *getURL* in s tem se začne preiskovanje nove strani. Potrebno pa je upoštevati, da se vse to dogaja v posameznih nitih vzporedno. Lahko se torej zgodi, da več niti istočasno kliče metodo *getURL* in s tem vse niti dobijo isti naslov, kar pomeni da se ista stran preišče večkrat. Če je recimo brisanje naslovov iz seznama čakajočih realizirano tako, da se izbriše le prvi naslov v seznamu, se lahko zgodi, da se izbriše več naslovov, kljub temu

da je bil le eden dejansko preiskan. Potrebno je zagotoviti, da do takih primerov ne more priti. Znotraj samega programa je potrebno poskrbeti za sinhronizacijo. Ker je pajek v tem diplomskem delu implementiran v Javi, bo predstavljena rešitev za omenjene težave v tem programskem jeziku. V Javi je sinhronizacija omogočena s pomočjo zaklepanja objektov in ključne besede *synchronized* [2].

S tem, ko v glavo metode dodamo to ključno besedo, določimo kritično sekcijo. Na primer metoda "public synchronized void getParam()". Telo metode je sedaj kritična sekcija, kar pomeni, da v nekem trenutku lahko le ena nit izvršuje ukaze znotraj te metode. Ko ena izmed niti vstopi v kritično sekcijo, se ta sekcija zaklene in ostane zaklenjena dokler trenutna nit ne konča z izvajanjem te metode. Če katera izmed drugih niti kliče sinhronizirano metodo medtem, ko je zaklenjena, mora čakati dokler se metoda ne odklene. Poljubno metodo torej sinhroniziramo takrat, ko želimo, da jo ob določenem času lahko izvaja le ena nit. Potrebno pa je biti previden pri njeni uporabi, saj lahko s prekomernim sinhroniziranjem metod tudi poslabšamo učinkovitost programa. Če bi bil celoten program kritična sekcija, bi uporaba večnitnosti izgubila svoj pomen, saj bi se vedno lahko izvajala le ena nit.

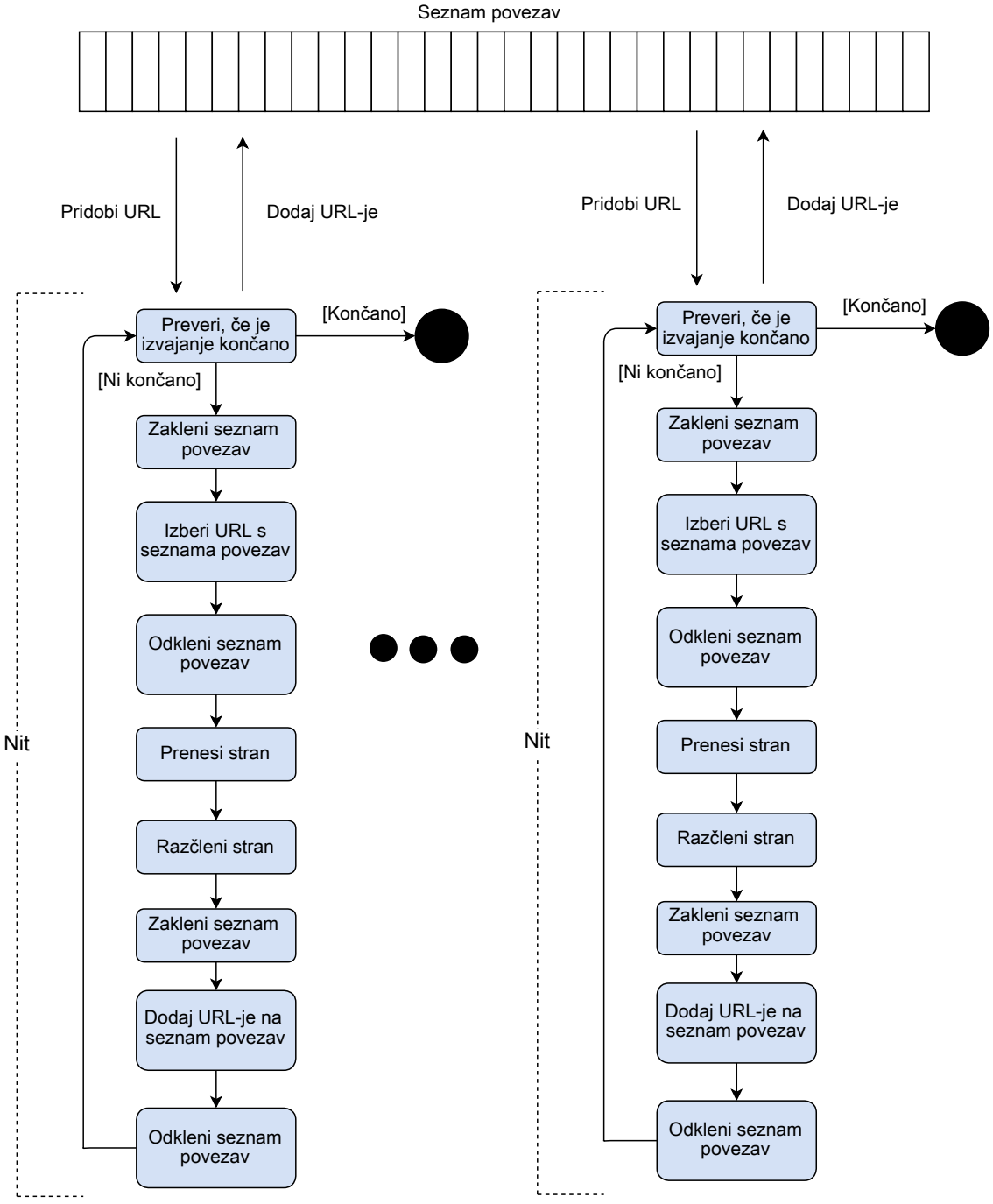
Uporaba niti in s tem paralelnega izvajanja občutno izboljša učinkovitost programa, ki se meri v količini preiskanih strani, potrebno pa je paziti na nove morebitne težave, ki se s tem pojavijo.

Prednosti

- Večja prepustnost kot pri enotnem pajku.
- Boljše izkoriščanje računalnikovih virov.

Slabosti

- Potrebna komunikacija med nitmi.
- Težave pri izbiranju naslednje povezave iz seznama čakajočih.



Slika 3.3: Diagram večnitnega pajka.

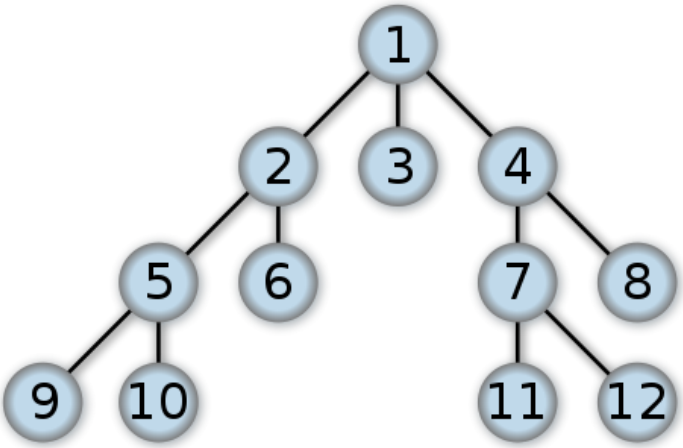
3.1.3 Distribuirani pajek

Distribuirani spletni pajek (ang. distributed web crawler) je daleč najkompleksnejši od vseh izvedb, vendar je ravno zato lahko tudi najbolj učinkovit. Za razliko od enotnega in večnitnega spletnega pajka, se ta izvaja na več računalnikih. Beseda distribuirani se nanaša na geografsko lokacijo, računalniki, na katerih se programi izvajajo, so v različnih mestih, državah ali kontinentih. Zaradi njegove arhitekture je distribuiran pajek najbolj skalabilen izmed vseh implementacij in je zato tudi najbolj primeren za preiskovanje večjih količin strani. Še najbolj ustrezen je recimo za namene spletnega iskalnika, kjer se delo pajka nikoli ne konča, pomembna je le hitrost preiskovanja. V osnovi je algoritem pajka enak prejšnjim različicam pajkov, vendar se sedaj pojavijo težave pri seznamu naslovov, ki jih je potrebno preiskati. Seznam ni več centraliziran oziroma skupen vsem na enem mestu, temveč se deli (ang. partitioning) med posameznimi vozlišči (ang. nodes) distribuiranega sistema [3].

Deljenje prostora, ki ga je potrebno preiskati, lahko realiziramo s pomočjo *hash* funkcije in s tem porazdelimo naslove med posamezna vozlišča. Druga možnost deljenja je glede na geografsko lokacijo, na primer vozlišče, ki je situirano v Evropi, preiskuje evropske domene. Ne glede na izbrani način deljenja prostora, je potrebno poskrbeti za odstranjevanje duplikatov, preden se naslov strani lahko doda na seznam povezav določenega vozlišča. Vsako vozlišče ima svoj sistem za odstranjevanje duplikatov, ki novo povezavo primerja z vsemi ostalimi znotraj internega seznama že obiskanih povezav. Preden se nov naslov strani doda v eno izmed vozlišč, gre pred tem ta naslov skozi sisteme za odstranjevanje duplikatov vseh ostalih vozlišč. Tako se prepreči večkratno preiskovanje istih strani na različnih vozliščih [3].

Prednosti

- Največja prepustnost.
- Delitev dela med več sistemov.



Slika 3.4: Pajek s preiskovanjem v širino.

Slabosti

- Usklajevanje dela in rezultatov.
- Težave pri izbiranju naslednje povezave sz seznama čakajočih.

3.2 Tipi spletnih pajkov

Obstajajo različni tipi spletnih pajkov, razlikujejo se predvsem po načinu pridobivanja zaporednih spletnih strani. V nadaljevanju so opisani najpogostejši tipi pajkov.

3.2.1 Pajek s preiskovanjem v širino

Pajek s preiskovanjem v širino (ang. breadth first crawler) začne preiskovanje z majhno množico spletnih strani, nato pa preiskuje ostale strani tako, da sledi povezavam na "breadth first"način oziroma "v širino". Pri tem načinu pajek takoj obišče vsako novo povezavo, ki jo najde, ne glede na to, kateri spletni strani pripada.

Na sliki 3.4 lahko vidimo postopek preiskovanja posameznih vozlišč, ki so v našem primeru spletne strani. Vedno se najprej preišče nova stran, ko se odkrije, šele nato se bolj podrobno do konca preiščejo že znane strani.

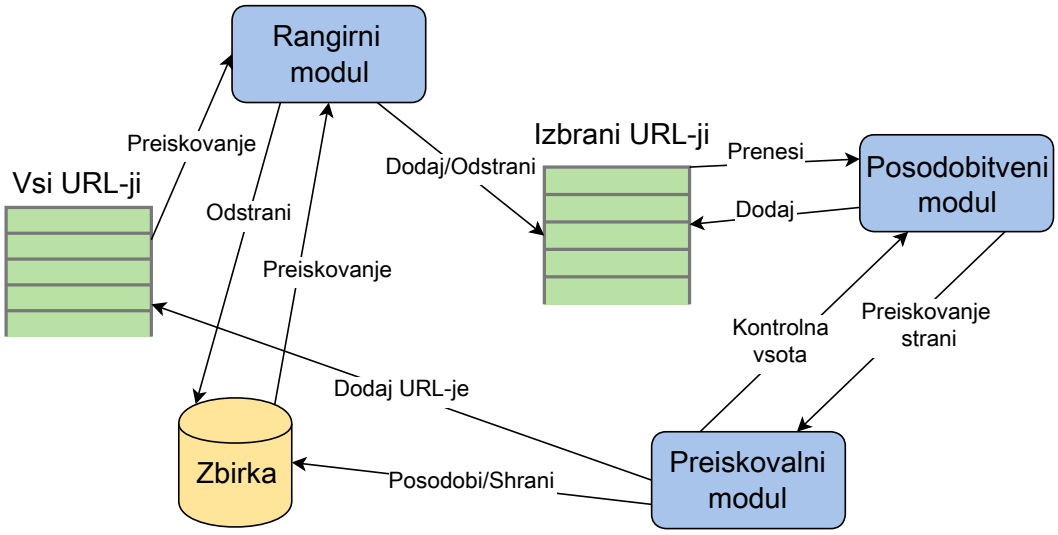
Na primer, pajek preiskuje stran www.najdi.si in odkrije povezave www.google.com, www.najdi.si/novice, www.facebook.com, www.twitter.com in www.najdi.si/zemljevid. Namesto da bi najprej preiskal vse podstrani, ki pripadajo k Najdi.si, bo pajek obiskal vse povezave v enakem vrstnem redu, kot jih je odkril. Če pajek ne bi uporabil iskanja v širino, bi preiskal strani v tem vrstnem redu:

1. www.najdi.si/novice
2. www.najdi.si/zemljevid
3. www.google.com
4. www.facebook.com
5. www.twitter.com

Pri tem tipu pajka, pa bi bil potek preiskovanja videti takole:

1. www.google.com
2. www.najdi.si/novice
3. www.facebook.com
4. www.twitter.com
5. www.najdi.si/zemljevid

Spletni pajek zaradi preiskovanja v širino odkrije veliko število povezav v zelo kratkem času, zato je še posebej primeren za večje sisteme, kjer je veliko procesov ali niti, saj jim potem ni potrebno čakati na nove povezave [4].



Slika 3.5: Inkrementalni pajek.

3.2.2 Inkrementalni pajek

Inkrementalni pajek (ang. incremental crawler) deluje tako, da konstantno ažurira množico že prenesenih strani, namesto da bi vsakokrat ponovno začel preiskovanje z ničle [4]. To nam na nek način omogoča, da ugotovimo, če se je določena stran spremenila, odkar je bila nazadnje preiskana. Je tip pajka, ki neprestano preiskuje celotni splet na osnovi množice preiskovalnih ciklov. Uporablja adaptivni model za določanje vrstnega reda preiskovanja strani na osnovi podatkov iz prejšnjih preiskovalnih ciklov. S tem dosežemo visoko stopnjo ažuriranosti repozitorija spletnih strani.

Na sliki 3.5 vidimo priporočeni način implementacije inkrementalnega pajka. Črte in puščice prikazujejo pretok podatkov med posameznimi moduli, oznake na črtah pa pripadajoči ukaz. Seznam *Vsi URL-ji* vsebuje vse URL-je oziroma naslove, ki jih je pajek odkril, seznam *Izbrani URL-ji* pa vse URL-je, ki so, ali še bodo v *Zbirki* (ang. Collection). *Izbrani URL-ji* je implementiran kot prioritetna vrsta, kjer so URLji, ki bodo najprej preiskani, postavljeni na začetek.

URLje v izbranih naslovih izbere *Rangirni modul*. Rangirni modul kon-

stantno preiskuje vse naslove v seznamu *Vsi URL-ji* in ocenjuje njihovo pomembnost glede na izbrani parameter. Če se zgodi, da med izbranimi naslovi ni strani, za katero se izkaže, da je bolj pomembna od strani, ki tam je, potem rangirni modul zamenja najmanj pomembno stran v izbranih naslovih z novo odkrito stranjo. Poleg tega rangirni modul odstrani najmanj pomembne strani iz zbirke, da naredi prostor za novo stran.

Posodobitveni modul skrbi za ažuriranost zbirke. Konstantno jemlje prvi URL iz izbranih naslovov, zahteva, da ga preišče *Preiskovalni modul* in ga nato vstavi nazaj v izbrane naslove. Poleg tega posodobitveni model neprestano ugotavlja ažuriranost posamezne stvari in sproži ponovno preiskovanje, če je potrebno [15].

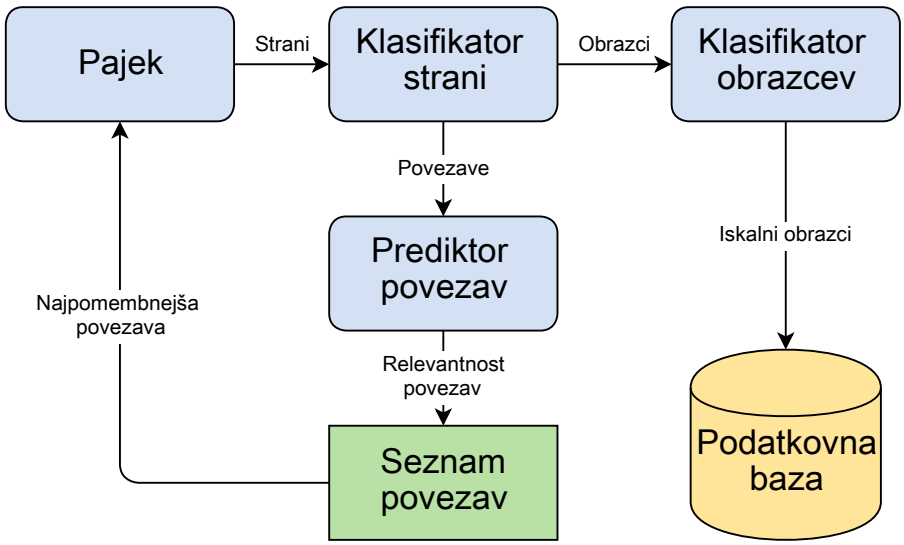
3.2.3 Pajek, usmerjen na obrazce

Veliko število pajkov se ujame v tako imenovane pasti, ki se po navadi pojavijo pri obrazcih (ang. form). Ker je pri nekem obrazcu lahko neomejeno število povezav (parametri pri get zahtevi), se pajek lahko ujame v zanko in neprestano preiskuje povezave, pridobljene s pomočjo obrazca. Prav zato je večina spletnih pajkov implementirana tako, da ignorira povezave iz obrazcev. Težava je v tem, da lahko na ta način izpustimo veliko vsebine, ki nas sicer zanima. Zato se je razvil pajek, osredotočen na obrazce (ang. form focused crawler). Ta pajek se izogiba neproduktivnim potem tako, da:

- omeji iskanje na določeno temo,
- prepoznava lastnosti povezav in poti, ki vodijo do strani z obrazci,
- uporablja primerne kriterije za zaustavitev.

3.2.4 Usmerjeni pajek

Usmerjeni pajek (ang. focused crawler) je vrsta pajka, ki preiskuje le tiste spletne strani, ki ustrezajo določenim pogojem [4]. Ti pogoji se lahko precej razlikujejo. Nekaj primerov uporabe usmerjenega pajka:



Slika 3.6: Usmerjeni pajek.

- preiskovanje le tistih strani, ki imajo domeno .si,
- preiskovanje visoko ocenjenih strani,
- preiskovanje spletnih strani, ki se nanašajo na določeno temo.

Na sliki 3.6 je vidno, da je usmerjeni pajek sestavljen iz treh glavnih komponent. Klasifikator, ki določa relevantnost preiskanih spletnih strani, prediktor, ki poskuša določiti prioriteto strani, še preden je le-ta obiskana, in dinamično nastavljiv pajek, ki ga upravljata klasifikator in prediktor.

3.2.5 Skriti spletni pajek

Precejšnja količina podatkov na internetu se skriva znotraj podatkovnih baz, do katerih lahko pridemo le s pošiljanjem ustreznih poizvedb (ang. query) ali z izpolnjevanjem obrazcev na internetu. V zadnjem času je veliko zanimanja za podatke znotraj tako imenovanega "globokega"ali skritega interneta (ang. deep web). Današnji pajki preiskujejo le tisti del interneta, ki je javno indeksiran (ang. publicly indexable web – PIW), torej spletne strani, ki so

dosegljive s sledenjem povezav, pri čemer ignorirajo iskalne strani in obrazce, ki zahtevajo avtorizacijo ali predhodno registracijo. V realnosti lahko tovrstni spletni pajki izpustijo ogromne količine visoko kakovostnih podatkov, ki se skrivajo za iskalnimi obrazci (ang. search forms). Zato se je razvil skriti spletni pajek (ang. hidden web crawler).

3.3 Podatkovne strukture

Pri preiskovanju uporabljajo pajki različne podatkovne strukture, ki zagotavljajo učinkovito delovanje.

3.3.1 Repozitorij

Shranjuje in upravlja z velikimi količinami spletnih strani. Vsebuje celotno kodo HTML vsake posamezne strani, ki jo spletni pajek obišče in prenese. Preden se shranijo v repozitorij, se najprej kompresirajo, s čimer zagotovimo, da lahko shranimo kar največ strani. Za kompresiranje strani lahko uporabimo različne kompresijske algoritme, ki so kompromis med hitrostjo in stopnjo kompresiranja. V repozitoriju se dokumenti shranjujejo eden za drugim [4].

3.3.2 Dokumentni indeks

Vzdržuje informacije o vsakem dokumentu. Je indeks fiksne dolžine tipa ISAM (Index Sequential Access Mode), urejen po parametrih dokumenta, ki vključujejo status dokumenta, kazalec na repozitorij (kje se dokument dejansko nahaja) in nekatere statistične podatke. Če je dokument že preiskan, vsebuje tudi kazalec na datoteko, ki vsebuje njegov URL in naslov. Sicer kaže na seznam URL-jev, ki vsebuje le njegov URL [12].

3.3.3 Indekser

Je program, ki bere vse strani, ki so bile prenesene. Internetno indeksiranje vključuje indekse posameznih spletnih strani in dokumentov zaradi zagotavljanja čim boljšega slovarja za internetne iskalnike [4]. Indekser prav tako obravnava kodo HTML, ki sestavlja spletno stran in išče besede, ki so pomembne. Sem spadajo besede, ki so krepke ali ležeče in tiste, ki ležijo v glavi dokumenta.

3.3.4 Seznam zadetkov

Shranjuje seznam pojavitev določene besede v določenem dokumentu, med drugim podatke o položaju, pisavi, kapitalizaciji. Obstajata dva tipa zadetkov:

- Posebni zadetki: zadetki znotraj URLja, naslovov, sider (ang. anchor) in zadetki znotraj meta oznak.
- Navadni zadetki: zajemajo vse ostalo.

Poglavje 4

Funkcionalnosti pajkov

Kot smo spoznali v poglavju 3, obstaja veliko različnih tipov spletnih pajkov. Kljub temu pa obstajajo splošna pravila ali funkcionalnosti, ki so zaželena pri vsaki implementaciji pajka.

4.1 Hitrost

Zaradi praktično neskončnega dela, ki ga morajo opravljati spletni pajki, je hitrost preiskovanja ena najpomembnejših lastnosti. Vzemimo za primer Google. Po zadnji oceni imajo indeksiranih 48 milijard spletnih strani. Če lahko naš pajek preiskuje le eno stran naenkrat in je za eno zahtevo HTTP in njen odgovor potrebno čakati eno sekundo, to pomeni, da lahko v enem dnevu preišče 86.400 spletnih strani. Da bi indeksirali toliko strani, kot jih ima Google, bi torej potrebovali 578.704 dni oziroma 1585 let, pri čemer ne upoštevamo, da je strani potrebno redno ponovno preiskovati, da so iskalni rezultati čim bolj sveži. Z optimizacijo programa in paralelnostjo je potrebno zagotavljati visoko hitrost pajka, sicer je praktično neuporaben.

4.2 Vljudnost

Spletni pajki lahko pridobivajo podatke veliko hitreje in bolj učinkovito kot človeški uporabniki, kar lahko privede do hudih posledic pri razpoložljivosti spletne strani. Ni potrebno posebej poudarjati, da lahko pajek pošlje več zahtev na sekundo oziroma opravlja večje prenose datotek na strani, zaradi česar ima lahko strežnik težave s serviranjem vseh zahtev, če stran preiskuje več spletnih pajkov naenkrat.

Uporaba spletnih pajkov je lahko zelo koristna, vendar ima tudi svojo ceno. Za splošnega uporabnika lahko prinese tudi negativne posledice, med drugim:

- Poraba omrežnih virov, saj spletni pajki zahtevajo velik del pasovne širine in delujejo z visoko stopnjo vzporednosti v daljšem časovnem obdobju.

- Preobremenitev strežnika, še posebej, če je frekvenca dostopov do določenega strežnika previsoka.

- Slabo napisani pajki, ki lahko zrušijo strežnike ali usmerjevalnike, ali pa prenesejo strani, ki jih ne morejo obdelati.

- Zasebni pajki, ki lahko povzročijo motnje v delovanju omrežij ali strežnikov, če jih požene preveč uporabnikov. Tako obliko napada imenujemo DOS (ang. Denial of Service) oziroma DDOS (ang. Distributed Denial of Service). Gre za preprečitev uporabe storitve njenim uporabnikom.

Delna rešitev te težave je izključitveni protokol, znan tudi kot *robots.txt* protokol [6]. Je standard za administratorje, da navedejo, do katerih delov njihovega strežnika spletni pajki nimajo dostopa. Ta standard ne vključuje predloga za časovni interval med obiski istega strežnika, kljub temu da je omenjeni interval najbolj učinkovit način za preprečevanje preobremenitve strežnika. Nedavno so komercialni spletni iskalniki, kot so Ask Jeeves, MSN

in Yahoo postali sposobni uporabiti dodaten parameter, imenovan zamik preiskovanja (ang. Crawl Delay) v *robots.txt* datoteki, ki označuje zamik med zahtevami v sekundah.

Če želimo, da spletni pajki katerih od naših spletnih strani ne preiskujejo, lahko torej na strežnik dodamo datoteko *robots.txt*, ali pa dodamo tak primer kode znotraj HTML-ja tiste strani, za katero želimo, da je roboti ne obiščejo:

```
<META NAME=ROBOTS CONTENT="NOINDEX, NOFOLLOW">.
```

4.3 Podvojena vsebina

Po nekaterih ocenah je trenutno do 40 % vsebine na internetu podvojene [3]. Velik del teh strani je legitimnih, na primer informacijska skladišča (repozitoriji) so kopirani zaradi zagotavljanja redundance in zanesljivosti. Spletni iskalniki se poskušajo izogniti indeksiranju strani z enako vsebino zaradi zmanjševanja stroškov skladiščenja in za zniževanje potrebnega procesiranja.

Najlažji način odkrivanja duplikatov je z izračunom izvlečka ali prstnega odtisa vsake spletne strani. Kadar sta prstna odtisa dveh različnih strani identična, je potrebno testirati, ali sta tudi sami spletni strani enaki, in če sta, označimo eno od strani kot duplikat. Težava pri tem načinu odkrivanja duplikatov je v tem, da večina strani na internetu ni povsem enaka med sabo, temveč so skorajšnji duplikati. V velikem številu primerov je vsebina ene spletne strani povsem identična drugi, z izjemo nekaj znakov, recimo zapis, ki prikazuje čas in datum strani, ko je bila nazadnje spremenjena. Tudi v takem primeru si želimo, da bi lahko odkrili, da sta si spletni strani tako podobni, da se indeksira le ena izmed njiju. Tudi za to obstajajo rešitve, vendar so računsko veliko bolj zahtevne.

4.4 Neželena vsebina

Internet je dandanes najpomembnejši in najdonosnejši način oglaševanja. Prav zato je vse bolj pomembno, da se naša stran pojavi čim višje, ko nekdo

v spletni iskalnik vpiše neko besedo ali besedno zvezo, ki je povezana z našo stranjo. Tu se pojavi neželena vsebina (ang. spam).

Spletni iskalniki imajo kriterije, po katerih ocenjujejo pomembnost strani. Na osnovi te ocene se razvrstijo rezultati iskanja uporabnika. Da bi stran prišla čim višje, nekateri na svoje strani dodajajo vsebino, ki služi le pridobivanju višje ocene, ne pa tudi uporabnikom. Sem spada na primer dodajanje besedila na stran, ki ga uporabnik sploh ne vidi. To dosežejo z barvo, saj je barva ozadja enaka barvi pisave. Druga, precej popularna neželena vsebina, ki se pojavlja na spletnih straneh, je pretirano uporabljanje določene ključne besede (ang. keyword). Ker je ključna beseda precej pomembna pri rangiranju strani, jo poskušajo čim večkrat ponoviti v besedilih, da bi s tem privabili čim več ljudi na svojo stran. Težavo rešijo tako, da omejijo število ključnih besed, ki se lahko pojavijo, sicer se stran oceni slabše.

K neželeni vsebini spada tudi prekomerno dodajanje povezav, ki prav tako izboljšajo oceno strani pri spletnih iskalnikih. Več je strani, ki imajo povezavo na našo stran, bolje bo naša stran ocenjena. Prav zato je ustvarjenih veliko število lažnih strani, katerih naloga je le vsebovati povezave na določene strani in s tem povečevanje njihove ocene.

4.5 Robustnost

Obstajajo strežniki oziroma spletne strani, ki vsebujejo tako imenovane "pasti za pajke"(ang. spider traps). V te pasti, ki so bile ustvarjene namerno ali pa tudi ne, se lahko ujamejo spletni pajki med preiskovanjem ene od omenjenih strani. Gre za neskončno avtomatsko generiranje novih povezav, ki jih mora pajek obiskati. Ker pajek vedno znova dobiva nove spletne strani, nikoli ne zaključi s preiskovanjem in tako obtiči na strani.

Eden od načinov za realizacijo pasti je s pomočjo dodajanja poizvedb oziroma parametrov v sam URL (ang. query strings). Po navadi se pojavijo pri straneh s spletnimi iskalniki. Za primer vzemimo povezavo na strani Google, https://www.google.si/?q=web+crawler. Vse, kar je v naslovu napisano

za vprašajem, so podani parametri. V tem primeru "q"označuje parameter, poizvedbo (ang. query), vse naprej pa so ključne besede, ločene z znakom "+". Ker ta povezava ni dejanski naslov, ki kaže na obstoječo spletno stran, temveč je le domena ("www.google.si") z dodanimi poljubnimi parametri, je možnosti za ustvarjanje novih povezav praktično neomejeno.

Nekatere pasti so narejene tako, da se k povezavi stalno dodajajo novi imeniki, na primer www.example.com/images/documents/images/documents/... Take pasti so večinoma ustvarjene namerno, torej, da ustavijo pajke, ki pridejo na to stran.

Obstajajo tudi pasti, do katerih pride čisto naključno. Najbolj pogost primer je spletna stran, na kateri je koledar. Nekateri koledarji imajo samodejno generirane povezave na prejšnji in naslednji mesec na vsaki strani. Zaradi konstantnega sledenja novim povezavam se lahko pajek recimo znajde na koledarju za leto 3000.

Čisto vsem pastem se je praktično nemogoče izogniti. Obstajajo pa nekateri načini, ki lahko težavo vsaj omilijo. Obstajajo tudi precej enostavne rešitve, ki vsaj delno pripomorejo. Pajek je lahko napisan tako, da ima omejen čas preiskovanja za vsako spletno stran. Po pretečenem času enostavno začne preiskovati naslednjo stran na seznamu povezav. S tem zagotovimo, da pajek ne more obtičati v pasti za dlje časa, kot je čas, ki ga določimo sami.

Še eden od možnih načinov je odstranjevanje parametrov pri naslovih strani, s čimer preprečimo eno izmed možnih pasti. Na primer, namesto www.google.si/q=example, dodamo na seznam povezav v podatkovni bazi le www.google.si.

Poglavje 5

Zajemanje podatkov

Pridobitev podatkov je osnovni cilj vsakega spletnega pajka. Tip vsebine, ki nas zanima, pa se razlikuje od enega do drugega, odvisno od njihovega namena. Na primer pri pajku, katerega naloga je pridobivanje podatkov za spletni iskalnik, je najpomembnejša vsebina besedilo na spletnih straneh in njihove ključne besede, saj se v večini primerov, ko uporabnik vpiše besedno zvezo v iskalnik, le-ta nanaša na besedilo s spletnih strani.

5.1 Povezave

Tip podatkov, ki je pomemben za vsakega pajka, ne glede na njegov namen, so povezave. Osnovni princip delovanja spletnega pajka je namreč izluščiti vse povezave z začetne strani na druge spletne strani, slediti tem povezavam, shraniti obiskane strani in ponovno preiskovanje novo odkritih strani.

Razčlenjevanje vsebine strani je mogoče na dva načina. Pri prvem se stran – dokument le shrani, kasneje pa ga kateri drugi proces ali nit razčleni na elemente. Pri drugem načinu se vse dogaja sproti, torej takoj, ko se stran odpre, se tudi obdela. Prva naloga je preiskati odprto stran in izluščiti vse povezave na njej. To naredimo s pomočjo razčlenjevalnika (ang. parser), ki razčleni dokument HTML na posamezne elemente. V tem primeru nas zanimajo vsi elementi z atributom "href"(ang. hypertext reference), saj je s

tem ustvarjena povezava na podan naslov. S tem pridobimo vse povezave, ki se skrivajo na trenutni strani, vendar v našem primeru niso vse povezave, ki jih dobimo, tudi veljavne. Namesto na določeno spletno stran lahko kaže tudi na kaj drugega, recimo sliko: "www.hostname.com/image.jpg"ali pa na dokument "file.pdf"in podobno. V takih primerih pajek povezave ne more odpreti. Potrebno je zagotoviti, da ne pride do napak, če neka povezava ne kaže na spletno stran.

Preden vsako novo povezavo dodamo na seznam čakajočih naslovov, je potrebno preveriti, če je ta povezava že na katerem od seznamov. Težave se pojavijo, ko so naslovi nekonsistentni. Na primer, tako http://www.google.com kot tudi HTTP://www.google.com sta veljavni povezavi, ki sicer kažeta na isto stran, vendar niza nista enaka. Kljub temu da je prva povezava že v seznamu čakajočih, se bo tudi druga dodala na naš seznam, kar je vzrok za večkratno preiskovanje iste spletne strani. Vedno, preden se katera od novo odkritih povezav doda na naš seznam, jih je potrebno poenotiti. Tukaj je nekaj korakov, ki nam pomagajo preprečiti večkratno preiskovanje istih strani [10]:

- Pretvori protokol (HTTP) in naslov strani v male črke, HTTP://www.NAJDI.si se pretvori v http://www.najdi.si.

- Odstrani sidro oziroma referenco na del strani, stran http://www.spletnastran.si/domov.html/#element se zmanjša na http://www.spletnastran.si/domov.html

- Opravi znakovno pretvorbo pogostih simbolov. S tem preprečimo, da bi pajek obravnaval http://www.spyder.edu/files/$\tilde{}$file/ drugače kot http://www.spyder.edu/files/%7file/

- Dodaj oziroma odstrani "/"na koncu naslova, tako da je pri vseh povezavah enako. Namesto http://www.google.com/ je sedaj http://www.google.com.

5.2 Ključne besede

Zajemanje oziroma odkrivanje ključnih besed je del procesa indeksiranja strani in spada predvsem v področje spletnih iskalnikov. Spletni pajki v tem primeru ne iščejo ključnih besed direktno, temveč najprej shranijo kar celotno stran, pozneje pa se vsaka prenesena stran razčleni in opravi se analiza besedila. Vsak razčlenjevalnik, ki se uporablja za razčlenjevanje najrazličnejših spletnih strani na celotnem spletu, mora biti spisan tako, da je sposoben obvladovati ogromno število možnih napak. Razčlenjevalnik lahko izlušči relevantne podatke iz spletne strani z izključevanjem pogostih besed (po navadi "a", "an", "the"v angleščini), oznakHTML, kode javascript in še nekatere druge nam neuporabne nize [12]. Dober razčlenjevalnik lahko tudi odstrani elemente, ki se pogosto pojavijo na spletnih straneh, a za nas niso zanimivi, kot so navigacijske povezave, tako da se ne upoštevajo kot del vsebine strani. Pri iskanju ključnih besed in določevanju pomembnosti posameznih besed se upošteva več faktorjev, med drugim:

- gostota ključnih besed,
- položaj besed (naslovi, meta oznake),
- teža besed (krepko, ležeče, podčrtano),
- velikost besed.

Po končanem indeksiranju se rezultati shranijo v vnaprej določenem vrstnem redu, kar omogoča hitrejše pridobivanje informacij. Indeksi se periodično posodabljajo, ko se preiščejo nove strani ali ponovno preiščejo že preiskane. Velik del uspešnosti spletnih iskalnikov leži prav v tem, na kakšen način so zgrajeni indeksi in kako se uporabljajo [12]. Različni algoritmi se uporabljajo za optimizacijo teh indeksov, tako da se pomembni podatki najdejo zelo enostavno, z minimalno uporabo računalnikovih virov [13].

5.3 Druga vsebina

Na povprečni spletni strani najdemo vrsto različnih vsebin, ki so potencialno lahko cilj spletnih pajkov. Od določenih nizov v besedilu, telefonskih številk in elektronskih naslovov do slik ali drugih tipov datotek, na primer "pdf"ali "doc". V vsakem primeru je potrebno želeno vsebino najprej odkriti. Preiskovanje oziroma razčlenjevanje strani je mogoče opraviti na več načinov.

Prvi način je razčlenjevanje spletne strani s pomočjo HTML-ja. Kodo HTML na vsaki spletni strani je možno predstaviti kot drevesno strukturo DOM (Document Object Model). DOM je standard za pridobivanje, spreminjanje, dodajanje in brisanje elementov HTML. S pomočjo te strukture oziroma standarda lahko dostopamo do posameznih delov spletne strani brez večjih težav in preiščemo tiste elemente, ki nas zanimajo.

Pri drugem načinu gre za obravnavanje celotne spletne strani kot besedilo. Ta metoda se imenuje ujemanje vzorcev (ang. pattern matching) [14]. Obstaja več načinov izvedbe te metode, osredotočili se bomo na ujemanje vzorcev s pomočjo regularnih izrazov. Regularni izraz je niz znakov, ki opisuje druge nize znakov (ang. string) z določenimi sintaksnimi pravili. Vsak znak v regularnem izrazu se obravnava kot "metaznak", ki lahko označuje vrsto drugih znakov oziroma njihovo zaporedje, ali pa navaden znak s svojim dobesednim pomenom. Skupaj se jih lahko uporablja za identifikacijo določenih nizov znotraj besedila. V tabeli 5.1 je opisanih nekaj najpogostejših regularnih izrazov.

Ko določimo naš regularni izraz, se besedilo, ki ga želimo preiskati (v tem primeru celotna spletna stran), preda procesorju regularnih izrazov skupaj s samim izrazom in začne se proces ujemanja vzorcev. Ob koncu nam procesor vrne seznam besed, ki so v podanem besedilu in se ujemajo z izbranim regularnim izrazom. Nato se vsa najdena vsebina shrani v podatkovno bazo, skupaj z naslovom spletne strani, na kateri je bila odkrita.

[in]	začetek in konec niza	
a	katerikoli posamezen znak ([a])	”a”
-	razpon znakov ([a-z] ali [0-9])	”a”, ”b”, ”0”, ”1”
\	ubežni znak	”.”
\w	vsak besedni znak	”a”, ”b”
\s	presledek	
\S	vsak znak, ki ni presledek	”a”, ”b”, ”1”
\d	vsaka decimalna številka	”0”, ”1”, ”2”

Tabela 5.1: Tabela pogostih regularnih izrazov.

Poglavje 6

Implementacija

Poleg teoretičnega dela spletnih pajkov spada k diplomskemu delu tudi praktični - implementacija lastnega spletnega pajka. V tem poglavju je opisano, kako pajek deluje, kaj so njegovi sestavni deli in podobno.

6.1 Cilji

Cilj je implementirati delujočo različico spletnega pajka v programskem jeziku Java. Poleg osnovnih nalog vsakega spletnega pajka, torej iskanje vseh povezav na podani spletni strani in sledenje tem povezavam, je namen tega programa iskanje in shranjevanje vseh elektronskih naslovov, ki jih odkrije.

6.2 Sestavni deli

V nadaljevanju so našteti in opisani vsi pomembnejši razredi in njihove metode ter algoritem, ki so potrebni za delovanje našega pajka.

6.2.1 Uporabljen algoritem

Algorithm 1 Algoritem, uporabljen pri implementaciji spletnega pajka.

```
1: procedure Crawl(seed URL):   //začni s proceduro, podan začetni URL
2:     insert seed URL into Frontier;   //dodaj začetni URL na seznam
3:     WHILE Frontier is not empty      //preiskovalna zanka
4:         link = get first URL from Frontier;      //izberi URL
5:         webPage = open page (link);      //odpri pripadajočo stran
6:         newLinks = extract links (webPage);      //izlušči povezave
7:         emails = find emails (webPage);      //najdi elektronske naslove
8:         insert into Frontier (newLinks);      //dodaj URL-je na seznam
9:         remove from Frontier (link);      //odstrani začetni URL s seznama
10: end WHILE      //konec zanke
```

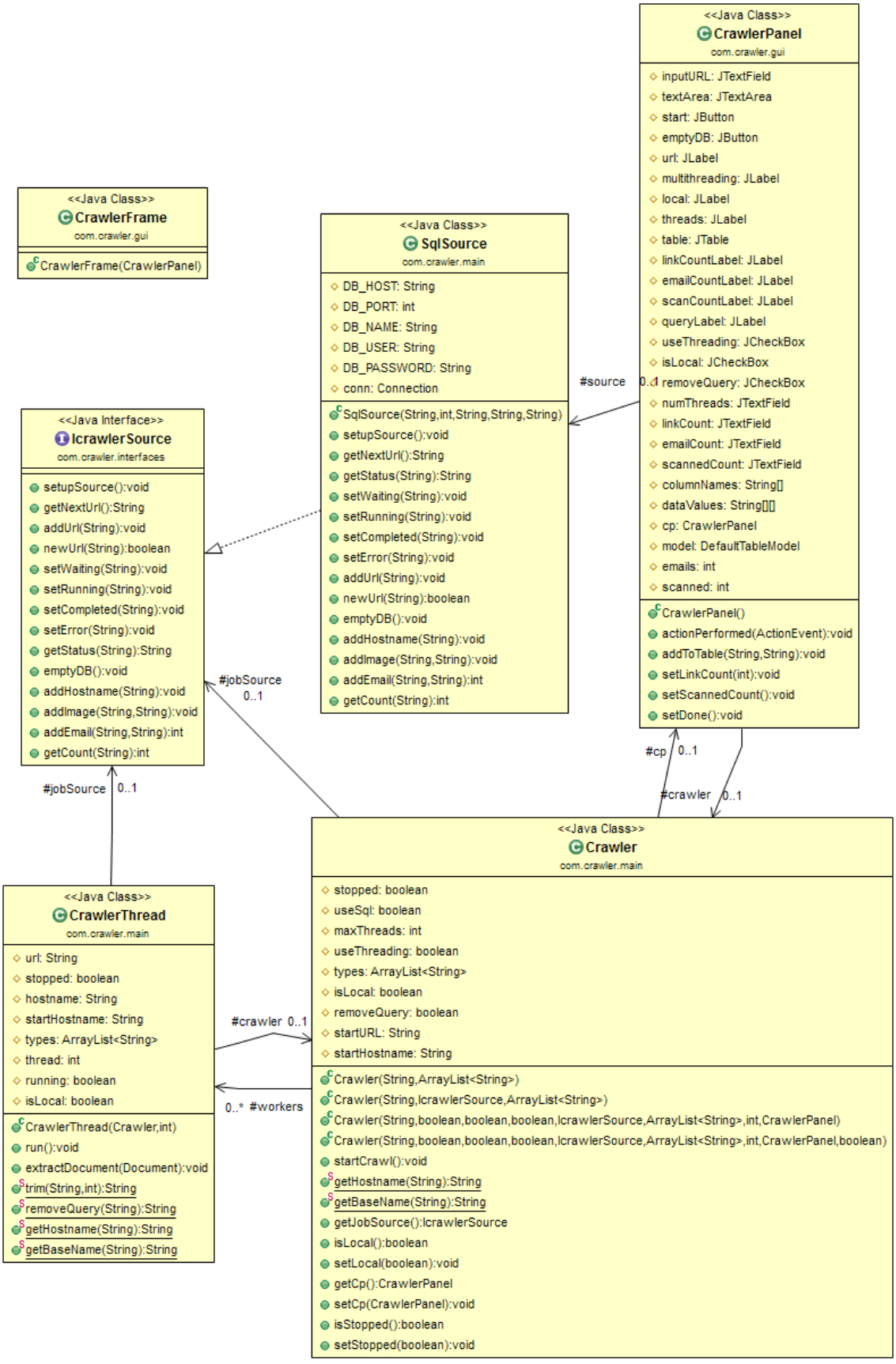
Algoritem 1 se začne s podanim začetnim URL-jem, ki se doda na seznam povezav. V vsakem ciklu zanke se najprej pridobi nov URL s seznama povezav, nato se prenese spletna stran, ki ji pripada pridobljeni URL. Prenesena stran se razčleni in shranijo se vse odkrite povezave in elektronski naslovi. Nove povezave dodamo na seznam povezav. Ko se stran obdela, se iz seznama povezav odstrani prvi URL, ki je bil uporabljen. Zanka se izvaja, dokler seznam povezav ni prazen.

6.2.2 SQLSource.java

Razred, katerega glavna naloga je priskrbeti pajku naslednjo ustrezno povezavo, ki jo je potrebno preiskati. Obenem skrbi tudi za vso interakcijo s podatkovno bazo SQL.

Pomembne metode

public SqlSource(String host, int port, String name, String user, String password); Konstruktor razreda SQLSource. Za kreiranje pove-



Slika 6.1: Razredni diagram implementiranega pajka.

zave do podatkovne baze je potrebno podati gostitelja baze, številko vrat, ime podatkovne baze ter uporabniško ime in geslo za dostop do baze.

public void setupSource(); Poskrbi za kreiranje povezavnega niza (ang. connection string) v obliki jdbc:mysql://"+DB_HOST+":"+DB_PORT+"/"+DB_NAME; iz parametrov, podanih v konstruktorju, ki se nato uporabi za ustvarjanje povezave z bazo.

public synchronized String getNextUrl(); Poišče in vrne naslednjo povezavo iz podatkovne baze, ki jo potem preišče pajek oziroma ena od njegovih niti. To doseže tako, da vrne naslednji naslov strani, ki ima status "waiting"v seznamu povezav z SQL stavkom SELECT * FROM queue WHERE status = 'W'.

public boolean newUrl(String url); in public void addUrl(String url); Skrbita za dodajanje vseh novih povezav, ki jih pajek najde na neki strani, pod pogojem, da podani naslov strani še ne obstaja v seznamu povezav. Če se povezava že nahaja znotraj omenjenega seznama, torej če metoda "newUrl"vrne "false", se podan naslov enostavno preskoči in se ne doda še enkrat na seznam. S tem zagotovimo, da se ista spletna stran ne preišče večkrat.

public void setWaiting(String url), setRunning(String url), setCompleted(String url), setError(String url); Metode za nastavljanje trenutnega stanja podane povezave. Možna stanja so *waiting*, *running*, *completed* in *error*, ki so obrazložena v poglavju 1.

6.2.3 CrawlerThread.java

Deduje razred Thread, s tem omogočimo izvajanje dela programa v več nitih. V tem razredu se opravlja najpomembnejši del pajka, torej samo preiskovanje posameznih strani. Med drugim sem spada pridobitev novega naslova strani,

vzpostavljanje povezave in prenos te strani ter razčlenjevanje dobljenega dokumenta HTML, s čimer pridobimo nove povezave in elektronske naslove.

Pomembne metode

public CrawlerThread(Crawler crawler, int thread); Konstruktor s parametroma *Crawler crawler* in *int thread*. Crawler oziroma pajek je glavni program, h kateremu pripadajo vse niti. S tem je omogočena komunikacija med nitjo in glavnim programom. Na primer, ko je odkrit nov elektronski naslov, se obvesti program, da ga lahko ustrezno prikaže na grafičnem vmesniku. S parametrom thread oziroma nit podamo število niti, ki jih želimo uporabiti.

public void run(); Metoda, ki jo je potrebno implementirati vsakič, ko dedujemo razred Thread. Na tem mestu se izvaja vse, kar se lahko izvaja ločeno na posameznih nitih. Vsebuje zanko, ki se izvaja, vse dokler so na voljo naslovi za enkrat še nepreiskanih spletnih strani. Ko jih zmanjka, se izvajanje zanke ustavi. V vsakem obhodu zanke se najprej pridobi nov naslov URL, pajek potem preišče stran s tem naslovom ter poišče in shrani vse najdene povezave znotraj dokumenta. Na koncu je potrebno še odkriti vse elektronske naslove, ki se pojavijo kjerkoli na strani, kar storimo s pomočjo regularnih izrazov (ang. regular expressions). Regularni izraz, ki je uporabljen v našem pajku:

```
"[a-zA-Z0-9\._%+-]+(\s*@\s*|\s*[\[|\{|\(]+
\s*(AT|at|@)\s*[\)|\}\]]+\s*)([a-zA-Z0-9\.-]+
(\.|\s*[\[|\{|\(]+\s*(DOT|dot|\.)\s*[\)|\}|\]]+\s*))+
[a-z]{2,6}".
```

Običajen regularni izraz za odkrivanje elektronskih naslovov je sicer precej bolj enostaven, pri našem pajku pa želimo odkriti tudi vse tiste naslove, ki so na nek način skriti, oziroma jih nek uporabnik želi prikriti tako, da elektronski naslov napiše nekoliko drugače. Na primer, namesto info@gmail.com

lahko napiše info (at) gmail (dot) com. Naš pajek odkrije tudi take, na videz skrite elektronske naslove. Še nekaj primerov prikritih naslovov, ki jih lahko odkrije:

- example@email.com
- example @ email.com
- example@email dot com
- example at email dot com
- example(AT)email(DOT)com
- example (at) email.com
- example [at] email [dot] com

6.2.4 Crawler.java

Glavni razred pajka, ki na enem mestu združi ostale, pomožne razrede tako, da dobimo delujočega spletnega pajka.

Pomembne metode

public Crawler(String startURL, boolean isLocal, boolean useThreading, IcrawlerSource jobSource, int maxThreads, CrawlerPanel cp); Konstruktor za kreiranje spletnega pajka. Potrebno je podati naslednje parametre:

1. String startURL: naslov strani, na kateri pajek začne svoje preiskovanje.
2. Boolean isLocal: določa lokalnost (preiskuje le strani na podani domeni) oziroma globalnost (preiskuje vse odkrite strani) pajka.

3. useThreading: omogoča izvajanje programa v večnitnem (ang. multithreading) načinu.
4. IcrawlerSource jobSource: objekt, ki skrbi za seznam povezav in interakcijo s podatkovno bazo.
5. Int maxThreads: število niti, ki jih program lahko uporablja, če je useThreading omogočen.
6. CrawlerPanel cp: objekt, ki omogoča komunikacijo pajka z grafičnim vmesnikom.

public void startCrawl(); Ustvari in požene toliko niti, kolikor smo jih določili s parametrom "maxThreads", oziroma le eno, če je parameter "useThreading"onemogočen. S tem se začne preiskovanje prve podane strani.

6.2.5 CrawlerPanel.java

Skrbi za grafični vmesnik programa in vso uporabnikovo interakcijo s pajkom. Sproti izpisuje seznam najdenih elektronskih naslovov, njihovo število, število vseh trenutno odkritih povezav in število strani, ki jih je pajek že pregledal.

6.3 Lastnosti

V nadaljevanju je naštetih nekaj najpomembnejših lastnosti implementiranega spletnega pajka:

- Istih strani ne preiskuje večkrat.
- Omogoča večnitno izvajanje.
- Uporablja bazo SQL za shranjevanje seznama povezav in najdenih elektronskih naslovov.
- Omogoča lokalno in globalno preiskovanje.
- Izogiba se pastem z odstranjevanjem poizvedb iz naslova.

6.4 Delovanje

Delovanje implementiranega pajka lahko demonstriramo v devetih korakih:

1. Uporabnik požene Email Crawler 1.0.

2. V polje na vrhu programa vpiše naslov strani, na kateri pajek začne s preiskovanjem. Potrebno je vpisati celoten naslov, vključno s shemo spredaj, na primer http://www.google.com.

3. Uporabnik izbere še nekatere možnosti. Če je izbrana možnost *Search Locally*, bo spletni pajek sledil le vsem stranem, ki se nahajajo na podani domeni. Sicer bo program sledil vsaki povezavi, ki jo najde, in preiskal vse odkrite spletne strani.

4. Z možnostjo *Use Multithreading* omogočimo izvajanje programa v več nitih. Pojavi se okno, v katerega vpišemo število niti, ki jih želimo pri preiskovanju. Večnitno izvajanje pajka poteka tako, da vsaka od niti obišče in preišče svojo spletno stran. S tem se občutno zmanjšajo zakasnitve zaradi čakanja na odgovor strani.

5. S pritiskom na gumb *Empty Database* se pobriše vse, kar je trenutno shranjeno v podatkovni bazi. To je potrebno v primeru, da je bilo kakšno preiskovanje že predhodno izvedeno.

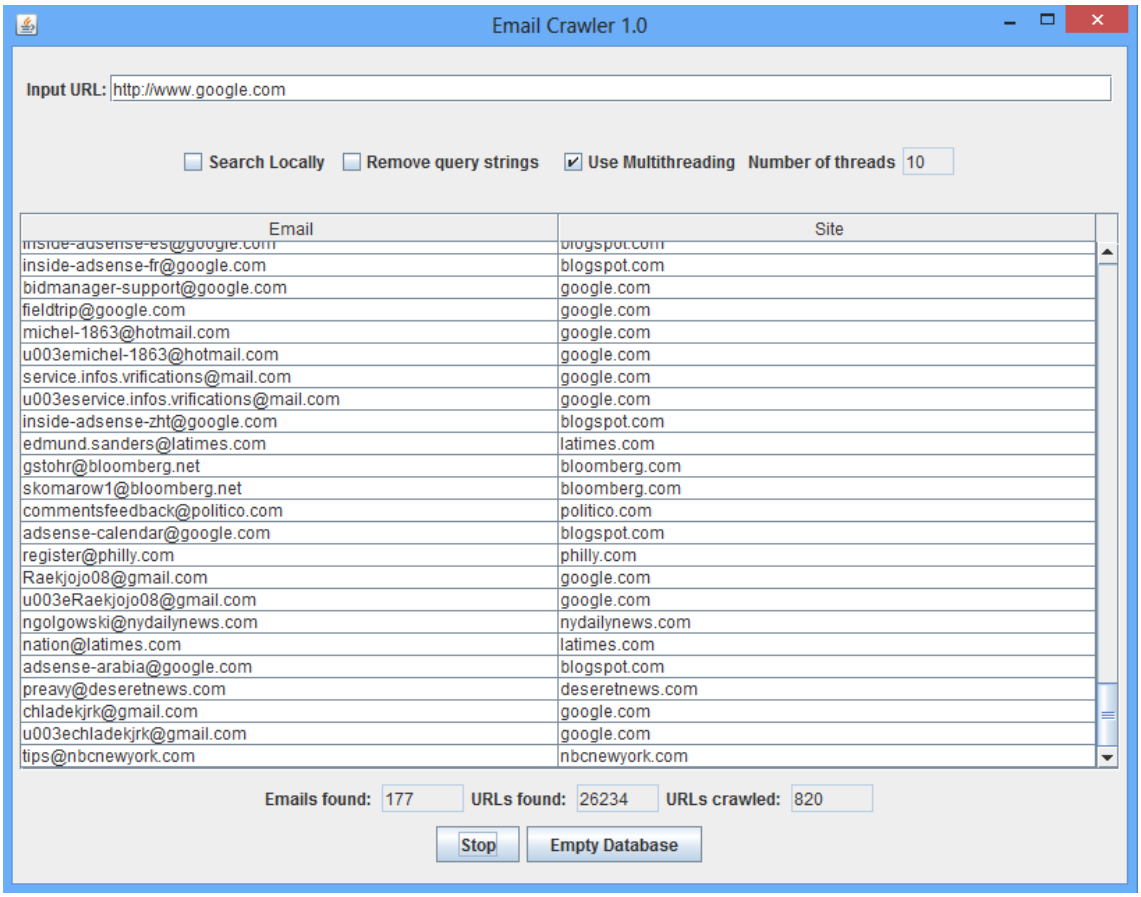
6. Gumb *Start!* začne izvajanje našega spletnega pajka. Naslov, ki je bil podan, se najprej shrani na seznam povezav, ki je v podatkovni bazi. Ustvari in požene se toliko niti, kolikor smo jih določili, sicer se ustvari le ena nit. Vsaka od njih pošlje zahtevo za pridobitev naslova strani s seznama povezav. Ker je na začetku izvajanja na voljo le ena povezava (tista, ki jo je vpisal uporabnik), se lahko v prvem obhodu zanke preiskovanje izvaja le v eni niti. Ostale čakajo, dokler se v bazo ne dodajo nove povezave. V večini primerov morajo niti na povezavo čakati le, dokler se ne konča preiskovanje prve strani. Takrat se v bazo doda dovolj povezav za vse nadaljnje preiskovanje, saj se novo odkriti

naslovi strani dodajajo v bazo precej hitreje, kot je samo obdelovanje strani.

7. Na vmesniku se sproti izpisujejo vsi elektronski naslovi, ki jih pajek odkrije med preiskovanjem, ter stran oziroma domena, na kateri je bil naslov najden. V primeru, da je omogočeno lokalno iskanje, bodo vsi elektronski naslovi le z domene, ki jo je vpisal uporabnik.

8. Na spodnjem delu se ažurirano izpisujejo nekateri podatki. Število do sedaj odkritih elektronskih naslovov, število povezav, ki jih je pajek do sedaj odkril, ter število že preiskanih strani.

9. S pritiskom na gumb *Stop* se preiskovanje pajka zaustavi.



Slika 6.2: Implementirani pajek.

Poglavje 7

Zaključek

Večina uporabnikov na internetu se spletnih pajkov sploh ne zaveda, kljub temu da jim ti pravzaprav omogočajo brskanje po neštetih spletnih straneh. Čeprav se le malo ljudi zaveda njihovega obstoja, njihova vloga vseeno ni zanemarljiva. Zaradi trenutne velikosti spleta, hitrosti, s katero se vsakodnevno veča in konstantnega spreminjanja vsebin spletnih strani, bodo imeli pajki v prihodnosti vse večji pomen. Uporabljeni bodo v bolj raznolike namene, zaradi česar bo tudi vse več različnih tipov spletnih pajkov, ki bodo služili posameznim aplikacijam.

V uvodu je predstavljeno, kje vse se spletni pajki uporabljajo in kakšna je njihova vloga. Videli smo, da je najpomembnejša in daleč najbolj popularna uporaba spletnih pajkov preiskovanje interneta in ustvarjanje repozitorijev spletnih strani, ki jih potem uporabljajo spletni iskalniki. Vse bolj pogosti bodo pajki, katerih naloga je iskanje specifičnih podatkov.

V nadaljevanju smo govorili o osnovnem delovanju in o sami strukturi pajka. Osnovni algoritem spletnih pajkov je še posebej pomemben zato, ker je praktično enak pri vsakem pajku, ne glede na to, kakšna je njegova naloga. Odločiti se je potrebno med rekurzivno in iterativno različico, vendar je v večini primerov iterativna različica precej bolj primerna, še posebej pri bolj obširnih preiskovanjih. Pri iterativnem pajku je predstavljen eden izmed načinov shranjevanja povezav z več seznami.

Nato se seznanimo z arhitekturo pajkov. Najprej, kakšna je lahko porazdeljenost spletnih pajkov, nato pa še nekaj najpogostejših tipov spletnih pajkov, ki se razlikujejo po načinu preiskovanja. Na koncu so opisane še podatkovne strukture, ki so potrebne za učinkovito delovanje pajkov.

Sledijo vse funkcionalnosti, ki jih mora pajek običajno imeti oziroma upoštevati. Še posebej pomembno je, da je pajek prijazen do spletnih strani, ki jih preiskuje, torej čim manjša poraba pasovne širine in omejeno število istočasnih povezav do te strani. Omeniti je potrebno še hitrost in robustnost, ki sta bistveni lastnosti vsakega dobrega spletnega pajka. Hitrost zaradi ogromnega volumna strani, ki jih je potrebno preiskati, robustnost pa zaradi nemotenega delovanja.

Nato je opisano, kateri podatki so po navadi cilj spletnih pajkov in na kakšen način lahko pajek te podatke odkrije in shranjuje. Zanimiv je predvsem odsek o zajemanju povezav, saj je značilen za vse spletne pajke.

Na koncu smo implementirali sistem, katerega naloga je iskanje in shranjevanje vseh elektronskih naslovov, ki jih odkrije na podani spletni strani in na vseh drugih straneh, ki so povezane z začetno stranjo. Od posameznih sestavnih delov programa do samih metod, ki so pomembne za delovanje pajka. V nadaljevanju so opisane še nekatere lastnosti implementiranega pajka, nato pa je po posameznih korakih razdelano še njegovo delovanje, od vnosa začetne spletne strani, do končanega preiskovanja. Kljub temu, da se uporabljajo za najrazličnejše namene in zbirajo podatke v različnih oblikah, je osnova vsakega pajka enaka, torej začetek iskanja na podani strani, iskanje povezav in nadaljevanje preiskovanja na novih straneh. Večja razlika je le pri podatkih, ki nas zanimajo. Kot smo videli v poglavju 6, je v tem diplomskem delu implementiran pajek, ki išče in shranjuje elektronske naslove. Če so podatki, ki nas zanimajo precej podobni, lahko pajka precej hitro spremenimo tako, da išče kakšne druge nize v besedilu. Potrebno bi bilo le spremeniti regularni izraz, ki je bil prej uporabljen za prepoznavanje elektronskih naslovov. Seveda je odvisno tudi, kakšna vsebina spletnih strani nas zanima, prilagajanje programa je lahko precej bolj težavno.

Kako torej implementirati učinkovitega pajka? Na prvem mestu je hitrost preiskovanja. Kot smo že omenili v tem diplomskem delu, delo spletnega pajka praktično nikoli ni končano, zato je potrebno implementirati večnitnost ali program izvajati na več različnih sistemih (distribuirani pajek). Pri samem preiskovanju je pomembna izbira algoritma. V praksi to pomeni, da algoritem določa vrstni red preiskovanja spletnih strani. Če upoštevamo kompleksnost algoritma v razmerju z njegovo učinkovitostjo, se največkrat priporoča pajek s preiskovanjem v širino. Zaradi volumna URL-jev, ki jih je potrebno hraniti, lahko za shranjevanje povezav le redko uporabimo delovni spomin računalnika. Vedno je priporočena uporaba baze SQL. Poleg tega je potrebno tudi paziti na pasti, v katere se lahko ujame naš spletni pajek. Nekatere so bile ustvarjene namenoma, druge čisto slučajno, v vsakem primeru pa se jim je potrebno izogniti. Na koncu ne smemo pozabiti na same spletne strani. Poskrbeti moramo, da ne škodimo nobeni izmed strani, ki jih preiskujemo. V dobi, ko imamo na spletu na voljo obilico informacij, a premalo časa, da bi sami lahko vse preiskali, so spletni pajki dobrodošla rešitev. Za večino populacije so sicer skriti, vseeno pa za nas lahko opravijo ogromno dela, če jih dobro implementiramo.

Literatura

[1] M. Najorc. Web crawler acrhitecture. Dostopno na: http://research.microsoft.com/pubs/102936/eds-webcrawlerarchitecture.pdf

[2] N. Padua, B. Pugh. Synchronization in Java. Dostopno na: http://www.cs.umd.edu/class/spring2006/cmsc132/Slides/lec35.pdf

[3] C. D. Manning. Introduction to information retrieval. Cambridge university press, 2008.

[4] D. Khurana, S. Kumar. Web Crawler: A Review. IJCSMS, 2012. Dostopno na: http://www.ijcsms.com/journals/Volume%2012, %20Issue%2001,%20January%202012_WEB_CRAWLER1.pdf

[5] Web Crawling and Indexes. Cambridge University Press, 2009. Dostopno na:http://nlp.stanford.edu/IR-book/pdf/20crawl.pdf

[6] M. Koster, Standard for Robots Exclusion, 1996. Dostopno na http://www.robotstxt.org/eval.html

[7] S. Kumar.Design and Implementation of Scalable, Fully Distributed Web Crawler for a Web Search Engine. International Journal of Computer Applications Vol. 17, februar 2011. Dostopno na: http://www.ijcaonline.org/volume15/number7/pxc3872629.pdf

[8] J. Heaton. Programming Spiders, Bots and Aggregators in Java. Sybex, 2002.

[9] C. Olson, M. Najorc. Web Crawling. Foundations and Trends in Information Retrieval No. 3, 2010. Dostopno na: http://homepages.dcc.ufmg.br/ñivio/cursos/ri12/transp/olston-najork@web-crawling10.pdf

[10] G. Pant, P. Srinivasan, F. Menczer. Crawling the Web. Indiana University Press. Dostopno na: http://informatics.indiana.edu/fil/Papers/crawling.pdf

[11] C. Franklin. How Internet Search Engines Work, 2002.

[12] M. Peshave How Search Engines Work. University of Illinois. Dostopno na: http://citeseerx.ist.psu.edu/viewdoc/ download?doi=10.1.1.133.6059&rep=rep1&type=pdf

[13] S. Chakrabarti. Mining the Web: Analysis of Hypertext and Semi Structured Data, 2003.

[14] Wikipedija, Regular expressions. Dostopno na: http://en.wikipedia.org/wiki/Regular_expression

[15] J. Cho. Crawling the Web: Discovery and Maintenance of Large Scale Data. November 2001. Dostopno na: http://oak.cs.ucla.edu/ cho/papers/cho-thesis.pdf

[16] The size of the world wide web. Dostopno na: http://www.worldwidewebsize.com/