Forside

Titel

Elev Navn

Vejleder Navn

Dato

Evt. Billede

Indholdsfortegnelse

indholdsfortegnelse

"Jeg bekræfter herved med min underskrift, at opgavebesvarelsen er udarbejdet af mig. Jeg har ikke anvendt tidligere bedømt arbejde uden henvisning hertil, og opgavebesvarelsen er udfærdiget uden anvendelse af uretmæssig hjælp og uden brug af hjælpemidler, der ikke har været tilladt under prøven."

A close-up of a signature

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Indholdsfortegnelse

Contents

[Indledning 4](#_Toc166336651)

[Problemformulering 4](#_Toc166336652)

[Programbeskrivelse 4](#_Toc166336653)

[Kodedokumentation 5](#_Toc166336654)

[fGetArticleInfo 5](#_Toc166336655)

[fGetLinks 7](#_Toc166336656)

[fSortList 8](#_Toc166336657)

[fGetPageTitle 8](#_Toc166336658)

[funcStartTimer og funcStopTimer 9](#_Toc166336659)

[Test af programmet 9](#_Toc166336660)

[Konklusion 9](#_Toc166336661)

[Kilder 9](#_Toc166336662)

[Bilag 9](#_Toc166336663)

# Indledning

Vi har alle lyst til at vide en masse om mange forskellige ting. Hvordan gør vi det? Ved at læse selvfølgelig! Men hvad skal du læse, og hvad er det faktisk du gerne vil vide mere om? Det er ligegyldigt bare du ved en masse! Det er her vi har en løsning! Lad mig introducere ✨Wikipedia Speedrun✨[[1]](#footnote-1). Dette fantastiske lille spil handler om at vide en masse, mens man samtidig bruger og udvikler sin logiske tankegang for at kunne navigere rundt i det store hav af 6,8 millioner Wikipedia artikler[[2]](#footnote-2). Det handler om at kunne hurtigst muligt navigere fra en artikel til en anden gennem de links der er på artiklen. Når du så er blevet mester til ✨Wikipedia Speedrun✨, og har brug for en større udfordring, eller bare ikke har lyst til at læse artiklen for at finde linkene, så kan du prøve det nye og forbedrede ✨Wikipedia Speedrun lite pro+✨. I denne forbedrede version bliver man kun givet linkene til artiklen med ingen kontekst overhovedet! Dette giver en mere bare bones oplevelse hvor brugeren selv er nødt til at kende indholdet i artiklen.

*\*Spillet køres kun i terminalen og en terminal som kan vise 1000+ linjer sælges separat.*

# Problemformulering

Så hvordan vil Wikipedia Speedrun lite pro+også kaldt Wikipedia speedrun i terminalen udformes. For at lave spillet er der to hovedopgaver som vil skulle laves, 1) vi skal kunne hente og sortere den relevante data fra Wikipedia, og 2) vi skal lave ”reglerne” for spillet, hvordan man navigere mellem sider, hvornår man har vundet osv. Her blev jeg ansvarlig for den første del, altså at hente og sortere dataene. Så der er to opgaver.

1. Hvordan henter man dataene fra Wikipedia?
2. Hvordan sortere man den data for at få noget der kan bruges til vores formål?

Andre ting som det også skal findes ud af er.

1. Hvordan kan man sørge for at front end developer let kan bruge funktionerne?
2. Hvordan skal der findes to tilfældige Wikipedia artikler som skal bruges til start og slut?

# Programbeskrivelse

For at få en brugbart spil skal vi bruge visse ting. Så for at spillet fungere skal der være visse kriterier som skal opfyldes, her i blandt som tidligere nævnt, navigering mellem artikler, det at hente og sortere data fra Wikipedia. Så derfor er der opstillet en MVP som set under.

**Need**

* Programmet skal kunne finde to tilfældige sider.
* Programmet skal kunne vide hvornår brugeren har nået den designeret mål artikel.
* Programmet skal kunne hente data fra en specifik Wikipedia artikel.
* Programmet skal kunne finde alle links i en Wikipedia artikel
* Programmet skal kunne vise alle links på en given Wikipedia artikel.
* Brugeren skal kunne navigere mellem artikler for at finde målet.

**Nice**

* En main menu.
* Timer til at finde tiden brugt til at nå i mål.
* Programmet skal kunne gemme valgene som brugeren har taget.
* Brugeren skal kunne vælge egen start og slut artikel.
* Programmet skal kunne vide om der reelt set er vej mellem de to artikler.

**Good**

* Programmet skal kunne finde kortest mulig vej mellem to artikler.

Siden, som tidligere nævnt, fungerer spillet kun gennem en text-based user interface/en terminal betyder det er UI’et er relativt simpelt. Nå programmet startes op. Vil man blive mødt med en simpel main menu. Her vil brugeren have 5 valg.

1. Brugeren kan starte et nyt spil, som vælger 2 fuldkommen tilfældige artikler.
2. Brugeren vælger egen start og slut artikel, og derefter starter spillet.
3. Brugeren kan vælge hvor mange artikler, der skal være mellem start og slut, og derefter starter spillet.
4. Brugeren kan se hvilke valg der blev truffet i sidste spil.

Q) Lukke programmet. Her fungerer Q også som knappen for at gå tilbage til main menu.

A screenshot of a computer program

Description automatically generated

Figur 1 Main menu for spillet.

Alle gamemodes fungerer ligeledes efter spillet er startet. Når spillet starter, bliver der vist tre ting, en timer, start og slut artiklerne og så alle tilgængelige links på nuværende artikel. Selvom der kun bliver vist 5 muligheder på figur 2, så er der i alt 192 links på artiklen.

A screenshot of a computer

Description automatically generated

Figur 2 UI når spillet er i gang.

For flowdiagram for hele spillet referer til bilag 1.

# Kodedokumentation

## fGetArticleInfo

fGetArticleInfo er inddelt i tre forskellige segmenter, med tre forskellige if-statements som bestemmer, hvilken gamemode det er den skal hente data til. De tre gamemodes er navngivet ”connected”, ”random” og list. Dette svarer til de tre gamemodes nævnt tidligere. De tre if-statements er lavet så Christian/front end developer bare kan kalde funktionen med en af tre valgmuligheder. De kan kalde fGetArticleInfo(”connected”, iterations), hvor iterations er antal artikler mellem start og slut, fGetArticleInfo(”random”), for at blive givet to givet to fuldkommen tilfældige artikler, og fGetArticleInfo([”URL1”, ”URL2”]), hvor der gives en liste med to strings som er henholdsvis start og slut. Der er også en fjerde mulighed som er hvis ingen af if-statementsne er opfyldt som bare returnere en fejl.

A diagram of a flowchart

Description automatically generated

Figur 3 Flowdiagram for fGetArticleInfo's if-statements. Flowdiagram ses også i bilag 2.

De to første if-statements starter med at tjekke om chosenGameMode er en string, dog tjekker den første if-statement om stringen er == ”connected”, og den anden tjekker om det er ”random”. Den tredje if-statement tjekker om det er en liste, og om den liste har to elementer.

Et billede, der indeholder skærmbillede, tekst

Automatisk genereret beskrivelse

Figur 4 De tre if-statements som giver start og slut baseret på chosenGameMode.

## A/”connected”

Når chosenGameMode == ”connected” bliver funktionen fGetPageTitle() kaldt med argumentet ”https://en.wikipedia.org/wiki/Special:Random”. https://en.wikipedia.org/wiki/Special:Random er en funktion som er del af Wikipedia. Det er en funktion som vælger en tilfældig Wikipedia artikel så, hvis man sender en request til URL’en, bliver man omdirigeret til en tilfældig Wikipedia artikel.

fGetPageTitle returnere så URL’en på den tilfældige artikel samt artiklens titel. Det bliver så initialiseret variablen pageURL samt pageTitle, med værdierne som fGetPageTitle(pageURL) returnerer. Der køres nu en for-løkke hvor ”i” vil være alle tal mellem 0 og værdien iterations. I løkken startes det med at give variablen chosenSite værdien af et tilfældigt link fra currentSiteReferences. Derefter vil variablen currentSiteReferences, som holder en liste med links, få værdien som fGetLinks(f"https://en.wikipedia.org{chosenSite}") returnere. Grunden til at argumentet ikke bare kan være chosenSite er fordi alle links på en Wikipedia artikel er relative, altså vi får ikke hele linket men kun den relative path til artikler. I tilfældet i ”linket” på figur 5 er det relative link ”wiki/Rickrolling"



Figur 5 Billede af HTML reference på Wikipedia.

Til sidst vil funktionen returnere to lister goalLinksList og goalTitleList. Listen goalLinksList vil indeholde URL’en pageURL som blev fundet i starten samt den endelige chosenSite https://en.wikipedia.org{chosenSite}. goalTitleList vil indeholde titlen originalt fundet tidligere, samt titlen som findes ved at kalde fGetPageTitle med https://en.wikipedia.org{chosenSite}.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype

Automatisk genereret beskrivelse

Figur 6 Koden for "connected".

Referer til bilag 3 for flowdiagram.

## B/”random”

”random” er meget simpelt. Der startes med at blive initialisere to lister, en til links og en til titler. Der køres nu en for-løkke to gange, hvor der bruges fGetPageTitle(https://en.wikipedia.org/wiki/Special:Random), til at få navnet og linket på artiklen, og de bliver derefter tilføjet til deres respektive liste. Derefter returneres de to lister. Referer til bilag 4 for flowdiagram.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype

Automatisk genereret beskrivelse

Figur 7 Koden for "random".

## C/list

List er en relativt simpel stykke kode. Der startes med at initialisere to lister, hvorefter der startes en for-løkke som kører to gange. Løkken starter med at initialisere tempLink og tempTitle med værdierne fra fGetPageTitle(chosenGameMode[i]. De to værdier bliver så tilføjet til de førnævnte lister og løkken vil køre igen, og de to lister vil blive returneret.

For flowdiagram referer til bilag 5.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype

Automatisk genereret beskrivelse

Figur 8 Koden for List

## fGetPageTitle

fGetPageTitle går, som navnet hentyder til, den henter titlen på en given artikel. Funktionen tager en parameter ved navn URL som er en string, hvor denne string skal være et link til en Wikipedia artikel. Funktionen starter med at sende en web request til URL hvor responsen bliver gemt i variablen pageRequest. Den HTML kode som er gemt i pageRequest bliver nu brugt til at lave et BeatifulSoup object gennem bs4. Derefter bruges funktionen .find, fra bs4 til at finde et objekt, som har klassen firstHeading og returnere værdien af det objekt, som er titlen på artiklen. Der returneres nu artiklens url, som bliver fundet gennem web requesten, siden URL kan være ”https://en.wikipedia.org/wiki/Special:Random”, samt den fundne titel på artiklen.

Referer til figur 10 eller bilag 6 for flowdiagram.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype

Automatisk genereret beskrivelse

Figur 9 Koden for fGetPageTitle

Et billede, der indeholder tekst, diagram, linje/række, origami

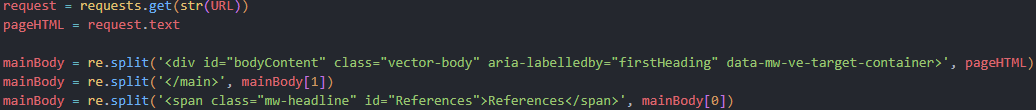
Automatisk genereret beskrivelse

Figur 10 Flowdiagram for fGetPageTitle

## fGetLinks

fGetLinks har en parameter som er URL. Funktionen starter med at sende en web request til URL for at få artiklens HTML , som vil blive gemt i variablen pageHTML.

pageHTML bliver nu splittet der hvor der står ”<div id="bodyContent" class="vector-body" aria-labelledby="firstHeading" data-mw-ve-target-container>”, dette er for at fjerne alt der er før brødteksten, derefter vil der blive splittet ved alt under ”</main>” og igen splittet ved ”<span class="mw-headline" id="References">References</span>” for at fjerne referencer til andet end Wikipedia artikler.



Figur 11 Koden for at hente og sortere unødvendig data fra.

På figur 11 er der vist en visuel repræsentation af, hvilken data det er der bruges, hvor alt inde i den røde boks er der hvor hentes links fra.

Et billede, der indeholder tekst, skærmbillede, software, nummer/tal

Automatisk genereret beskrivelse

Figur 12 Visuel repræsentation af den data der bliver brugt.

Efter vi har afskaffet alt den unødvendige data sorteres alle linkene fra. For at ekstrahere linkene fra bliver der brugt et lille regex filter, ’ <a href="/wiki/.\*?</a>’, som siger at noget der starter med <a href="/wiki/ og slutter med </a>, med hvilken som helst karakter, en eller flere gange, i mellem med ”.\*?” . Nu hvor alle linkene er fundet og gemt i listen siteReferences. Der kaldes nu fSortList med argumentet siteReferences, for at ekstrahere alle links og titler på links, og derefter returneret.

Referer til bilag 7 for flowdiagram.

Et billede, der indeholder tekst, Font/skrifttype, skærmbillede, linje/række

Automatisk genereret beskrivelse

Figur 13 Koden for filtrering af links.

## fSortList

fSortList tager en parameter som er siteReferences, dette er en liste af links givet af fGetLinks.

Der startes med at initialisere listen falseSiteReference, hvorefter der startes en for-løkke som itererer over alle elementer i siteReferences og tjekker om der er et kolon i elementet, og hvis der er så tilføj det til falseSiteReferences. Dette er for at fjerne alle ”falske” links fra siteReferences, siden alle sider som referer til noget som ikke er en Wikipedia artikel har formatet set i figur 14, hvor den øverste linje er et eksempel på en ”falsk artikel”.



Figur 14 Eksempel på et reelt link og et falsk link. Det øverste ”falske” link har ”/wiki/noget:noget andet”, mens reelle links har ”/wiki/noget”.

Derefter køres en list comprehension, som kun tilføjer elementer fra siteReferences, hvis de ikke er i falseSiteReferences. Hvor x er elementet og for alle elementer ser den om elementet ikke er i falseSiteReferences, og hvis det er sandt, bliver det tilføjet til listen.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype

Automatisk genereret beskrivelse

Figur 15 Koden beskrevet tidligere.

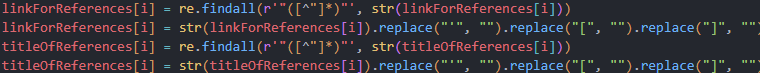
Den næste del af koden deler titler og links op i to forskellige lister. Dette bliver gjort ved hjælp af en for-løkke som itererer over alle elementer i siteReferences. Løkken starter med at bruge endnu et regex filter, af samme type som set på figur 13, hvor der er et start mønster, et slut mønster og alt det i mellem. Dette blive gjort for henholdsvis både titler og links, hvorefter de bliver tilføjet til deres respektive liste.

Et billede, der indeholder tekst, Font/skrifttype, skærmbillede, linje/række

Automatisk genereret beskrivelse

Figur 16 For-løkke som sorterer titler og links til hver sin liste

Nu for at gøre dataene brugbar skal de unødvendige karakterer fjernes, altså i eksempel linket 'href="/wiki/Why\_programmers\_dont\_like\_grass" 'skal der fjernes href=” ” så kun ’Why\_programmers\_dont\_like\_grass’ står tilbage. Dette bliver gjort gennem en blanding af regex, samt replace funktionen fra Python. Der startes med at finde alle steder som starter med ” og slutter med ” og så tager alt det der er imellem de to ”. Derefter erstattes alle ” ’ ”, ”[” og ”]” med ingenting. Dette bliver gjort for både links og titler.



Figur 17 Koden der fjerner unødvendige karakterer fra de givne links og titler.

linkForReferences og titleOfReferences returneres derefter.

A diagram of a flowchart

Description automatically generated

Figur 18 Flowdiagram over fSortList. Ses også i bilag 8.

# Test af programmet

Programmet fungerer godt, men har stadig fejl. En af dem er omdirigeringer. Hvis man for eksempel går ind på den danske artikel om ræve, og man skal prøve at finde artiklen ”Egentlige ræve”, vil man let kunne gøre det gennem browseren, men på grund af nogle ”schenanigans” er det er lille problem med programmet. Artiklen man prøver at nå er ”https://da.wikipedia.org/wiki/Egentlige\_r%C3%A6ve”, men man kan ikke tilgå denne artikel direkte, ikke fra min lille research. man kan kun tilgå denne side gennem en omdirigering fra siden ”Vulpes”. Med dets reference fra artiklen ræve som er set på figur 19.



Figur 19 Reference til egentlige ræve, fra artiklen ræve.

Og på grund af den måde fGetPageTitle fungerer på, at den ikke tager det omdirigerede link og returnere, men det link som funktionen laver web requesten med. Så linket som fGetPageTitle giver er til ”https://da.wikipedia.org/wiki/wiki/Vulpes” ikke ” https://da.wikipedia.org/wiki/Egentlige\_r%C3%A6ve”. Dette er dog kun et problem for start og slut artiklerne. Hvis der igen ses på figur 19, kan man se at i forhold til eksemplet på figur 14 har dette omdirigeringslink en ekstra attribut som er class. dette vil bruges til at vide om linket skal gen tjekkes, dog vil det kræve ekstra research for at finde en løsning.

Dog er der stadig problemer med hvordan der ekstraheres data fra referencerne. Denne fejl er i at tage titlen ”Vulpes” fordi det er den værdi title er lig med. Dette gør det svært at vide, i hvert fald i dette tilfælde at ”Vulpes” faktisk er ”Egentlige ræve”. Dette er relativt let at løse. Det er bare at ændre regex filteret i fSortList for at få ”Egentlige Ræve” i stedet for ”Vulpes”.

Et andet problem der kom var, at nogle gange gik man ikke til de artikler man trykkede på og, at det sidste link nogle gange bare ikke havde et link. Buggen blev dog først løst den 12/05, og problemet var en list comprehension, i fSortList, som var sat ind for at fjerne duplikerede links eller titler. Problemet var dog ikke i måden de blev fjernet men, at hvis der blev fjernet et link betød det nødvendigvis at titlen også var duplikeret og dermed vil titlen blive der men linket vil blive fjernet. En grund til at sådan et tilfælde kunne opstå kunne være de førnævnte omdirigeringer hvor to forskellige titler havde samme link pga. af en omdirigering. Dette blev løst ved at fjerne den list comprehension, dog kunne det også løses ved at køre en fælles for-løkke som tjekker links for to af samme link, også fjerner indekset for både link listen og titellisten.

Et billede, der indeholder tekst, Font/skrifttype, skærmbillede

Automatisk genereret beskrivelse

Figur 20 Eksempel på artikel med fire ens titler med fire forskellige links. Her kan der også ses den problematiske kode.

# Konklusion

Svarer kort på problemformuleringen, ud fra de ting man har konkluderet i løbet af rapporten. Det er meningen at en problemformulering og en konklusion skal kunne læses uafhængigt af resten af rapporten, og give et godt indblik i hvad rapporten egentlig handler om.

# Bibliografi

Gayor, I. (30. juli 2021). *A Random Wikipedia Article Generator*. Hentet 11. maj 2024 fra Stack Overflow: https://stackoverflow.com/questions/68596742/a-random-wikipedia-article-generator

w3schools. (u.d.). *Python RegEx*. Hentet 11. maj 2024 fra w3schools: https://www.w3schools.com/python/python\_regex.asp

w3schools. (u.d.). *Python Requests Module*. Hentet 11. maj 2024 fra w3schools: https://www.w3schools.com/python/module\_requests.asp

Wikipedia. (2. marts 2022). *Wikipedia:Random*. Hentet 11. maj 2024 fra Wikipedia: https://en.wikipedia.org/wiki/Wikipedia:Random

# Bilag

**Bilag 1**

A diagram of a flowchart

Description automatically generated

**Bilag 2**

A diagram of a flowchart

Description automatically generated

**Bilag 3**

A diagram of a flowchart

Description automatically generated

**Bilag 4**

A diagram of a flowchart

Description automatically generated

**Bilag 5**

Et billede, der indeholder diagram, skitse, tegning, tekst

Automatisk genereret beskrivelse

**Bilag 6**

A diagram of a website

Description automatically generated

**Bilag 7**

A diagram of a flowchart

Description automatically generated

**Bilag 8**

A diagram of a flowchart

Description automatically generated

1. <https://wikispeedrun.org/> eller <https://wikispeedruns.com/> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Main_Page> Sidst tjekket 11/05/2024 [↑](#footnote-ref-2)