**Vizualizacija algoritama za sortiranje**

**Vizualizacija podataka**

**Dominik Bošnjak,**

**Osijek 2020.**

Sadržaj

[1. Opis projektnog zadatka 1](file:///C:\Users\Korisnik\Desktop\viz_pro_dokumentacija.docx#_Toc4797)

[2. Programsko rješenje 2](file:///C:\Users\Korisnik\Desktop\viz_pro_dokumentacija.docx#_Toc4798)

[2.1 Inicijalizacija 2](file:///C:\Users\Korisnik\Desktop\viz_pro_dokumentacija.docx#_Toc4798)

[2.2 Ponašanje 2](file:///C:\Users\Korisnik\Desktop\viz_pro_dokumentacija.docx#_Toc4798)

[2.3 Algoritmi sortiranja 2](file:///C:\Users\Korisnik\Desktop\viz_pro_dokumentacija.docx#_Toc4798)

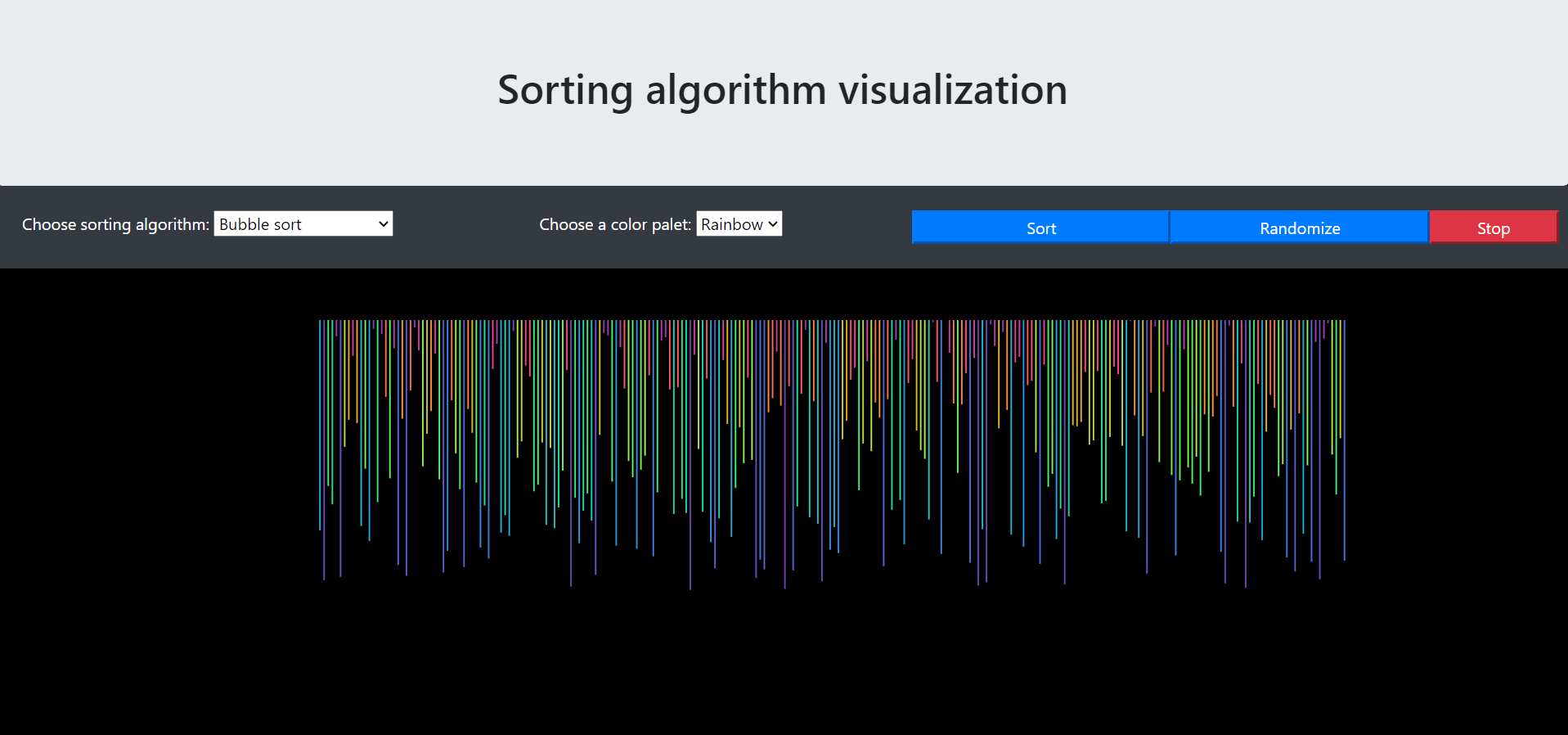
[3. Zaključak 4](file:///C:\Users\Korisnik\Desktop\viz_pro_dokumentacija.docx#_Toc4801)

[4. Korišteni izvori 4](file:///C:\Users\Korisnik\Desktop\viz_pro_dokumentacija.docx#_Toc4801)

# **Opis projektnog zadatka**

Ideja za temu ovog projektnog zadataka proizašla je iz praćenja stanice <https://www.reddit.com/r/dataisbeautiful/> na kojoj se često objavljuje vizualizacija različitih algoritama za sortiranje uz razne druge zanimljive vizualizacijske uratke. Cilj same vizualizacije jest pokazati pojedine korake prilikom sortiranja određenog skupa podatka.

Rješenje se sastoji od jednostavne web stranice prikazane na slici 1. Stranica se sastoji od jednostavnog korisničkog sučelja unutar kojeg je moguće odabrati algoritam sortiranja, mijenjati boje te pokrenuti i zaustaviti vizualizaciju.



Slika1 izgled web stranice

# **Programsko rješenje**

Web aplikacija pisana je u *JavaScript* programskom jeziku uz HTML (*engl. HyperText Markup Language*). Za dizajn izgleda stranice korišten je *bootsrap 4* pomoću kojeg je kreirana navigacijska traka koristeći *grid* sustav. Izgleda glavne stranice definiran je unutar *index.html* datoteke. Navedena datoteka sadrži poveznice na vanjske resurse kao što je već navedeni *bootstrap 4* te ostale komponente koje potrebno uključiti kako bi on ispravno radio. Nadalje poveznice na *JS d3* biblioteku koja se koristi kao osnovni alat za kreiranje vizualizacije uz uključeni modul *d3-scale-chormatic*[1] koji sadrži podršku boje.

*JavaScript* kod koji određuje ponašanje stranice odvojen je unutar zasebnog direktorija te je u nastavku objašnjena svaka skripta pojedinačno.

## Inicijalizacija

*Initialize.js* sadrži inicijalizaciju osnovnih varijabli koje su potrebne za rad vizualizacije sortiranje. Unutar nje definirana je veličina SVG elementa, kvantitativne skale za preslikavanje podatkovnog skupa u odgovarajući izlazni raspon, broj koji uz *d3.shuffle* naredbu kreira podatkovni skup za prikaz linija, skala boja. Na osnovi svih ovih varijabli kreiran je osnovni SVG element te je popunjen linijama. Skale su korištene za određivanje visine, pozicije i boje pojedine linije.

## Ponašanje

Osnovno ponašanje stranice odnosno ponašanje pri korisničkoj interakciji definirano je unutar *app.js*. Navedena skripta sadrži četiri funkcije koje omogućavaju promjenu boje, odabir algoritma sortiranja, onemogućavanje gumbova kako bi se izbjeglo neželjeno ponašanje te opcija „miješanja“ podatkovnog skupa, odnosno ponovno iscrtavanje linija nakon što je algoritam završio sa sortiranjem.

## Algoritmi sortiranja [2]

Implementacija pojedinih algoritama za sortiranje nalazi se unutar *sortingAlgorithms.js* i sadrži algoritme koji su u nastavku objašnjeni.

**Bubble sort –** Najjednostavniji algoritam sortiranja. Radi na način da uspoređuje dva susjedna elementa te ako su u pogrešnom redoslijedu zamjeni ih. Najgori slučaj pri sortiranju je kada je niz obrnuto posložen, tada je kompleksnost . Kompleksnost u najbolje slučaju jest , a to se dogodi kada je niz već sortiran

**Selection sort –** Ovaj algoritam radi na način da pri svakoj iteraciji pronalazi najmanji element u nizu (minimum) te ga zamjenjuje sa svakim elementom koji je veći dok ne dođe na ispravnu poziciju. Kompleksnost algoritma je .

**Insertion sort –** Pri svakoj iteraciji uzima jedan element te ga umeće na odgovarajuće mjesto unutar prijašnjih elementa koje je već stavio u odgovarajući poredak. Kompleksnost u najgorem slučaju, kada su elementi obrnuto poslagani, jest . Algoritam je dobar odabir kada je niz djelomično sortiran te ako je niz već sortiran ne mora se napraviti niti jedna zamjena te je potreban samo jedan prolaz kroz niz pa je kompleksnost u tom slučaju .

**Gnome sort –** Radi na osnovi koncepta kako „vrtni patuljak“ slaže svoje „lončiće sa cvijećem“. Promatra trenutni element („lončić“) i sljedeći, ako su u ispravnom redoslijedu pomiče se jedan element naprijed, u suprotnom zamjenjuje ih i pomiče se jedan element unazad. Ako nema prijašnjeg elementa pomiče se naprijed, a ako nema sljedećeg elementa, niz je sortiran. Pošto nema ugniježđenih petlji može se činiti da je kompleksnost linearna , što i je najbolji slučaj (niz je već sortiran), no u prosijeku je kompleksnost jer se vrijednost trenutne poziciji osim povećavanja može i smanjivati. Vidljivo je da radi slično kao i Insertion sort***,*** što je također uočljivo u vizualizaciji.

**Odd-even sort –** Inačica Bubble sorta***,*** a alternativni nazivi su **Brick** ili **Parity sort**. Radi tako što primjenjuje Bubble sort na parnim elementima polja pa Bubble sort na neparnim elementima, skroz dok niz nije sortiran. Originalno je napravljen za rad na paralelnim procesorima. Kompleksnost mu je .

**Double selection sort –** Varijacija Selection sorta, jer osim bilježenja minimuma, također bilježi i maksimum u svakoj iteraciji te ih smješta na odgovarajuću poziciji. Obavljaju se tri usporedbe s obzirom na dva elementa (par elementa je uspoređen, veći se uspoređuje sa trenutnim maksimumom, a manji sa trenutnim minimumom). Kompleksnost je i dalje , no radi 25% manje usporedbi nego Selection sort [3] .

**Cocktail sort –** Također varijacija Bubble sorta, no za razliku od njega, ovaj algoritam prolazi kroz niz u oba smjera. Prilikom prolaska od prvog do zadnjeg elementa, zamjenjuje elemente koji nisu u pravilnom redoslijedu i tako na kraj stavi najveći elementa. Nakon toga prolazi kroz niz u obrnutom smjeru no i dalje koristi istu usporedbu i tako najmanji element dovede na početak niza. Ne daje značajno poboljšanje u odnosu na Bubble sort, kompleksnost je i dalje .

**Merge sort –** Radi na principu *divide and conquer* („podijeli pa vladaj“) algoritama, koji su dizajnirani na osnovi granajućih rekurzija. Algoritam podjeli niz u dva dijela te napravi rekurzivni poziv (poziva sam sebe) na oba dijela niza te nakon toga spoji (*engl. merges)* obje polovice pozivajući drugu funkciju koja pretpostavlja da su obje polovice već sortirane. Kompleksnost je jer dijeli niz na dva dijela te mu je linearno vrijeme potrebno za spajanje dva dijela.

Osim navedenih algoritama sortiranja, unutar skripte nalazi se funkcija koja odrađuje pojedinu tranziciji. Naime algoritmi sortiranja se izvršavaju puno brže nego što je potrebno SVG tranziciji da se izvede pa nastaje problem da se tranzicije krivo izvedu ili uopće ne izvedu. Stoga je potrebno postupak sortiranja podijeliti na nekoliko dijelova *(eng. batch jobs*) ili koristiti asinkroni pristup. Iz navedenih razloga spomenuta funkcija deklarirana je kao asinkrona te prima dva parametra pomoću koji odrađuje translaciju te nakon što sačeka da se translacija izvrši vraća odgovor pozivajućem algoritmu sortiranja da može nastaviti sa iteracijama. Također zbog toga su i algoritmi sortiranja definirani kao asinkrone funkcije.

# **Zaključak**

D3 biblioteka vrlo je koristan alat za vizualizaciju i kako takav pokazao se vrlo pristupačan i učinkovit prilikom vizualizacije algoritama sortiranja. Jedna od mana jest ta što se sortiranje ne izvršava u stvarnom vremenu, odnosno prilikom svake zamjene potrebno je čekati da se zadana tranzicija obavi što rezultira da algoritmi vizualiziraju svoje postupke sporije nego što ih stvarno naprave.

# **Korišteni izvori**

* [1] <http://using-d3js.com/04_05_sequential_scales.html>
* [2] kolegij Algoritmi i strukture podataka
* [3] Mahesh, Goyani & Chharchhodawala, Mohammad & Mendapara, Bhargav. (2013).

Min-Max Selection Sort Algorithm -Improved Version of Selection Sort. 1-6.

* Vizualizacija podataka LV priručnik – tranzicije