Rakenne ja O-analyysi

Algoritmi luo uusia sukupolvia ja jokaisessa sukupolvessa se luo 100 uutta DNA:ta, mutatoi ne, pisteyttää ne ja lopuksi kaikki DNA:t järjestetään. Yhdessä DNA:ssa on yksi ratkaisu.

Joka kerta kun DNA luodaan, se saa isältään ja äidiltään sattumanvaraisesti arvot. isältä arvojen hakeminen kestää O(l) aikaa, missä l on sijaintien määrä. Äidiltä arvojen hakeminen kestää O(l^2) aikaa, missä l on sijaintien määrä. Toisen asteen potenssi johtuu siitä että samalla pitää tarkistaa onko sama sijainti tullut jo isältä. Täten crossover on aikavaatimukseltaan O(l^2)

Mutaatioiden maksimimäärä on käyttäjän asettama. Yksi mutaatio vain vaihtaa kahden sijainnin paikkaa, jolloin mutaation aikavaatimus on O(1). Näin ollen yhden DNA:n kaikkien mutaatioiden aikavaatimus on O(m), missä m on max mutaatioiden määrä.

Pisteiden laskuun tarvitaan laskea reitti, minkä DNA pitää sisällään. Reitin laskemisessa käydään jokaisen sijainnin jokainen paketti läpi. Näin ollen reitin laskemiseen menee O(p) aikaa missä p on kaikkien pakettien määrä.

Pisteisiin lasketaan myös DNA:n eroavaisuus suhteessa vanhempiinsa. Eroavaisuuden laskemisessa käydään isän, äidin ja lapsen sijainnit läpi. Eroavaisuuden laskemisessa menee siis O(l) aikaa missä l on sijaintien määrä. Vakiokerroin on 3.

Reitin ja eroavaisuuden yhdistävä pisteidenlasku algoritmi tekee vain normaaleja laskutoimituksia vakioajassa. Täten koko pisteiden laskuun menee O(p+l).

DNA:n pisteitä käytetään järjestämisessä. Järjestämiseen käytetään merge sorttia, jonka aikavaativuus on O(nlog(n)). DNA:n pisteet laitetaan muistiin ensimmäisen laskemisen jälkeen. Tällöin järjestämiseen menee O(p*d + d*log(d)) missä d on sukupolven DNA määrä. Sukupolvessa on aina 100 DNA:ta. Joten järjestämiseen menee O(100p + 460) => O(p), missä vakiokerroin on 100

Sukupolvia tehdään käyttäjän valitsema määrä. Täten koko algoritmiin menee O(g*(l^2+m+p)) => O(gl^2 + gm + gp), missä g on sukupolvien määrä, ja vakiokerroin on noin 100

O-analyysistä näkee, että jos haluaa suorittaa monta sukupolvea, pitää sijaintien, mutaatioiden ja pakettien määrä pitää alhaisena. jos jotain näistä kolmesta nostaa suureksi, algoritmi alkaa käyttäytymään kuin O(n^2) algoritmit, jolloin ajaminen on hidasta.