Stijn Janssens, Anthony Hermans, Evert Heylen, Pieter Coeck
Universiteit Antwerpen
België
Stijn.Janssens@student.uantwerpen.be, Anthony.Hermans@student.uantwerpen.be,
Evert.Heylen@student.uantwerpen.be ,Pieter.Coeck@student.uantwerpen.be

Voor fase 1 hebben we de volgende algoritmes gekozen:
1. Anthony: DFA -> Regex
2. Pieter: Subsetconstruction (NFA->DFA)
3. Stijn: Productautomaat
4. Evert: ε-NFA -> DFA

Voor fase 2 hebben we gekozen om ons project te gaan zoeken in een biologische context. In deze context kwamen we op het idee om gebruik te maken van het concept DNA-strings. DNA-strings zijn sequenties van DNA bouwstenen (Adenine, Guanine, Thymine en Cytosine). DNA-cellen liggen aan de basis van hoe onze cellen werken en zijn dus de kern van ons bestaan.

Aan de hand van deze informatie hebben we beslist om een systeem implementeren dat nagaat of een kleine DNA string voorkomt in een grotere DNA string. Ons project is maatschappelijk relevant want het gebruikt kan worden in de medische sector zoals bv bij medicijnen, ziektes opsporen , DNA-matching etc. Het DNA-matchen houdt dan in dat we kunnen nagaan of een hele string gelijk is aan een gegeven string. Dit is dan later uit te breiden aan de hand van complementaire strings.

Dit systeem hebben we de naam HELIX gegeven. Deze naam is een verwijzing naar de dubbele helix van een DNA-molecule.

HELIX leunt ook sterk aan bij de inhoud van de cursus Talen en Automaten aangezien we deze DNA-strings kunnen zien als regex. Bovendien gebruiken we ook NFA's bij het voorstellen van de strings. Naast het gebruik van NFA's en regex gaan we ook onze performantie zo goed mogelijk maken dankzij het "Table Filling Algorithm".

De DNA-strings kunnen aanzienlijk groot worden waardoor we ook onze algemene performantie zo goed mogelijk moeten maken.
Aangezien dat DNA-strings opgebouwd uit 4 bouwstenen kunnen we 2 bits gebruiken in het geheugen. Met andere woorden een DNA-string van lengte x zal dus een grootte van ⌐x/4¬ bytes innemen van het geheugen. Dit gaan we opvangen door middel van memory optimalisatie. Naast memory optimalisatie gebruiken we ook multithreading en een profiler.

We gaan overigens ook gebruik maken van QT project voor het creëren van een grafische interface. Dit laat toe om HELIX gemakkelijker te gebruiken en daardoor is het dus toegankelijker voor een groter publiek.

Dit systeem kunnen we dan afhankelijk van de vooruitgang van het project uitbreiden door de complementaire DNA string op te stellen en deze te vergelijken. Overigens kunnen we ook nog meer gradaties in matchen voorzien als extra functionaliteit. Dankzij deze uitbreidingen zullen we zeker wel aan de 90 werkuren per groepslid komen.