De klasse Regexp is bekend.

De klasse ThompsonNA in [thompsonna.h](https://intranet.tiwi.ugent.be/Algoritmen-II/reeks11/thompsonna.h) implementeert het algoritme van Thompson en gebruikt Regexp als hulpklasse om op basis van een reguliere uitdrukking een lijst op te stellen van de verbindingen in de NA. Hoe je de verbindingen kan opvragen volgt vrij eenvoudig uit de hoofding van de klasse.

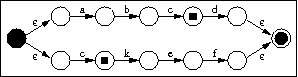
Schrijf je eigen versie van grep, mijngrep, die gebruik maakt van regexps van bovenstaande vorm. Dit programma wordt opgeroepen met de opdracht

grep <regexp> <bestandsnaam>

en schrijft dan alle regels van het bestand uit waarin een deelstring zit die overeenkomt met de opgegeven regexp uit.

Een automaat kijkt of een bepaalde string behoort tot de taal gedefinieerd door de bijbehorende regexp. Om te zien of een gegeven string een *deel* bevat dat voldoet aan de regexp (en grep doet dit met elke lijn uit een bestand) moeten we twee dingen doen:

1. We moeten nagaan of de automaat zijn eindtoestand heeft bereikt op elke plaats van de string (want de deelstring die overeenkomt met de regexp kan daar eindigen).
2. We moeten de automaat herstarten op elke plaats van de string, want de gezochte deelstring kan daar beginnen.

Het eerste kan zonder probleem gebeuren bij zowel een NA als bij een DA. Het tweede is veel gemakkelijker met een NA. Immers, met een deterministische automaat moet je voor elke plaats in de hooiberg opnieuw beginnen om te zien of er daar geen string begint die voldoet aan de regexp. Met een NA kan je alles tesamen doen door op elke plaats in de string de begintoestand van de NA terug toe te voegen aan de toestandsverzameling van de NA (en dat is doodsimpel: leg gewoon een epsilonverbinding van *elke* toestand van de NA naar de begintoestand). Nemen we bijvoorbeeld de regexp **abcd|ckef**. Als we al ...abc hebben ingelezen, dan kan het zijn dat we al drie letters van abcd hebben, maar ook dat we één letter van ckef hebben. De NA ziet er dan uit zoals in de tekening.  


**Tip:**

* In principe kan je een automaat voorstellen als een graaf, en dus lijkt overerving van een of ander graaftype logisch. Is dit wel zo? De structuur moet geschikt zijn om strings te herkennen.

Natuurlijk is een DA meestal efficiënter dan een NA, maar bij een NA kan je elke plaats in de string als startplaats beschouwen om een overeenkomende deelstring te vinden op de manier zoals hiervoor beschreven. Wil je zowel de efficiëntie van de DA als de uniforme benadering van de NA, dan kan je een DA maken en die beschouwen als een NA, waardoor je op elk ogenblik een begintoestand kan toevoegen, en dan het resultaat terug omzetten naar een DA.

Ook hier moet je de vraag stellen welke structuur het beste is om opzoeken toe te laten. Is dit dezelfde als bij een NA?