Derde serie sommen.

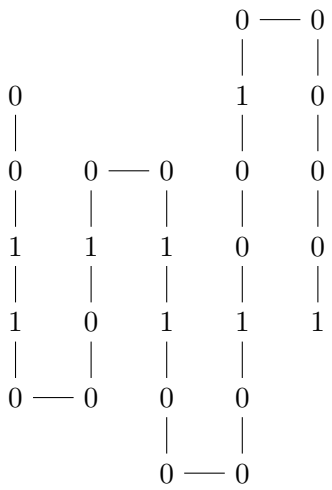
1. Een veelhoek heet *convex* als elke lijn tussen twee hoekpunten helemaal binnen die veelhoek valt. Zo'n veelhoek kan opgedeeld worden in driehoeken door (niet snijdende) lijnen tussen genoeg hoekpunten te trekken. Natuurlijk kunnen door het trekken van die lijnen nieuwe veelhoeken ontstaan, die ook moeten worden opgedeeld. Net zolang tot alleen driehoeken over zijn. Bij een driehoek en een vierhoek heb je niet veel keus, maar een vijfhoek kan al op een aantal verschillende manieren worden opgedeeld. Stel dat we zo'n veelhoek opdelen, en dan de som van alle omtrekken van de resulterende driehoeken nemen als kosten voor die opdeling. We nemen voor lief dat we dan sommige lijnen dubbel tellen. We willen die kosten minimaliseren.

   (a) Een greedy aanpak zou kunnen zijn: Ga elke keer rond de veelhoek, en verbind twee hoekpunten waarbij je één hoekpunt overslaat. Wat kost dit als er n hoekpunten zijn? Geef een voorbeeld waarbij dit niet werkt (i.e., waarbij de kosten niet minimaal zijn).

   (b) Een recursieve aanpak zou kunnen zijn. Verbind twee hoekpunten en bereken het optimum voor de overblijvende veelhoeken (één of twee). Wat mankeert er aan deze aanpak?

   (c) Bedenk een dynamic programming aanpak en laat zien dat die werkt. Wat kost die?

2. Protein Folding. Proteïnen zijn (meestal lange) ketens aminozuren. Die liggen niet gestrekt, maar zijn opgevouwen als een harmonica om energie te besparen. Een keten bestaat uit hydrofiele en hydrofobe aminozuren. Hydrofobe aminozuren liggen graag tegen elkaar aan. Als we de hydrofobe aminozuren representeren met een 1 en de hydrofiele met een 0, merken we op dat lage energie kan worden bereikt door de string zo op te vouwen dat horizontaal zo veel mogelijk paren 1'en naast elkaar komen te liggen. Als 0011000100110000100100001 als onderstaand wordt opgevouwen zien we dat vier paren naast elkaar komen te liggen.



   (a) Bedenk een dynamic programming algoritme dat voor een gegeven string deze functie maximaliseert (en dus de energie minimaliseert), en daarbij de plaats(en) aangeeft waar de keten gevouwen moet worden om het maximum te bereiken. (Als er meer dan één maximum is, kan het algoritme toch slechts één van die maxima teruggeven). Leg uit waarom je algoritme werkt, zo mogelijk aan de hand van een klein voorbeeld.

   (b) Wat is de complexiteit van je algoritme? Wat zijn de parameters die een rol spelen in de afschatting van de complexiteit?

(Voor wie een extra 50 minuten heeft om te investeren staat een goede uitleg van het probleem en de aanpak op `https://www.youtube.com/watch?v=h60raqnvm0s`. Verwijs bij het beantwoorden van de vragen niet naar deze clip, maar leg uit wat je doet.)

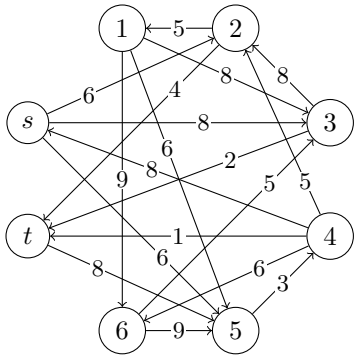
3. Network flow.

   (a) Teken een voorbeeld van een netwerk waarin het Ford-Fulkerson-algoritme een cyclische stroom kan introduceren.

   (b) Als er een cyclische stroom in een netwerk loopt, heeft de stroom dan dezelfde waarde langs alle kanten van de cykel?

   (c) Kan er bij het MKM-algoritme (gelaagd netwerk, blokkerende stroom, gelaagd netwerk, etc.) ook een cyclische stroom ontstaan? Waarom wel/niet?

4. Gegeven is het volgende netwerk met capaciteiten langs de kanten. De stroom is (nog) overal 0.



   (a) Vind in dit netwerk een maximale stroom met behulp van het Ford-Fulkerson-algoritme.

   (b) Teken het gelaagde netwerk dat bij dit netwerk hoord (als de stroom nog 0 is) en geef daarin een blocking flow aan.

   (c) Tel de blocking flow op bij de flow in het netwerk. Geef de waarden van stroom (f) en capaciteit (c) langs de kanten in het formaat f/c. Bereken met het resultaat een nieuw gelaagd netwerk.

   (d) Geef een voorbeeld van deen cut met minimale capaciteit. (Merk op dat de capaciteit van je doorsnijding gelijk moet zijn aan de waarde van de maximale stroom gevonden bij het eerste antwoord.)