Tweede serie sommen.

1. Dit is een programmeeropgave. Programmeeropgaven zijn doorgaans niet moeilijk, maar kunnen wel veel tijd kosten. Programmeeropgaven moeten worden ingeleverd met code, documentatie (evt. in de vorm van commentaar in de code), complexiteitsanalyse, en testresultaten. De opgave bestaat uit de volgende onderdelen:

   (a) schrijf een functie die een random graaf op NODES knopen produceert. Bijvoorbeeld:

   ```
   int graph[NODES][NODES];
   int i,j;
   srand(4321854);
   for(i=0;i<NODES;i++)
   for(j=0;j<NODES;j++)
   if( rand() %2 )
   graph[i][j]=rand() %100+1;
   else
   graph[i][j]=-1;
   ```

   (b) Schrijf een functie die op een naïve manier het korste pad van knoop 0, naar knoop NODES-1 vindt. Dat gaat bijvoorbeeld door recursief *alle* paden van 0 naar NODES-1 te vinden en dan de kortste terug te geven.

   (c) Schrijf een functie die met de methode van Dijkstra het kortste pad vindt.

   (d) Vergelijk de twee methoden voor een paar waarden van NODES door een clock te gebruiken.

   Note: het is mij bekend dat voor beide methoden implementaties (vooral in Python) "all over the internet" te vinden zijn. Ze zijn echter ook niet moeilijk zelf te implementeren. Als u een "gevonden" programma voor de test gebruikt, maak dan wel netjes een verwijzing, zoals het hoort.

2. Gegeven is een graaf met gewichten en twee knopen s en t in die graaf. Sommige kanten kunnen weggenomen worden zonder dat het kortste pad van s naar t verandert. Misschien is er een kant in de graaf zodat er na wegnemen van die kant *geen* pad meer is van s naar t. Beschrijf een efficiënte algoritme die de kant selecteert met de eigenschap dat na wegnemen van die kant de lengte van het kortste pad van s naar t *zoveel mogelijk* toeneemt. Wat is de complexiteit van uw algoritme?

3. Een greedy algoritme voor MST kan (ook) als volgt gaan.
   Herhaal totdat je een boom hebt: 1. Zoek een cykel met Depth First Search (noteer knopen totdat je een keer dezelfde tegen komt);2. Neem de/een zwaarste kant uit die cykel weg.

   (a) Waarom is dit een greedy algoritme?

   (b) Wat is de complexiteit van deze algoritme?

4. In het INDEPENDENT SET probleem proberen we in een graaf $G = (V, E)$ een zo groot mogelijke onafhankelijke verzameling te vinden. Dat is een verzameling knopen $V' \subseteq V$ zodat geen enkele kant uit E tussen twee knopen uit V' loopt. Beschrijf een divide-and-conquer algoritme die op invoor G, k zegt of G een independent set van minstens k knopen heeft. Wat is de complexiteit van uw algoritme?