1. Er zijn twee mogelijke oplossingen. Gebruik eerst Dijkstra om het kortste pad tussen s en t te bepalen. Haal vervolgens om beurten *elke* kant uit de graaf en gebruik opnieuw Dijkstra. Geef de oplossing terug die de grootste increase in lengte geeft. Een (iets) slimmere aanpak is om alleen de kanten uit het kortste pad om beurten weg te halen. Beide oplossingen zijn worst-case $O(n)$, maar de tweede oplossing geeft gemiddeld een betere performance.

2. Een greedy algoritme begint bij de gegeven k werknemers en streept van elke werkennemer de directe chef (en de chef van de chef en de chef van de chef van de chef etc) uit de verzameling (van *alle* werknemers). De verzameling die overblijft kan worden uitgenodigd. Dit is de maximale legale verzameling. Immers, als x en y allebei uitgenodigd worden dan kan x niet de ondergeschikte zijn van y want anders was y weggestreept. Als er een grotere verzameling van werknemers zou zijn dan de geselecteerde, dan *moet* daar een paar x, y in zitten zo dat één van de twee ondergeschikt is aan de andere. Onze algoritme nodigt dan de ondergeschikte uit, en *niet* de superieur. Wegens de structuur van het bedrijf heeft de ondergeschikte minder conflicten met anderen, dus dit is een tegenspraak. De complexiteit is lineair

3. Sorteer de dozen op volgorde van grondoppervlak aflopend. Nu kan een doos die lager in de rij staat niet meer bovenop een doos passen die hoger in de rij staat. Immers als het grondoppervlak kleiner is, dan is of de breedte of de diepte kleiner. (Gelijke dozen geven geen probleem, dus we nemen aan dat die er niet zijn). Noteer boven de eerste doos het getal h_1 (de hoogte van de eerste doos) en maak als volgt de rij af. Voor elke doos i kijken we bij het genooteerde getal h_j voor alle dozen j die eerder in de rij staan. Neem het maximum van die getallen waarbij alleen de dozen j meedoen als doos i erbovenop kan. Voor dat maximum H noteren we $H + h_i$ bij doos i. De maximum bereikbare hoogte is tenslotte het maximum van de genoteerde getallen. De complexiteit is $O(n \log n)$ wegens het sorteren.

4. (a) Neem een balk die op plaatsen 4,5, en 6 gesneden moet worden. De greedy algoritme kost 10+5+5. Beter is 10+6+2 wat je krijgt door de balk op plaats 6 te laten zagen en vervolgens op plaatsen 4 en 5.

   (b) Stel dat de balk op plaatsen $\ell_0, \ldots, \ell_{n+1}$ gesneden moet worden, waarbij ℓ_0 het beginpunt en ℓ_{n+1} het eindpunt van de balk is. Laat $c[i, j]$ de optimale kosten zijn voor het snijden van het stuk dat begint bij ℓ_i en eindigt bij het stuk ℓ_j. We willen dan weten wat $c[0, n + 1]$ is. We weten wat $c[i, i + 1]$ is voor elke i. Immers daar zitten geen sneden in, dus die kosten zijn 0. Aangezien we de lengte van het stuk ℓ_i naar ℓ_{i+2} kennen, en zo'n stuk, maar op 1 manier gezaagd kan worden, kunnen we ook deze entries in de tabel invullen. Nu kan een stuk $c[i, j]$ worden ingevuld door voor alle k tussen i en j te kijken wat de kosten zijn voor het snijden van het stuk ℓ_i tot ℓ_k plus de kosten van ℓ_k tot ℓ_j. Hierbij moet dan natuurlijk nog $\ell_j - \ell_i$ worden opgeteld. De complexiteit van deze algoritme is $O(n^2)$, de grootte van de tabel die moet worden ingevuld.

5. (a) Er zijn twee mogelijkheden. Als de berekende maximale stroom door (u, v) niet gelijk is aan de capaciteit, verandert er niets. Als dat wel zo is, dan kijken we of er een stroomvergrotend pad kan worden gevonden van s naar u en een stroomvergrotend pad van t naar v in het netwerk waar de maximale stroom is ingetekend. Als dat allebei kan, dan kan de stroom met 1 langs die paden worden vergroot.

   (b) Als de maximale stroom door (u, v) niet gelijk is aan de capaciteit, gebeurt er weer niets. Anders kijken we of er een stroomvergrotend pad kan worden gevonden van u naar s en

een stroomvergrotend pad van t naar v. (Dat moet kunnen, anders liep er geen stroom door (u, v). Langs die paden kan de stroom met 1 worden verkleind.

6. (a) Van twee gegeven verzamelingen V_1 en V_2 kan gemakkelijk worden gecontroleerd of ze:
   i. Samen alle addressen bevatten (door wegstrepen in lineaire tijd).
   ii. Allebei een rondje van minder dan K kilometer voorstellen.

   PAKKETJES is dus een probleem in NP.

   (b) We bewijzen volledigheid door een reductie van TSP. Van een gegeven instantie van TSP maken we een instantie van PAKKETJES die bestaat uit *twee* identieke grafen, allebei een copie van de TSP graaf. We zetten de afstand tussen twee plaatsen in verschillende delen van de nieuwe graaf op oneindig (of een heel groot getal) Als de gegeven instantie een rondje kleiner dan K heeft, heeft de nieuwe instantie natuurlijk twee rondjes kleiner dan K (in elke graaf één). Als de nieuwe instantie van PAKKETJES twee rondjes kleiner dan K heeft, dan zijn die rondjes allebei in afzonderlijke delen van de graaf (er is geen snelle verbinding tussen de twee delen), en is er dus een rondje kleiner dan K in de oorspronkelijke graaf.