Algoritmiek, deeltoets 2, 3 november 2009, Voorbeeldantwoorden.

Opgave 1. Approximatie voor TSP.
Dit probleem is tweemaal op hoorcollege behandeld, namelijk als toepassing van MST en bij het onderwerp approximatie. In de contekst van het TSP-probleem moet het woord "cykel" natuurlijk worden opgevat als "Hamiltonian Cykel", ofwel "cykel door alle knopen".

(a) Een r-approx voor TSP moet
- in polynomiale tijd lopen
- opleveren: een TSP toer/cykel (volgorde van knopen)
- kosten van die toer: hoogsten r maal het optimum

(b) Vanuit een gegeven opspannende boom T kun je een route langs alle punten maken door de boom te doorlopen, bv. volgens een DFS pad. De lengte van zo'n route is tweemaal het gewicht van de boom omdat elke boomkant er exact tweemaal in zit.
(Aanname: symmetrie van kantgewicht)
Als je aanneemt dat je tussen elk tweetal knopen rechtstreeks kunt gaan, kun je dubbele voorkomens van een knoop elimineren. Als de afstandstabel aan de driehoeksongelijkheid voldoet, kan de route hierdoor alleen maar korter worden, en dan resulteert een TSP-route waarvan de lengte hoogstens tweemaal het gewicht van T is.

(c) Als C een cykel (door alle knopen) is, kun je uit C een kant verwijderen en (mits kantgewichten niet negatief zijn) resulteert een opspannende boom T van gewicht kleinergelijk de lengte van C. Een minimale opspannende boom heeft gewicht kleinergelijk dat van T, dus zeker begrensd door de lengte van C.

(d) Bereken een MST, neem de toer die ontstaat door hem te doorlopen. De ratio is nu 2.

2. Kortste paden.
(a,b) Zie de overzichtjes in boek of college.
(c) Als je hier met Dijkstra op de proppen komt krijg je 1 pt (en de letter D erbij). Je komt nog wat verder (lagere tijd) als je het gegeven gebruikt.
Mogelijkheid 1: vervang elke kant van gewicht 2 door twee van gewicht 1 met nieuwe knoop ertussen, en doe BFS. Tijd O(n+m).
Mogelijkheid 2: Dijkstra met uitgekiende PriorityQueue. Bij

het verwerken van een knoop op afstand l heb je alleen maar knopen met schatting l, l+1 en l+2. Je kunt de PQ dus maken met drie lijsten en alle operaties daarop in constante tijd doen.

Wat niet kan: "gewoon" BFS, kijk maar naar een s en t waartussen een pad bestaat van drie kanten met gewicht 1, en een ander pad van twee kanten met gwicht 2. Er is nu een pad van twee hops, maar het korste pad is drie hops.

3. Randomisatie.
(a) Hoe je de array ook benadert, het is mogelijk dat de eerste n/2 waarden die je bekijkt allemaal verschillend zijn. Geen van de bekeken waarden kun je opleveren, omdat de onbekeken waarden dan nog allemaal aan elkaar gelijk kunnen zijn, en willekeurig.
Je moet dus meer dan n/2, ofwel Theta(n) werk doen.
(Dat het ook in O(n) KAN is niet zo vanzelfsprekend.... want hoe houd je de geziene waarden bij? Gelukkig kun je op het idee komen dat de absolute meerderheid, ook de mediaan is, en dan LINEAIRE selectie toepassen.)

(b) Je kunt zonder verder maar iets te bekijken, een Random waarde uit A opleveren, wat met 50% kans de gezochte is. Dit algoritme levert 1 punt op.
Mooier is natuurlijk om meerdere waarden te bekijken en een dubbele waarde te constateren. Je kunt naar een VAST (maar wel parametriseerbaar) aantal kijken, dan is het MC, of doorgaan tot een dubbele, dan is het LV.

(c,d) Hangt natuurlijk van je algoritme in b af.
Het viel me wel op, dat veel mensen niet in staat zijn om de verwachte tijd, danwel slaagkans, van dit toch niet al te ingewikkelde algoritme te bepalen.

4 De lift.
Zodra je de woorden polynomiaal en NPC in een zin hoort kun je natuurlijk al nee antwoorden, maar voor de volle mep aan punten moet je dit toch wat uitwerken.
Wat je vooral NIET moet doen, is zeggen dat je het liftprobleem kunt oplossen met SSS. Zo'n opmerking is totaal zinloos, want SSS is NPC, dus je kunt vrijwel elk probleem oplossen met SSS.

Nou werd er meermalen beweerd, dat je voor een minimaal aantal ritten moet gaan proberen een zo vol mogelijke lift te krijgen.
("Dus je moet een SSS oplossen.")

Beweren is een ding, bewijzen probeerde niemand, en het klopt ook niet. Neem bv een klas met drie kinderen van 7kg en twaalf van 1kg, en een lift van omvang M=12.
De klas KAN in drie ritten worden vervoerd: 71111, 71111, 71111.
Ik kan ook van de klas een subset van gewicht M maken, namelijk de twaalf kleintjes. Voor een minimaal aantal ritten is dat echter hartstikke fout, want voor de andere drie heb ik dan nog drie ritten nodig.

Maar je moet terugdenken aan de wijze woorden van HansB:
als je wilt bewijzen dat een probleem B NPC is, moet je UITGAAN van een bekend NPC probleem A, en A reduceren naar B.
Je moet dus NIET laten zien dat je een lift-instantie kan oplossen met SSS, maar: dat je een willekeurige SSS instantie kan oplossen met Lift.
Zo'n SSS instantie bestaat uit getallen s1 tm sn plus T en vraagt of er een subset is van gewicht T.
Schrijf S = som si, en neem aan T >= S/2 (dat mag, want het bestaan van een subset met som T is equivalent met het bestaan van een subset met som S-T).
Vorm nu de volgende Lift-instantie: Verwek een EXTRA kind met gewicht 2T-S, dus de klasgewichten zijn
s1, .., sn, 2T-S,
voeg hieraan toe een lift met capaciteit T en vraag of de klas in 2 ritten vervoerd kan worden.

Let op, dat mijn kindertjes samen exact 2T wegen, dus dit zal alleen kunnen als de lift tweemaal nokkienokkie zit.
ALS het lukt om de klas met twee ritten te vervoeren: in 1 van de ritten zit mijn extra kind NIET, dus die rit is een subset van het oorpronkelijke SSS probleem met gewicht T.
ALS de SSS-instantie en JA-antwoord heeft: dan kan ik ook inderdaad mijn klas vervoeren in twee ritten.
Het antwoord van mijn geconstrueerde Lift-instantie is dus hetzelfde als van de gegeven SSS-instantie.

Hieruit volgt dus dat SSS <= Lift en dus dat Lift in NPC.

Omdat we denken dat NP /= P, en dus NPC is niet polynomiaal, is er WAARSCHIJNLIJK geen polynomiaal algoritme voor DeLift.