Algoritmiek, toets 4 november 2008.
Voorbeeldantwoorden.
Per vraag was zes punten te verdienen, uitslag is totaal gedeeld door drie.

Vraag 1, DFS.
(a) Een kant uv is een cross edge indien de dfs procedure in v geeindigd is voordat hij in u begint. In termen van de d/f parameters: f[v] < d[u].

Je kunt NIET zonder meer zeggen dat het een kant is tussen verschillende DFS bomen in het woud. Een definitie die dynamische eigenschappen gebruikt (tijdgebonden), bv dat het een kant van een grijze naar een zwarte knoop is oid, zijn minder goed.

(b) Wanneer de graaf ongericht is, heb je bij elke kant uv ook de tegengestelde richting vu. Dit impliceert dat u wordt ontdekt, op z'n laatst tijdens het doorlopen van de adjacency lijst van v, voordat v wordt afgesloten. Dus: d[u] < f[v].

Hier zie je dat je met een goede karakterisering van een cross edge, de vraag snel kunt beantwoorden.

(c) Die bewering is ONWAAR. Stel s heeft buren a en b, maar a heeft ook een kant naar b. De startknoop ontdekt a, vanwaar het zoeken wordt voortgezet en b wordt gevonden vaunuit a. Als b en a zijn afgesloten continueert het zoeken in s, maar nu wordt de kant sb een forward kant.

Vraag 2, Maximum Spannende Boom.
Je kunt dit probleem vertalen naar Minimum Spanning Tree. Elke spannende boom heeft namelijk evenveel kanten, exact n-1. Zij M het hoogste kantgewicht plus 1; definieer

$$w'(uv) = M - w(uv)$$

en W' het gewicht van een boom als som van deze gewichten. Voor elke Tree T is nu $W'(T) = (n-1).M - W(T)$, dus een W'-MINIMALE tree is gelijk aan een W-maximale tree. Je kunt dus een standaard MST algoritme gebruiken met de gemodificeerde gewichten. Of in een bestaand algoritme overal "grootste" lezen ipv "kleinste".

Vraag 3, Langste Simpel Pad.
(a) De redenering is ONJUIST. Een concatenatie van twee (k-1)-paden bevat wel knoop k ten hoogste 1x, maar lager genummerde knopen kunnen op beide deelpaden voorkomen en dan is het pad niet simpel.
Uiteindelijk vind je dus een of ander slingerend pad door de graaf, waarin lager genummerde knopen heel vaak kunnen voorkomen en hoger genummerde heel weinig.
De meesten hadden gezien dat de initialisatie op infty er voor zorgt dat er sowieso nooit iets verandert in de afstanden. Dat was een tweede "bug" in de code die ik zelf niet had gezien.

(b) Waarschijnlijk niet. Je kunt namelijk door een reductie van Hamiltonian Cycle aantonen dat LSP NPC is. Van de NPC-klasse wordt aangenomen (al is dit niet bewezen) dat er geen polynomiale oplossingen voor bestaan.
Nu iets over de reductie. Zij G met n knopen een instantie van Hamiltonian Cycle; maw, de vraag is, heeft deze graaf een simpele cykel met alle knopen erin.
Vorm de graaf G' als volgt. Kies een knoop v in G en vervang deze door v1 en v2, elk met verbindingen naar alle buren van v in G. Voeg knopen w1 en w2 toe, waarbij wi alleen is verbonden met vi. Er is geen kant tussen v1 en v2!
Stel de vraag "Heeft G' een LSP van lengte n+2?".
Merk op dat G', n+3 knopen heeft en een pad van lengte n+2 bevat dus alle knopen.
Als G een HC heeft, kun je in die cykel v door beginpunt v1 en eindpunt v2 vervangen, aanvullen met w1 en w2, en je hebt een LSP van n+3 knopen.
Als G' een LSP van n+3 knopen heeft, moeten w1 en w2 wel het begin en eindpunt zijn (graad 1) en dus is er een deel van dat pad van v1 naar v2 dat alle knopen behalve w1, w2 bevat. Als je v1 en v2 op elkaar plakt heb je een Ham cyk in G te pakken.

In een volledig bewijs moet je dus de implicatie tussen het bestaan van een HC en van een LSP tweee kanten op bewijzen!
Veel personen kwamen met een of andere omzetting waarbij je wel kon laten zien dat een HC in G, impliceert dat er een pad van lengte n in G' is. Echter, omgekeerd hoeft het dan niet te gelden: als je een pad van lengte n hebt, hoeven de

eindpunten daarvan geen buren te zijn en dat impliceert dus
niet dat er een cykel is.

Let er dus op, dat als je een (NPC) probleem A reduceert naar
probleem B (om die NPC te bewijzen), dan moet je
TRANSFORMEREN IN EEN RICHTING: Een willekeurige instantie a
  van probleem A moet je omschrijven naar een instantie b
  van probleem B.
BEWIJZEN IN TWEE RICHTINGEN: Als a een oplossing heeft dan
  heeft b een oplossing; als b een oplossing heeft dan heeft
  a een oplossing.

Opgave 4, Randomiserende Algoritmen.
Verschillende mogelijkheden.
(1) Pak random 2 waarden, stuur naar output.
MC met complexiteit 2 en slaagkans 50%.
(2) Pak random k waarden, stuur naar output.
MC met cpx k en slaagkans ~~ (1-1/2^(k-1)).
(3) Trek net zo lang random tot je twee verschillende hebt.
LV met verwachte cpx 3.
(4) Trek net zo lang random zonder terugleggen tot je
er twee gezien hebt.
Sherwood met verwachte cpx 3, WC cpx n/2+1.

Opgave 5, Cryptografie.
(a) De aanvaller vervangt y door y'=2y.
Bij decryptie krijg je x' = y'/z = 2(x.z)/z = 2x.

(b) Nee dat kan niet.
Motivering met definitie van "weten": Elke truuk die je,
naast y=x.z, ook de waarde y'=(x+1).z oplevert,
is voldoende krachtig om je z te laten uitrekenen als z = y'-y.
Volgens de definitie WEET je dus z al als je y' kunt geven.