Prüfer: Vöcking (V), Unger (U)
Fächer: Effiziente Algorithmen, Krypto, Compilerbau
Note: 1.0 (weil Nikolaus war)

Prolog:

V: Ahja, erster Versuch? Eigentlich sollten wir Dich ja beim ersten Versuch durchfallen lassen.
Ich: *grinse halb* Sehr witzig!
V: Naja, bei Prüfungen soll man so doofe Scherze ja eigentlich nicht machen, das kann sonst ganz unangenehme Folgen haben.. Naja, welche Reihenfolge hättest Du... Ähhh Sie denn gerne? Neuerdings sollen wir die Studenten ja siezen.
Ich: Effiziente, Krypto und dann Compilerbau.

Akt I: Effiziente Algorithmen

V: Womit möchtest du denn gerne anfangen?
Ich: Hmmm... Nehmen wir mal geometrische Algorithmen.
V: Ok. Wir hatten da ja so Algorithmen beschrieben, die mit einer Sweepline arbeiten. Wie ging das denn?
Ich: Erkläre den einfachen Sweepline-Algo für senkrechte Linien.
V: Ok, das war ja noch ein recht simpler Fall. Wie ist das denn mit allgemeinen Linien in der Ebene?
Ich: Erkläre den allgemeineren Fall.
V: Dann hatten wir ja noch einen anderen Ansatz mit Divide And Conquer. Insbesondere war da ja ein interessanterer Algorithmus für die Berechnung des kürzesten Abstands zweier Punkte innerhalb einer Punktemenge.
Ich: Erkläre den Algo.
V: Und wie ist das mit der Laufzeit?
Ich: Erkläre naive Implementation mit immer neuen Sortierungen. Erkläre Optimierung, sage allerdings, das man die Punkte auf der Y-Achse vorsortieren könne, und dann in O(n) die passenden rausfiltern kann. Insgesamt also O(n log n).
V: Na, das stimmt so nicht ganz... Erklärt, warum nicht, und sagt, dass das mit integriertem Mergesort klappt (ich verstehe immer noch nicht, warum das andere nicht auch gehen soll).
V: Wir hatten ja noch eine weitere Art, Probleme zu lösen. Da hatten wir ja was mit dem kleinsten umschließenden Kreis.
Ich: Erkläre.
V: Und wie sieht's aus mit der Laufzeit?

Ich: Stelle Rekursionsgleichung auf, erkläre.
V: Wie ist das denn mit höher dimensionierten Problemen. Wie geht denn da die Dimension mit ein?
Ich: (Erinnere mich an SeideLP und rate). Als Fakultät.
V: Genau. So, wir haben ja noch ein wenig Zeit. Wir hatten ja noch so ein paar Algorithmen bezüglich des Makespans. Was konnte man mit denen denn so erreichen?
Ich: Zähle auf die 2-Approx, 4/3-Approx, das PAS und die 2-Approx auf allgemeinen Maschinen.
V: Ok, dann erklär mir doch mal den letzten.
Ich: Fange an zu erklären. Komme ein wenig ins Stammeln als ich den Bezug auf's LP erkläre und den n*m Variablen. Ich konnte den Algorithmus zwar, es war aber alles ein wenig ungeordnet.

Akt II: Algorithmische Kryptographie

U: Ok, gerade hattest du ja so einen Algorithmus mit kleinstem umschließenden Kreis erklärt. Dabei hast du auch den Begriff einer Basis benutzt. Wie könnte man das denn kryptographisch verwenden?
Ich: Ähhh...
U: Ok, fangen wir mal anders an. Wir hatten ja mal ein Verfahren, um ein Geheimnis aufzuteilen. Wie ging das?
Ich: Beschreibe Threshold-Scheme über Polynom.
U: Und wie macht man das jetzt mit einem Kreis?
Ich: Definition einer Hyper-Kugel erst durch entsprechend viele Punkte eindeutig. Geheimnis ist dann Kugel-Mittelpunkt.
U: Genau. Und wo wurde das dann eingesetzt?
Ich: Grübel... Homomorphe Verschlüsselung?
U: Genau! Und weisst du noch was da so gemacht wurde?
Ich: Ähh.. Neee....
U: Naja, nicht so schlimm, eine Lücke kann man sich erlauben. Was hatten wir denn so für Protokolle, die man dann so mit asymmetrischen Verschlüsselungsverfahren bauen konnte?
Ich: Beginne aufzuzählen.
U: Beschreib mir doch mal ein Identifikationsprotokoll. Das von Shamir.
Ich: Beschreibe Shamir, sage, ist Zero Knowledge.
U: Aha! Da bleiben wir doch gleich mal, was heisst das etwas formeller?
Ich: Erkläre Sachen mit Transkript und Simulator.
U: Und wie kann jetzt hier betrogen werden?
Ich: Erkläre.
U: Ok. Was muss denn der Simulator alles als Eingabe bekommen, damit das klappt?
Ich: Laber ein bischen rum, aber komme nicht wirklich drauf.

U: Erklärt, dass der eine Instanz des Verifiers braucht, damit er berücksichtigen kann, wenn der Verifier schummelt. Naja, aber bleiben wir mal bei Zero Knowledge. Was hatten wir denn sonst noch so?
Ich: Zähle auf.
U: Ok, wir hatten da doch noch eins zum Unterschreiben. Wie ging das denn?
Ich: Zeige erst Schnorr normal, dann Nicht-Interaktiv.
U: So eine Hash-Funktion... Wie konstruiert man denn sowas? Ich: Hmm... In der Vorlesung hatten wir das Markles-Meta-Verfahren. U: Ok. Und wo setzt man denn diese Unterschrift ein?
Ich: *grüble einige lange Sekunden* Bei elektronischem Geld oder bei Wahlen?
U: Richtig! Bei elektronischem Geld oder bei Wahlen. Wir hatten ja diese drei großen assymetrischen Verfahren. Wie sieht's denn bei denen mit der Sicherheit aus?
Ich: Gebe obere Schranken für die Verfahren an. Sage Rabin ist so sicher wie Wurzelziehen ist so sicher wie faktorisieren.
U: Und wie sieht's mit der Bitsicherheit aus?
Ich: Beschreibe Bitsicherheit bei RSA. Gebe ganz grob an, wie man Zahlen halbiert bzw. verdoppelt.

Akt III: Intermezzo

V: Jetzt wo ich das hier gerade mit ggT höre (war beim Faktorisieren durch Wurzeln gefallen)... Wie geht das eigentlich?
Ich: Ähhh... Erkläre notdürftig mit Worten den euklidischen Algorithmus, müsste auch halbwegs richtig gewesen sein.
U: Das ist ok, das haben wir in der Vorlesung auch nicht wirklich gemacht.
V: Ahso. Wie schnell geht das denn?
Ich: Auf jeden Fall in Polynomialzeit.
V: Also der Algorithmus den ich kenne, geht so: ggt(a,b) = ggt(a-b,b). Das ist natürlich nicht mehr polynomiell.
Ich: Erkläre kurz, wie man durch benutzen des Rests aus a/b deutlich schneller ist. Ich meine, dann ist das sogar logarithmisch in der Eingabelänge.
V: Nee, nicht ganz. Das ist dann logarithmisch in der Zahlengröße, also insgesamt polynomiell. Naja, war jetzt aber auch nur für meine andere Vorlesung, ich möchte in BuK eine Übungsaufgabe daraus stellen.
Ich: Ähhh ok.

Akt IV: Compilerbau

V: In was für Schritte ist denn die Synthesephase unterteilt?
Ich: Ähh... Zwischencodeerzeugung, Codeoptimierung, Codegenerierung.

V: Genau. Synthesephase macht ihr gar nicht so viel, oder?
Ich: Naja, so die letzten 3-4 Wochen oder so.
V: Schade, dabei ist das ja eigentlich gerade der interessante Teil. Dann erzähl mit doch mal was zur lexikalischen Analyse. Was macht man denn da eigentlich?
Ich: Beschreibe Zerlegung eines Wortes in Matchings von regulären Ausdrücken, first longest match, Eindeutigkeit für Zerlegung und Analyse, Laufzeit.
V: Ja, die Laufzeit kann zwar theoretisch quadratisch werden, aber in der Praxis sind die regulären Ausdrücke kurz und unterschiedlich genug, dass das nicht passiert. Wie geht's denn dann weiter?
Ich: Mit der syntaktischen Analyse. Da wird dann aus der Token-Folge ein Baum erstellt. Beschreibe, warum aus Effizienz-Gründen LR(k) und LL(k) nötig sind.
V: Wie sieht denn die Definition von LR(k) aus?
Ich: Schreibe auf, zusammen mit Definition von fi, fo, und la-Mengen.
V: Was ist denn eigentlich mit LL(2) oder LL(3) Mengen?
Ich: Hmm, gute Frage, wurden in der Vorlesung nicht behandelt.
V: Ja, das liegt einfach daran, dass der zusätzliche Look-Ahead keine neuen Sprachen zulässt. Es bringt also nichts.
Ich: Oho, das ist ja interessant. Erzähle einfach noch was zur Teilmengen-Beziehung von LR(k) und LL(k), einfach, um wissend zu wirken.
V: Genau! So, das war's dann auch. Lass uns doch mal kurz allein.

Epilog

Die Atmosphäre war sehr freundlich. Als ich mich beim Makespan-Algo etwas verhaspelt hatte, stand mir Herr Vöcking hilfreich zur Seite. Walter hat mir auf zig verschiedenen Wege die Frage über die Eingabe für den Simulator gestellt, um mir eine Inspiration für die richtige Antwort zu geben. Ich kann den Lehrstuhl für Prüfungen nur weiterempfehlen!