

Prüfung

Datenstrukturen und Algorithmen (D-MATH RW)

Felix Friedrich, Aritra Dhar, Pesho Ivanov, Departement Informatik

ETH Zürich, 29.1.2020.

Name, Vorname:

Legi-Nummer:

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen konnte und dass ich die allgemeinen Richtlinien gelesen und verstanden habe.

I confirm with my signature that I was able to take this exam under regular conditions and that I have read and understood the general guidelines.

Unterschrift:

Allgemeine Richtlinien:**General guidelines:**

1. Dauer der Prüfung: 150 Minuten.
2. Erlaubte Unterlagen: Wörterbuch (für von Ihnen gesprochene Sprachen). 4 A4 Seiten handgeschrieben oder ≥ 11 pt Schriftgrösse.
3. Benützen Sie einen Kugelschreiber (blau oder schwarz) und keinen Bleistift. Bitte schreiben Sie leserlich. Nur lesbare Resultate werden bewertet.
4. Lösungen sind direkt auf das Aufgabenblatt in die dafür vorgesehenen Boxen zu schreiben (und direkt darunter, falls mehr Platz benötigt wird). Ungültige Lösungen sind deutlich durchzustreichen! Korrekturen bei Multiple-Choice Aufgaben bitte unmissverständlich anbringen!
5. Es gibt keine Negativpunkte für falsche Antworten.
6. Störungen durch irgendjemanden oder irgendetwas melden Sie bitte sofort der Aufsichtsperson.
7. Wir sammeln die Prüfung zum Schluss ein. Wichtig: Stellen Sie unbedingt selbst sicher, dass Ihre Prüfung von einem Assistenten eingezogen wird. Stecken Sie keine Prüfung ein und lassen Sie Ihre Prüfung nicht einfach am Platz liegen. Dasselbe gilt, wenn Sie früher abgeben wollen: Bitte melden Sie sich lautlos, und wir holen die Prüfung ab. Vorzeitige Abgaben sind nur bis 15 Minuten vor Prüfungsende möglich.
8. Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen.
9. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.

Exam duration: 150 minutes.

Permitted examination aids: dictionary (for languages spoken by yourself). 4 A4 pages hand written or ≥ 11 pt font size.

Use a pen (black or blue), not a pencil. Please write legibly. We will only consider solutions that we can read.

Solutions must be written directly onto the exam sheets in the provided boxes (and directly below, if more space is needed). Invalid solutions need to be crossed out clearly. Provide corrections to answers of multiple choice questions without any ambiguity!

There are no negative points for wrong answers.

If you feel disturbed by anyone or anything, let the supervisor of the exam know immediately.

We collect the exams at the end. Important: You must ensure that your exam has been collected by an assistant. Do not take any exam with you and do not leave your exam behind on your desk. The same applies when you want to finish early: Please contact us silently and we will collect the exam. Handing in your exam ahead of time is only possible until 15 minutes before the exam ends.

If you need to go to the toilet, raise your hand and wait for a supervisor.

We will not answer any content-related questions during the exam. Please write comments referring to the tasks on the exam sheets.

Question:	1	2	3	4	5	6	7	Total
Points:	22	16	20	13	13	18	18	120
Score:								

Verwenden Sie die Notation, Algorithmen und Datenstrukturen aus der Vorlesung. Falls Sie eine andere Herangehensweise wählen, erklären Sie Ihre Antworten in nachvollziehbarer Weise!

Use notation, algorithms, and data structures from the course. If you use a different approach, explain your answers in a comprehensible way!

Aufgabe 1: Verschiedenes (22P)

- /2P (a) Führen Sie auf dem folgenden Array einen Aufteilungsschritt des Sortieralgorithmus Quicksort durch. Benutzen Sie als Pivot das Element 5.

On the following array, perform a partitioning step of the sorting algorithm Quicksort. As pivot, use the element 5.

3	8	10	11	9	5	7	8	2	6	4
0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9	10

- /3P (b) Betrachten Sie folgende Adjazenzmatrix, welche zu einem ungerichteten Graphen mit vier Knoten gehört. Skizzieren Sie zuerst den Graph. Was ist die grösste ganze Zahl für x , so dass es ein Paar von Knoten a und b gibt, so dass die zu x zugehörige Kante zwingend auf einem kürzesten Pfad von a nach b liegt.

Consider the following adjacency matrix that corresponds to an undirected graph with four nodes. First, sketch the graph. What is the largest integer number for x such that there is a pair of nodes a and b such that a shortest path from a to b must include the edge that corresponds to x ?

$$\begin{pmatrix} 0 & 1 & 7 & 4 \\ 1 & 0 & 4 & 7 \\ 7 & 4 & 0 & x \\ 4 & 7 & x & 0 \end{pmatrix}$$

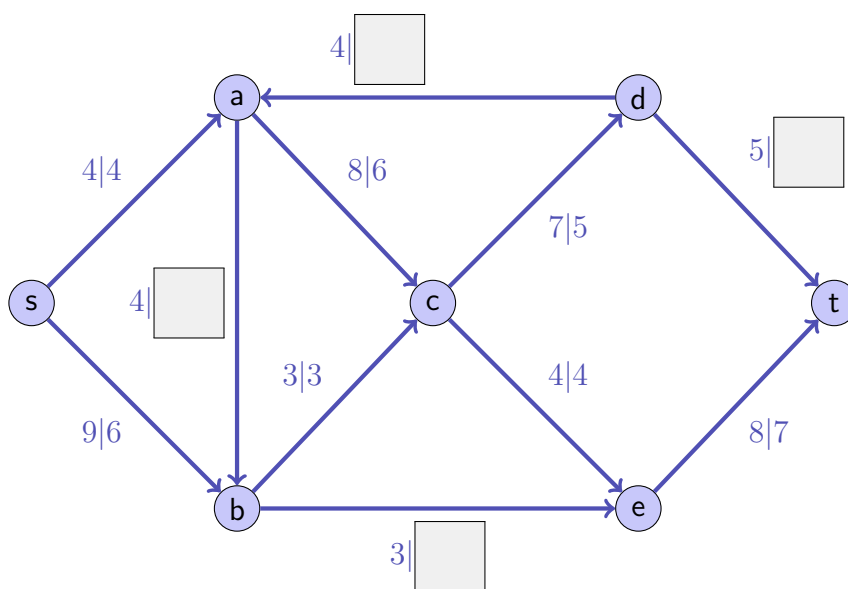
Graph

$x =$

- (c) Gegeben ist das folgende Flussnetzwerk mit Quelle s und Senke t . Die einzelnen Kapazitäten c_i und Flüsse ϕ_i sind an den Kanten angegeben als $c_i|\phi_i$. Ergänzen Sie die fehlenden Flusswerte auf den Kanten, so dass ϕ ein gültiger Fluss ist. Dieser wird (in diesem Beispiel) in jedem Falle maximal sein. Geben Sie den Wert des Flusses f an. Zeichnen Sie in der Abbildung einen Schnitt ein, der zeigt, dass ϕ maximal ist.

Provided in the following is a flow network with source s and sink t . Capacities c_i and flows ϕ_i are provided at the edges as $c_i|\phi_i$. Complete the missing flow values at the edges such that the overall flow ϕ is a valid flow. This will be maximal (in this example) in all cases. Provide the value f of the flow. Draw into the figure a cut that shows that ϕ is indeed maximal.

/5P



$f =$

- (d) Sie haben einen sehr grossen Datensatz aus n unterschiedlichen Zahlen und wollen das k -kleinste Element finden ($k \ll n$). Wie machen Sie das effizient? Benennen Sie verwendete Datenstrukturen und Laufzeit des Algorithmus.

You have a huge data set with n different numbers, and you want to find the k -smallest element ($k \ll n$). How do you do this efficiently? Provide the used data structures and the runtime of the algorithm.

/3P

Für die folgenden ja/nein Fragen geben Sie jeweils die Antwort mit kurzer Begründung. Fassen Sie sich kurz!

For the following yes/no questions provide the answer with a brief explanation. Be brief!

- /3P (e) Die asymptotische Laufzeit eines randomisierten Algorithmus ist bis auf eine Konstante im schlechtesten Fall gleich gross wie im Erwartungswert.

The worst-case asymptotic running time and expected asymptotic running time are equal to within constant factors for any randomized algorithm.

richtig/*correct* falsch/*wrong*

Begründung / *Justification*

- /3P (f) Jeder vergleichbasierte Sortieralgorithmus kann zu einem stabilen Sortieralgorithmus gemacht werden mit asymptotisch bis auf einen konstante Faktor unveränderter Laufzeit

Any comparison based sorting algorithm can be made to be stable, without affecting the asymptotic running time by more than a constant factor.

richtig/*correct* falsch/*wrong*

Begründung / *Justification*

/6P (b) Gegeben sei die folgende Rekursionsgleichung:

$$T(n) = \begin{cases} 2T(\frac{n}{4}) + 1, & n > 1 \\ 1 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (nicht rekursive), einfache Formel für $T(n)$ an und beweisen Sie diese mittels vollständiger Induktion. Gehen Sie davon aus, dass n eine Potenz von 4 ist.

Hinweis:

Für $q \neq 1$ gilt $\sum_{i=0}^k q^i = \frac{q^{k+1}-1}{q-1}$.

Consider the following recursion equation:

Specify a closed (non-recursive), simple formula for $T(n)$ and prove it using mathematical induction. Assume that n is a power of 4.

Hint:

For $q \neq 1$ it holds that $\sum_{i=0}^k q^i = \frac{q^{k+1}-1}{q-1}$.

In den folgenden Aufgabenteilen wird jeweils angenommen, dass die Funktion g mit $g(n)$ aufgerufen wird ($n \geq 0$). Geben Sie jeweils die asymptotische Anzahl von Aufrufen der Funktion $f()$ in Abhängigkeit von $n \in \mathbb{N}$ mit Θ -Notation möglichst knapp an. Die Funktion f ruft sich nicht selbst auf. Sie müssen Ihre Antworten nicht begründen.

In the following parts of this task we assume that the function g is called as $g(n)$ ($n \geq 0$). Specify the asymptotic number of calls of $f()$ depending on $n \in \mathbb{N}$ using Θ notation as succinct as possible. The function f does not call itself. You do not have to justify your answers.

(c)

```
void g(int n){
  for (int i = 0; i < n; ++i){
    for (int j = i; j < n; ++j){
      f();
    }
  }
}
```

Anzahl Aufrufe von f / *Number of calls of f*

/1P

(d)

```
void g(int n){
  f();
  if (n>1){
    f();g(n/2);
  }
  if (n>2) {
    f();g(n/2);
  }
}
```

Anzahl Aufrufe von f / *Number of calls of f*

/3P

/3P (e) Gegeben sei die folgende Rekursionsgleichung:

$$T(n) = \begin{cases} 2T(n/4) + \log_4 n, & n > 1 \\ 0 & n \leq 1 \end{cases}$$

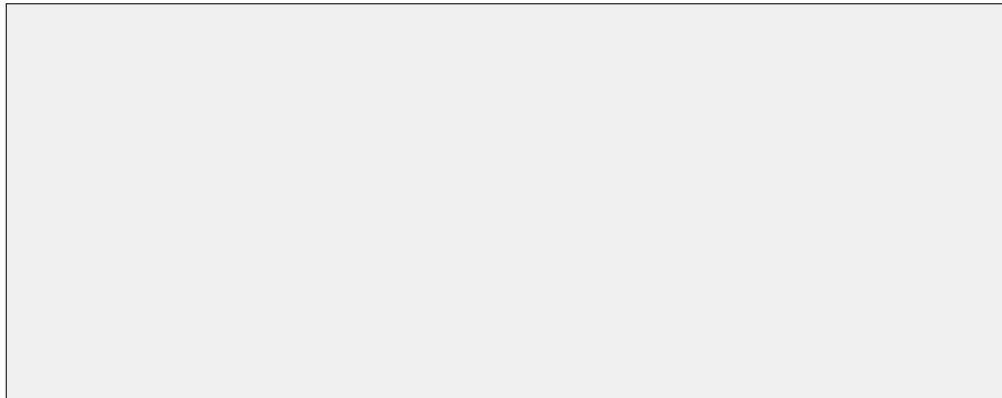
Consider the following recursion equation:

Schreiben Sie eine Funktion g , die bei Aufruf von $g(n)$ genau $T(n)$ Aufrufe von f erzeugt. Nehmen Sie an, dass $n = 4^k$ für ein $k \geq 0$.

Write a function g that when called as $g(n)$ will produce $T(n)$ calls to f . Assume that $n = 4^k$ for some $k \geq 0$.

```
// pre: n = 4^k for some k >= 0
```

```
void g(int n){
```



```
}
```

```
}
```

Aufgabe 3: Dynamic Programming: Spielstrategie (20P)

Gegeben ist eine Reihe von Münzen mit Werten v_1, v_2, \dots, v_n . n sei gerade. Wir spielen ein Spiel gegen einen Gegner mit wechselnden Zügen. In jedem Zug nimmt ein Spieler entweder die erste oder die letzte Münze der Reihe, entfernt sie und bekommt den Münzenwert. Aufgabe: Bestimmen Sie die maximale Summe an Münzwerten F_{max} , die wir garantiert gewinnen können, wenn wir zuerst ziehen (also unabhängig davon, was der Gegner macht). Beispiele:

Consider a row of n coins of values v_1, v_2, \dots, v_n , where n is even. We play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Task: determine the maximum possible sum of coin values F_{max} we can definitely win if we move first (i.e. no matter how the other player goes).

Examples:

▪ $v = (5, 3, 7, 10) : F_{max} = 15 \leftarrow (10+5)$.

▪ $v = (8, 15, 3, 7) : F_{max} = 22 \leftarrow (7+15)$.

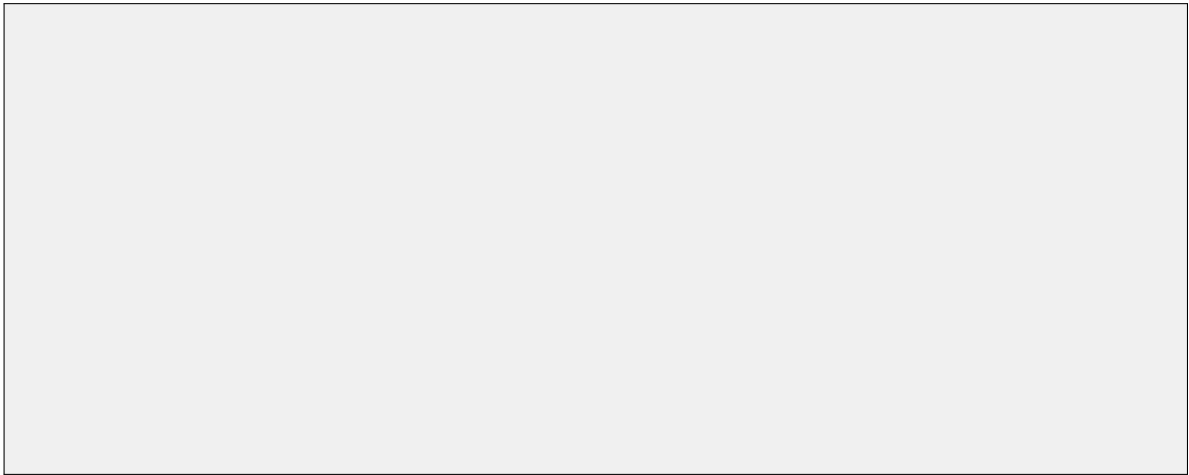
- (a) Liefert die Strategie, bei jedem Zug die Münze (von den beiden Münzen am Anfang und Ende) mit höchstem Wert zu wählen, eine optimale Lösung? Wenn ja, erklären Sie. Geben Sie andernfalls ein Gegenbeispiel aus den obigen Beispielen.

Does choosing the coin with highest value (out of the two first and the last remaining) at each move guarantee an optimal solution? If yes, provide a brief explanation. If not, give a counter example from the examples provided above.

/3P

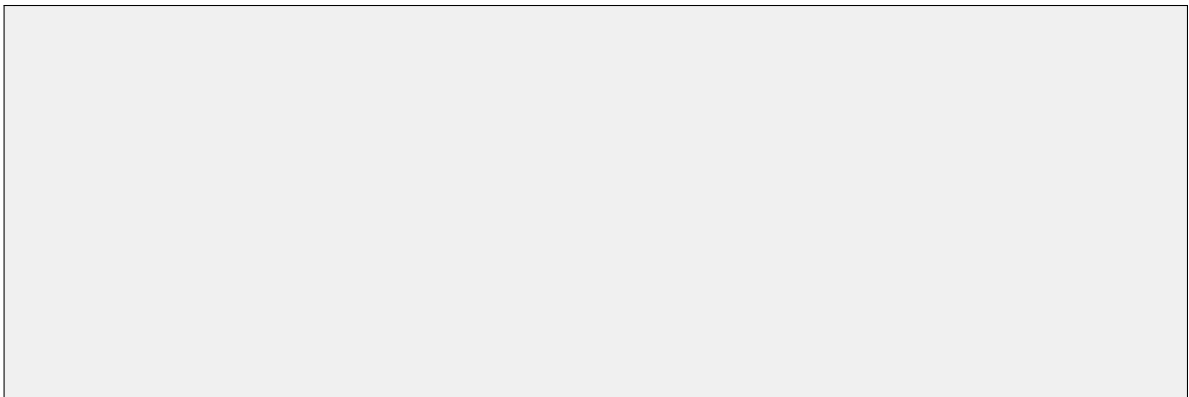
- /4P (b) Skizzieren Sie den Rekursionsbaum (z.B. für $n = 4$), der entsteht, wenn das Problem mit Rekursion naiv gelöst wird. (Beobachten Sie die Redundanzen dieser Lösung).

Sketch the recursion tree (e.g. for $n = 4$) that arises when the problem is solved with recursion in a naive way. (Observe the redundancies of this solution).



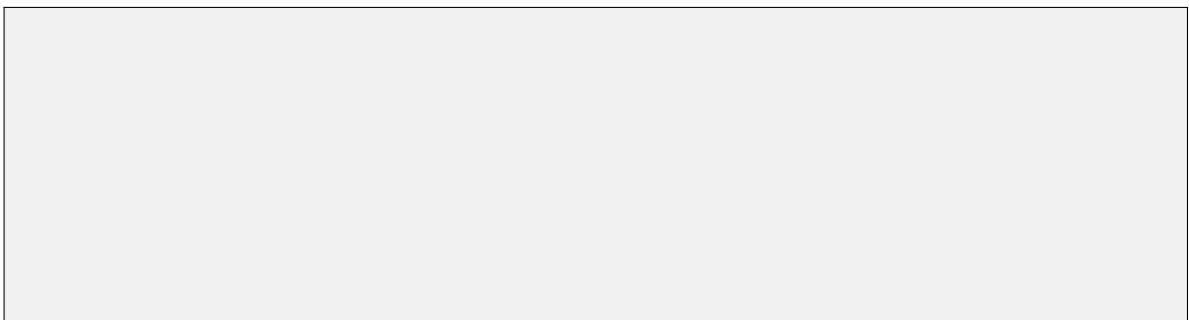
- /2P (c) Überlegen Sie sich nun eine Lösung mit dynamischer Programmierung. Was ist die Dimension und Grösse der DP-Tabelle. Welche Bedeutung hat jeder einzelne Eintrag.

Now think about a solution using dynamic programming. What is the dimension and size of the DP-table? What is the meaning of a table entry?



- /2P (d) Beschreiben Sie die Abhängigkeiten: in welcher Reihenfolge können die Einträge berechnet werden.

Describe the dependencies: in which order can the entries be computed?



- (e) Beschreiben Sie die eigentliche Berechnung. Wie berechnet sich ein Eintrag aus bereits berechneten Einträgen.

Describe that actual computation. How is an entry computed from the already computed entries.

/3P

- (f) Wie kann aus der DP Tabelle der maximale zu gewinnende Betrag abgelesen werden?

How can the maximum value the user can collect be obtained from the DP table?

/1P

- (g) Zeigen Sie die DP-Tabelle für das folgende Beispiel.

Show the DP-table for the following example.

/3P

$$v = (12, 4, 6, 10)$$

- (h) Geben Sie die asymptotische Laufzeit an, die Ihr Algorithmus für die Berechnung des maximalen Wertes im schlechtesten Fall benötigt.

Estimate the worst case asymptotic run-time for your algorithm used to compute the maximum value the user can collect in the coin game.

/2P

Aufgabe 4: Graphen: Wechselkurse (13P)

Die grosse Vielfalt von Cryptowährungen geht einher mit viel Hype und Verdrehtheiten. Vielleicht können Sie von den seltsamen Umrechnungskursen zwischen den Währungen profitieren?

Gegeben seien die festen Umrechnungskurse zwischen den n Währungen: $A_{ij} \in \mathbb{R}^+$ sei der Betrag in Einheiten der Währung j , den sie für 1 Einheit der Währung i erhalten ($1 \leq i, j \leq n$).

Sie starten mit 100 Einheiten der Währung 1 und Sie dürfen es in andere Währungen umtauschen, und diese weiter umtauschen (etc.), so oft Sie wollen.

- /4P (a) Sie möchten wissen, ob Sie mehr als 100 Einheiten der Währung 1 anhäufen können. Formulieren Sie dieses Problem als Graphenproblem. Beschreiben Sie Knoten und Kanten des Graphen, wie sie mit der Aufgabe zusammenhängen und was man für die Lösung der gestellten Frage im Graph finden muss.

The great variety of cryptocurrencies causes a lot of hype and craziness. Maybe you can profit from the weird exchange rates between different currencies.

You are given the fixed exchange rates between n currencies: $A_{ij} \in \mathbb{R}^+$ is the amount of currency j that you will receive for one unit of currency i ($1 \leq i, j \leq n$). You start with 100 units of currency 1, and you can exchange it for other currencies and exchange those currencies, etc. as many times as you wish.

You want to know it is possible to accumulate more than 100 units of currency 1. Formulate the task as a graph problem. You need to define what the nodes and edges correspond to in the original task and what has to be found in the graph.

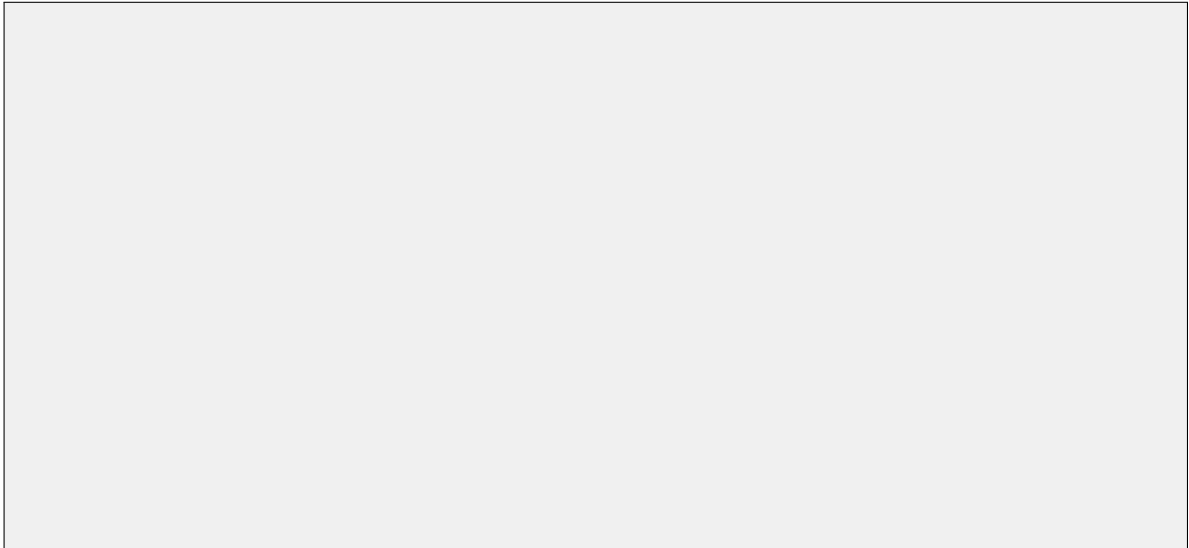
Tipp / *Hint*: $\log(a \cdot b) = \log(a) + \log(b)$

- (b) Zeichnen Sie den Graphen für $n = 3$ Währungen mit den folgenden Wechselkursen.

Draw the graph for $n = 3$ currencies with the following exchange rates.

/1P

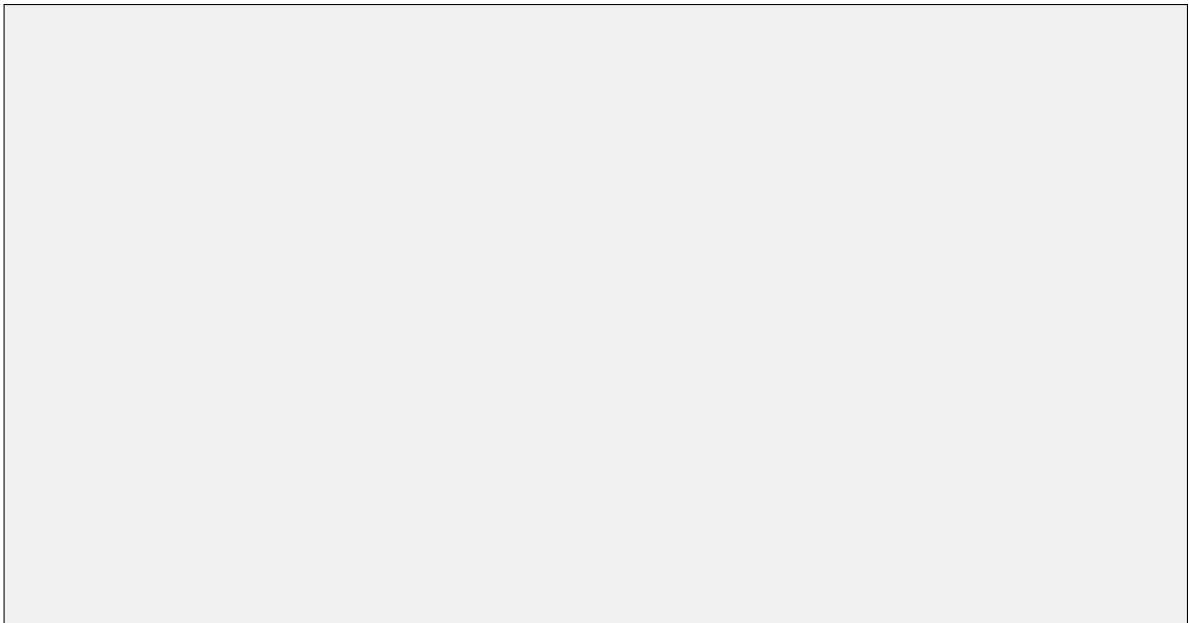
$$A_{1,2} = 2, A_{1,3} = 8, A_{2,1} = 0.25, A_{2,3} = 4, A_{3,1} = 0.5, A_{3,2} = 0.5$$



- (c) Was ist der schnellste Ihnen bekannte Algorithmus, um das Problem aus (a) zu lösen und wie wenden Sie diesen an? Geben Sie die asymptotische Laufzeit für den schlechtesten Fall an.

What is the fastest algorithm you can think of to solve the task described in (a), and how do you apply it? What is its asymptotic runtime complexity?

/3P



- /1P (d) Wie bestimmt man den maximalen erreichbaren Gewinn?

How can you find the maximal profit obtainable?

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question (d).

- /4P (e) Wie passen Sie Ihre Lösung an, wenn Sie nur k Devisenwechsel durchführen dürfen?

How can you adapt your solution if you are allowed to make only k currency exchanges?

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question (e).

Seite bewusst freigelassen / *Page left free intentionally*

Aufgabe 5: Writing with cubes (13P)



Sie haben N 6-seitige Wüfel und jeder Wüfel ist auf jeder seiner Seiten mit einem grossen Buchstaben bedruckt. Ist es möglich, ein Wort aus $K \leq N$ Buchstaben zu bilden, indem man einige der N gegebenen Wüfel aneinanderlegt? Von jedem der Wüfel darf also maximal einer der aufgedruckten Buchstaben benutzt werden.

Beispiel: $N = 3$, Wüfel 1 hat Buchstaben 'ABRACA', Wüfel 2 hat Buchstaben 'DABRAC' und Wüfel 3 hat Buchstaben 'SIMSAL'. Das Wort 'MAD' kann gebildet werden ('M' von Wüfel 3, 'A' von Wüfel 1 und 'D' von Wüfel 2), das Wort 'SIM' kann hingegen nicht gebildet werden.

- /2P (a) Beschreiben Sie ganz kurz (!) einen Brute-Force-Algorithmus zur Lösung des Problems und geben Sie dessen asymptotische Laufzeit im schlechtesten Fall an.

You have N 6-sided cubes, and each cube has a capital Latin letter on each of its sides. Is it possible to write a given word of $K \leq N$ letters by ordering some of the N cubes in a row? Thus, from each of the provided cubes at most, one letter can be used.

Example: $N = 3$: cube 1 has letters 'ABRACA', cube 2 has letters 'DABRAC', cube 3 has letters 'SIMSAL'. The words 'MAD' can be written (with 'M' from the third cube, 'A' from the first cube, and 'D' from the second cube), while the word 'SIM' cannot be written.

Briefly (!) describe a brute-force algorithm to solve this problem and provide the asymptotic worst-case runtime.

- (b) Das Problem lässt sich effizient lösen, indem man es als Problem des maximalen Flusses modelliert. Modellieren Sie das Problem als Flussproblem und geben Sie das Flussnetzwerk informell an.

The problem can be solved efficiently when it is modeled as a max-flow problem. Model the general problem as a flow problem. Describe the construction of the flow-graph informally.

/4P

- (c) Geben Sie das Flussnetzwerk an für das gesuchte Wort 'ETHZ' und die wie folgt gegebenen Würfel $c_1 \dots c_4$. Finden Sie dann den maximalen Fluss in dem Network.

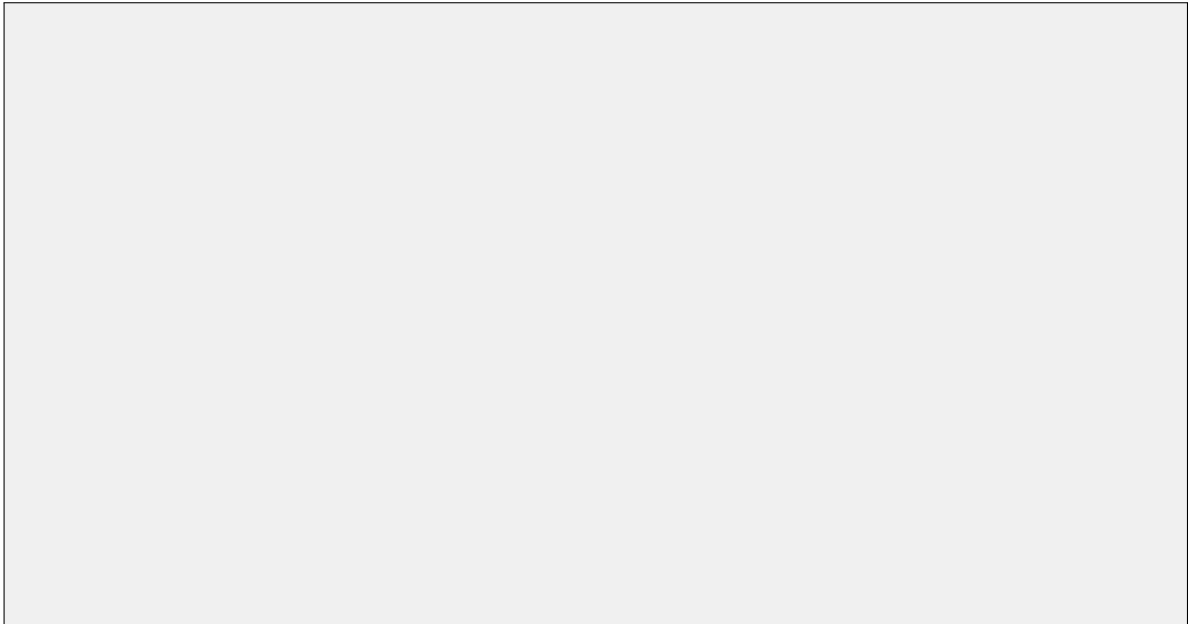
Provide the flow network for the word 'ETHZ' and the cubes $c_1 \dots c_4$ given as follows. After that, find the maximum flow in that network.

/3P

$c_1 = \text{'ETHABC'}$, $c_2 = \text{'EZABCD'}$, $c_3 = \text{'HABCXY'}$, $c_4 = \text{'ZFLIXW'}$

- /4P (d) Beschreiben Sie einen möglichst effizienten Algorithmus, mit dem Sie bestimmen können, ob das gesuchte Wort dargestellt werden kann. Geben Sie die asymptotische Laufzeit in Abhängigkeit von N und K an.

Describe an efficient algorithm that you can use in order to determine if the word in question can be written with the cubes. Provide the asymptotic running time of your algorithm as a function of N and K .



Aufgabe 6: Parallele Programmierung (18P)

- (a) Ein gieriger Scheduler auf einer idealisierten parallelen Maschine löst dieselbe Aufgabe mit 4 Prozessoren in 260 Sekunden und mit 32 Prozessoren in 90 Sekunden. Ist es plausibel, dass die zugrundeliegende Arbeit $T_1 = 1024$ und der kritische Pfad $T_\infty = 64$ ist? Begründen Sie Ihre Antwort.

A greedy scheduler on an ideal parallel machine runs a multithreaded computation in 260 seconds on 4 processors and in 90 seconds with 32 processors. Is it plausible that the computation has work $T_1 = 1024$ and the critical path is $T_\infty = 64$? Justify your answer.

/4P

- /4P (b) Betrachten Sie den folgenden Pseudocode für einen parallelisierten Divide-&-Conquer Algorithmus:

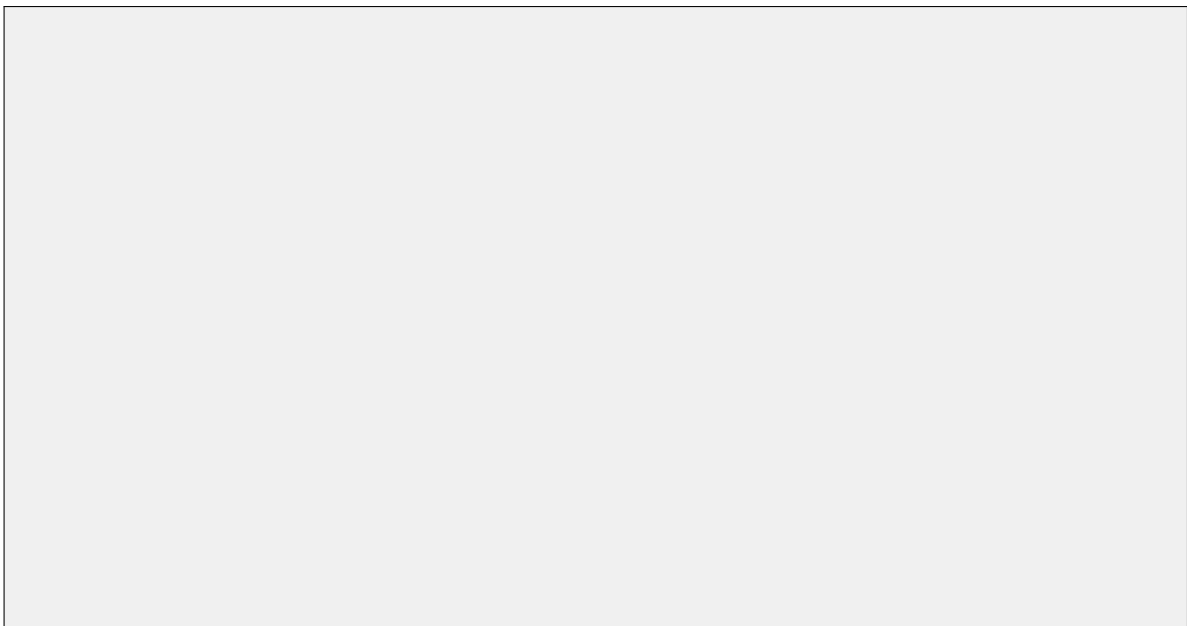
```
void ComputeProblem(n){
    if (n == 1)
        compute data point; // in Theta(1)
    else
        // divide
        thread t1 (ComputeProblem(n/4));
        thread t2 (ComputeProblem(n/4));
        thread t3 (ComputeProblem(n/4));
        thread t4 (ComputeProblem(n/4));
        t1.join(),t2.join(),t3.join(),t4.join() // in Theta(1)
}
```

Consider the following pseudocode for some parallelised divide and conquer algorithm.

Berechnen Sie die Arbeit T_1 , die Zeitspanne T_∞ und den bestmöglichen relativen Speedup (=Parallelität) des Algorithmus in Abhängigkeit von n . Gehen Sie davon aus, dass n eine Potenz von 4 ist. Verwenden Sie die Θ -Notation zur Abstraktion von Konstanten.

Tipp: schreiben Sie die Rekurrenzen für T_1 und T_∞ hin und / oder zeichnen Sie sich den Rekurrenzbaum der Berechnung auf. Sie müssen die Berechnung der Rekurrenzen nicht im Einzelnen zeigen.

Compute the work T_1 , the span T_∞ and the best possible relative speedup (parallelism) of the algorithm depending on n . Assume that n is a power of 4. Use Θ -Notation to abstract away constants. Hint: write down the recurrences for T_1 and for T_∞ and / or draw a recurrency tree of the computation. You do not need to show the computation of the recurrences in detail.



- (c) Professor Findig verändert den Code der vorigen Teilaufgabe in folgender Weise.

Professor Findig modified the code of the previous task in the following way.

/4P

```
void ComputeProblem(n){
    if (n == 1)
        compute data point; // in Theta(1)
    else
        // divide
        thread t1 (ComputeProblem(n/4));
        thread t2 (ComputeProblem(n/4));
        t1.join(); t2.join(); // in Theta(1)
        thread t3 (ComputeProblem(n/4));
        thread t4 (ComputeProblem(n/4));
        t3.join(); t4.join(); // in Theta(1)
}
```

Berechnen Sie erneut die Arbeit T_1 , die Zeitspanne T_∞ und den bestmöglichen relativen Speedup (=Parallelität) des Algorithmus in Abhängigkeit von n . Gehen Sie davon aus, dass n eine Potenz von 4 ist. Verwenden Sie die Θ -Notation zur Abstraktion von Konstanten.

Again compute the work T_1 , the span T_∞ and the best possible relative speedup (parallelism) of the algorithm depending on n . Assume that n is a power of 4. Use Θ -Notation to abstract away constants.

Wir verwenden im folgenden Teil der Aufgabe C++ Code und nehmen an, dass Sie die jeweilige Syntax und Semantik verstehen. Es geht um das theoretische Verständnis der nebenläufigen Vorgänge. Treffen Sie insbesondere keine aussergewöhnliche Annahmen über die verwendete Architektur, das Speichermodell oder das Verhalten des Compilers.

Es werden Threads ausgeführt. Wir gehen jeweils davon aus, dass die Funktion print jeweils einen Buchstaben ausgibt. Geben Sie für die Programme jeweils alle möglichen Ausgaben an, allenfalls getrennt durch Semikolon. Bestimmen Sie, ob das Programm terminiert. Wenn das Programm nicht terminiert, geben Sie alle Ausgaben immer so weit an, wie sie auftreten können.

For the following part of the task, we use C++ code and assume that you understand the syntax and semantics. But this task is more about the theoretical understanding of what can happen in a concurrent setup. Particularly do not make any exceptional assumptions about the architecture used, the memory model, or the behavior of the compiler.

Threads are executed. We assume for each part that the function print outputs a character. For each of the programs, provide all possible outputs, separated by semicolons if necessary. Specify if the program terminates. If the program does not terminate, provide all possible outputs as far as they can occur.

/2P (d)

```
void print(char c); // output character c
```

```
void A(char value){
    ++value;
    print(value);
}
```

```
int main(){
    char value = 'A';
    std::thread t1(A,value);
    std::thread t2(A,value);
    t1.join();
    t2.join();
    print(value);
}
```

mögliche Ausgabe(n)/*possible output(s)*

Das Programm terminiert/*the program terminates*

immer/*always* nie/*never* manchmal/*sometimes*

(e)

/2P

```
void print(char c); // output character c

char value; // global variable accessible by all threads

void A(){
    ++value;
    print(value);
}

int main(){
    value = 'A';
    std::thread t1(A);
    std::thread t2(A);
    t1.join();
    t2.join();
}
```

mögliche Ausgabe(n)/*possible output(s)*

Das Programm terminiert/*the program terminates*

- immer/*always* nie/*never* manchmal/*sometimes*

/2P (f)

```
void print(char c); // output character c

using guard = std::unique_lock<std::mutex>;
std::mutex m;
std::condition_variable c;

char value;

void A(){
    guard g(m);
    print(value);
    ++value;
    c.wait(g, [&]{return value == 'B';});
    ++value;
    print(value);
}

int main(){
    value = 'A';
    std::thread t1(A);
    std::thread t2(A);

    t1.join();
    t2.join();
}
```

mögliche Ausgabe(n)/*possible output(s)*Das Programm terminiert/*the program terminates* immer/*always* nie/*never* manchmal/*sometimes*

/18P

Aufgabe 7: Programmieraufgabe: Reentrant Lock (18P)

Diese Aufgabe zum Thema Parallele Programmierung soll am Computer gelöst werden. Sie können sich die Zeit frei einteilen. Wir **empfehlen** aber, dass Sie **nicht mehr als 45 Minuten** für diese Aufgabe aufwenden.

Lösen Sie diese Aufgabe in der Online-Umgebung (Code Expert via Moodle).

*This following part on parallel programming needs to be solved at the computer. You are free in how you divide the time. However, we **recommend to spend not more than 45 minutes** on that problem. Solve this task in the online environment (Code Expert via Moodle).*

Moodle-Passwort wird zu Beginn der Prüfung bekannt gegeben / *Moodle password will be announced at the beginning of the exam*