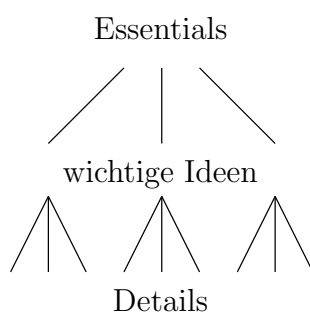


# Infos zur Prüfungsvorbereitung

## Prüfungen

- alles aus dem Skript kann drankommen (genauer: alles aus den Kapiteln, die in der Vorlesung behandelt wurden!)
- Vorbereitungsstrategie  
für jedes Unterkapitel einen Kurzvortrag einüben (ca. 4 Min.) „hierarchisch“ lernen



Zu einem algorithmischen Problem gehören

- Problemstellung
- Motivation
- Lösungsstrategie
- die „neuen“ oder „originellen“ Ideen
- der Algorithmus
- der Korrektheitsbeweis
- die Analyse  
auch der Ansatz zu den Rechnungen, aber nicht die Rechnungen
- die Angabe der Laufzeit

Die Schwerpunkte sind von Problem zu Problem verschieden!

## Kontrollstrategie

- erstelle Stapel von Karteikarten mit allen Unterkapiteln / Algorithmen / Datenstrukturen / Strategien / ...
- mische den Stapel
- ziehe Karteikarte zufällig
- beantworte die Frage laut und in vollständigen Sätzen und notiere, was du aufschreiben würdest
- miss die Antwortzeit
- mache dies möglichst oft mit jemanden, die oder der zuhört und möglichst etwas davon versteht.
- Prüfungssituation so realistisch wie möglich simulieren

## Eine kleine Auswahl möglicher Fragen

1. Was ist das Maxsummenproblem und wie löst man es mit dynam. Programmierung? Wie ist die Laufzeit und wieso ist sie so?
2. MAXMIN: Worum geht es in dem Problem? Entwirf einen guten Turnierplan dafür und zeige, dass er optimal ist.
3. Was ist der Unterschied zwischen der uniformen und der logarithmischen Zeitkomplexität?
4. Warum wird meistens die worst case und nicht die average case Rechenzeit untersucht?
5. Bei QUICKSORT gibt es dieselbe average case Rechenzeit bei Wahl des ersten Elements als Pivotelement und bei Wahl eines zufälligen Elements. Was ist der Unterschied zwischen den beiden „average cases“?
6. Definiere  $O, \Omega, \Theta, o, \omega$  und warum ist ihre Verwendung sinnvoll?
7. Sortiere nach Größenordnung:  $n^2 \log n, n^2 \log^2 n / (\log \log n)^{10}, 2^{n^{1/10}}, 2^{n/100}, n^2 + 3n^{3/2}, n^{15/8}, n^2 / \log n, n^{\log n}$ .
8. Beschreibe die Strategien lineare Suche, binäre Suche und geometrische Suche mit Vor- und Nachteilen.
9. Vergleiche lineare Listen und Arrays bzgl.
  - Suche nach Datum an Position  $n$

- Suche nach Datum  $x$
  - Einfügen von  $y$  hinter  $x$ , nachdem  $x$  gefunden wurde
10. Gib Beispiele an, bei denen es sinnvoll ist,
    - doppelte Verkettung zu haben
    - Zeiger auf das letzte Element zu haben
    - die Größe der Listen zu verwalten
  11. Topologisches Sortieren bei gerichtetem Graphen: Worum geht es dabei? Gib einen Algorithmus an, der das Problem löst.
  12. Vergleiche Bitvektordarstellung und geordnete Listen zur Verwaltung von Mengen
  13. Beschreibe DFS für gerichtete Graphen
  14. Segmentbäume
    - wofür
    - wie
    - Rechenzeit
  15. UNION-FIND mit Arrays und Listen  
Darstellung der Struktur, FIND, UNION, Analyse
  16. UNION-FIND mit Bäumen  
Darstellung der Struktur, max. Baumhöhe bei  $n$  Daten, Pfadkomprimierung
  17. Welche Operationen werden zur Verifikation von Schaltkreisen benötigt?
  18. OBDDs – Darstellung und Minimierungsregeln
  19. OBDDs – Synthese mit integrierter Reduktion / Minimierung
  20. Wie sieht ein OBDD für die Addition / den Multiplexer aus?
  21. Was sind dynamische Dateien?
  22. Vergleiche offenes und geschlossenes Hashing
  23. Beschreibe Strategien zur Kollisionsbehandlung, wieso unterscheiden sie sich in ihrem Verhalten überhaupt?
  24. Was versteht man unter idealem Hashing? Wie sieht dabei der Ansatz zur Berechnung der erwarteten Suchzeit aus?
  25. Was sind Suchbäume? SEARCH, INSERT, DELETE

26. Durchschnittliche Tiefe binärer Suchbäume mit  $n$  Daten, die in zufälliger Reihenfolge kommen (Ansatz + Ergebnis asymptotisch)
27. 2-3-Bäume  
Definition, min. und max. Tiefe, SEARCH, INSERT, DELETE  
(hier wie auch sonst: Erklärung nicht nur am Beispiel, sondern strukturell, aber auch Behandlung von Beispielen möglich)
28. Das gleiche für B-Bäume
29. Das gleiche für AVL-Bäume, insbes. Ansatz zur Berechnung der Maximaltiefe bei  $n$  Daten.  
Wann welcher Typ der Rotation?
30. Skiplisten: Struktur, durchschnittl. Höhe, Operationen, Analysetechnik und Anwendung
31. Liste die bekannten Sortieralgorithmen mit Vor- und Nachteilen auf
32. Quicksort
  - Wie kommt das Zerlegungsdatum an die richtige Position?
  - Strategien zur Wahl des Zerlegungsdatums
  - worst case
  - Ansatz zur average case Analyse und Ergebnis
33. Heapsort
  - Darstellung des Heaps im Array
  - das Rahmenprogramm
  - die reheap-Strategien
  - Warum ist „bottom-up“ besser?
  - Warum ist die HeapCreation Phase effizienter als die Selection Phase?
  - Analyse der worst case Rechenzeiten der verschiedenen Strategien
34. Was ist ein allgemeines Sortierverfahren?  
Untere Schranken für worst und average case
35. Bucketsort: Verfahren und Analyse
36. Beschreibe effiziente Algorithmen für das Auswahlproblem mit Analyseansatz
37. Batcher sort: Wie geht es und warum klappt es?  
Sequentielle und parallele Rechenzeit
38. Beschreibe die folgenden Optimierungsstrategien allgemein:

- Greedy Algorithmen
  - Dynamische Programmierung
  - Divide-and-Conquer
  - Branch-and-Bound
39. Gib Beispiele für Greedy-Algorithmen an, die
    - stets das Optimum berechnen
    - beliebig schlechte Ergebnisse berechnen können
    - nicht immer optimale, aber nie schlechte Ergebnisse berechnen
  40. Algo von Kruskal: wozu, wie, Datenstrukturen, Rechenzeit, immer optimal.
  41. Dynamische Programmierung: Was sind Probleme vom Intervalltyp?
  42. Berechnung statischer Suchbäume mit minimaler erwarteter Zugriffszeit
  43. Was sind pseudopolynomielle Rechenzeiten? Gib ein Beispiel an (Dyn. Progr. für das Rucksackproblem)
  44. Dynamische Programmierung für das APSP.
  45. Algorithmus von Dijkstra: Strategie, Korrektheit, Implementierung, Analyse, Datenstruktur
  46. Die Module von Branch-and-Bound Algorithmen am Beispiel des Rucksackproblems
  47. Divide-and-Conquer
 
$$R(n) = a \cdot R(n/b) + cn$$
 Wie wirken sich  $a$  und  $b$  auf die Rechenzeit aus? Und wie  $c$ ?
  48. Welche Divide-and-Conquer Algos wurden behandelt?
  49. Welche Algos der dyn. Programmierung wurden behandelt?
  50. Beschreibe die wesentlichen Ideen des Strassen-Algorithmus
  51. FFT – wozu? Wie funktioniert es?
  52. Berechnung der nächsten Nachbarn in der Ebene
  53. Was ist die Sweepline-Technik?
  54. Anwendung auf das Rechteckmaßproblem  
Wie wird der Segmentbaum verwaltet?
  55. Beschreibe das  $\alpha$ - $\beta$ -Pruning

56. Was ist das Szenario der Black-Box-Optimierung?  
Wie arbeiten randomisierte Suchheuristiken allgemein?
57. Beschreibe
- randomisierte lokale Suche
  - Simulated Annealing
  - evolutionäre Algorithmen
58. Welches Ergebnis der Vorlesung hat dir warum am besten gefallen / hat dich warum am meisten überrascht?

In Klausuren können auch Beispiele zu behandeln sein!  
Keine Aussagen ohne Begründung!