Algorithm Engineering 3
Generische Vorgehensweise

Karsten Weicker

F IMN, HTWK Leipzig

Anwendungsbeispiel

Prüfungsplanung

- Geg.: Prüfungen mit involvierten Matrikeln und Dozenten
- Ges.: Zuordnung zu 15 Prüfungstagen ohne „Kollisionen“

Randbedingungen

- jedes Matrikel: max. 3 Prüfungen pro Woche
- Sperrzeiten von Dozenten
- ...

1. Anwendungsbeispiel
2. How to Design Algorithms
   - Verstehe ich mein Problem?
   - Gibt es einen einfachen Algorithmus oder eine Heuristik?
   - Kann ich das Problem auf ein Standardproblem abbilden?
   - Gibt es Spezialfälle des Problems, die ich lösen kann?
   - Welches Entwurfsparadigma passt zu meinem Problem?
   - Hilft Randomisierung oder Approximation?

Verstehe ich mein Problem?

1. Wie sehen genau meine Eingabedaten aus?
2. Wie sieht genau das gewünschte Ergebnis aus?
3. Kann ich ein kleines Beispiel konstruieren, das ich von Hand lösen kann? Was passiert, wenn ich es zu lösen versuche?
4. Welche Qualität wird von einer Antwort erwartet? (Optimum? Reicht fast immer? Maximaler Fehler?)
5. Wie groß sind typische Instanzen meines Problems? 10? 1000? 1 Million?

Verstehe ich mein Problem?

6. Wie wichtig ist Geschwindigkeit in meiner Anwendung? Muss das Problem in 1 Sekunde gelöst sein? Oder 1 Minute, 1 Stunde, 1 Tag?
7. Wieviel Zeit und Aufwand kann ich in die Implementation investieren?
8. Ist es ein numerisches Problem? Ein Graphproblem? Ein geometrisches Problem? Ein Textproblem? Ein Mengenproblem? Welche Formulierung ist am einfachsten?

Gibt es einen einfachen Algorithmus oder eine Heuristik?

1. Kann Brute-Force mein Problem korrekt durch Durchsuchen aller Teilmengen/Anordnungen/... lösen?
   - Bin ich sicher, dass der Algorithmus immer die richtige Antwort liefert?
   - Wie messe ich die Qualität einer so konstruierten Lösung?
   - Welche Laufzeit ergibt sich? Polynomiell? Exponentiell? Sind meine Probleminstanzen klein genug?
   - Ist mein Problem überhaupt genau genug definiert, dass es eine korrekte Lösung gibt?

Gibt es einen einfachen Algorithmus oder eine Heuristik?

2. Kann ich das Problem durch eine iterativ angewandte einfache Regel lösen – wie zum Beispiel gierig das Maximum/Minimum/. . . erst wählen?
   - Falls ja, auf welchen Eingabetypen arbeitet diese Heuristik gut? Passt es für meine Anwendung?
   - Auf welchen Eingabe arbeitet sie schlecht? Wenn ich kein Beispiel finde, kann ich beweisen, dass die Heuristik immer gut arbeitet?
   - Wie schnell ist meine Heuristik? Gibt es eine einfache Implementation?

Kann ich das Problem auf ein Standardproblem abbilden?

1. Ist es im Katalog `http://www.cs.sunysb.edu/~algorith` enthalten? Passt eine dort beschriebene Lösung?
2. Gibt es dort ein verwandtes Problem? Kann ich Randbedingungen etc. nutzen?
3. WWW? Google scholar search?

Gibt es Spezialfälle des Problems, die ich lösen kann?

1. Wird das Problem effizient lösbar, wenn ich einige Eingabeparameter nicht berücksichtige?
2. Wird das Problem einfacher, wenn ich einige Eingabeparameter auf triviale Werte (z.B. 0 oder 1) setze?
3. Kann ich das Problem so vereinfachen, dass es effizient lösbar ist?
4. Warum kann ein solcher Algorithmus für den Spezialfall nicht generalisiert werden?
5. Ist mein Problem ein Spezialfall eines allgemeineren Problems?

Welches Entwurfsparadigma passt zu meinem Problem?

1. Hilft eine Sortierung von Elementen im Problem bei der Lösung des Problems?
2. Gibt es einen Weg, das Problem in kleinere Probleme zu teilen – durch binäre Suche, groß/klein, links/rechts, . . . ? Folgt ein Divide-and-Conquer-Algorithmus?
3. Haben die Eingabeobjekte eine natürliche Ordnung? Z.B. wie Zeichenketten, Permutationen, Blätter eines Baums? Kann Dynamisches Programmieren diese Ordnung nutzen?

Welches Entwurfsparadigma passt zu meinem Problem?

4. Werden Operationen wiederholt durchgeführt? Kann eine Datenstruktur diese beschleunigen?
5. Kann zufälliges Sampling nützlich sein? Bei der Wahl eines Elements? Zufällige Konfigurationen generieren und mit der besten weiterarbeiten? Hilft gerichtete Zufälligkeit wie in Simulated Annealing oder EAs?

Welches Entwurfsparadigma passt zu meinem Problem?

6. Kann ich mein Problem als lineares Programm oder Integer-Programm formulieren?
7. Ist mein Problem NP-vollständig? Ähnlichkeit zu SAT oder TSP? Listen im Internet/Büchern geprüft? Dann existiert kein effizienter exakter Algorithmus

Hilft Randomisierung oder Approximation?

1. Kann durch Randomisierung der Worst-Case-Fall unwahrscheinlicher werden?
2. Sind Falschaussagen mit einer gewissen Wahrscheinlichkeit eine Option? Randomisierung?
3. Ist ein Fehler bei einer Approximation eine Option? Gibt es entsprechende Algorithmen in der Literatur? Helfen Sonderfälle?