Informatik II

Übungsstunde 11

simon.mayer@inf.ethz.ch
Distributed Systems Group, ETH Zürich

Serie 10

- Mergesort
- Hanoi... ☺
- Alpha/Beta Spieler

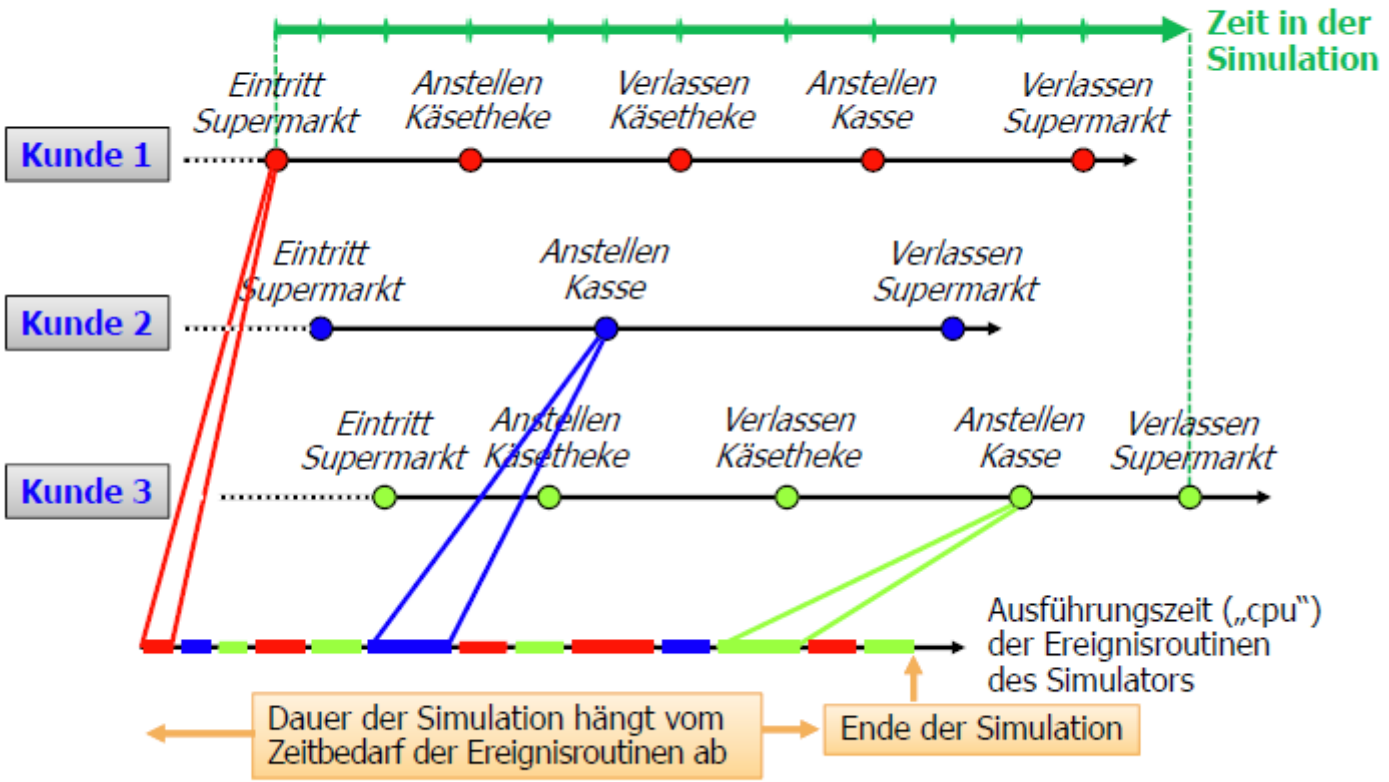
Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Simulation, Heaps
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

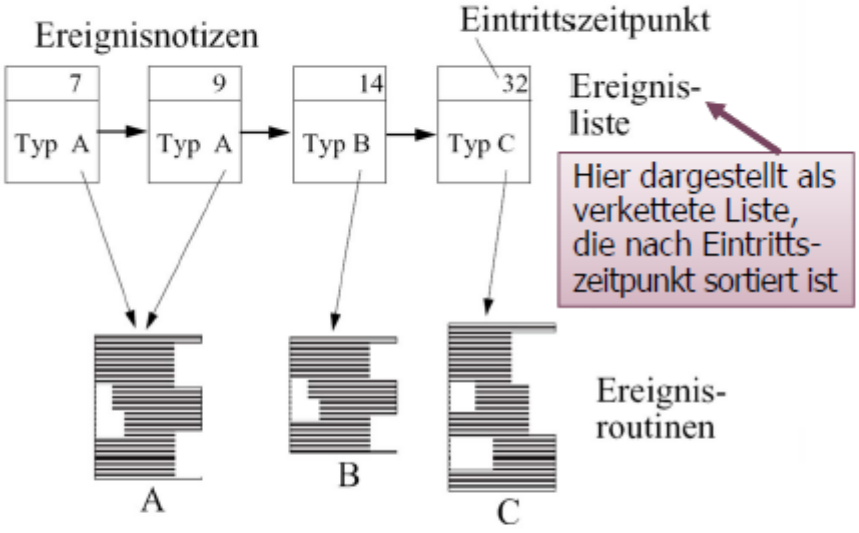
Quasi-Parallelismus

- In der Realität gleichzeitig ablaufende Aktivitäten werden in der Simulation „stückweise" sequentialisiert
- Durch die Auflösung in Ereignisse (am Anfang und Ende von Teilaktivitäten) werden Aktivitäten zeitlich verzahnt
- Beispiel: Zwischen Ankunftsereignis und Abgangsereignis eines Supermarktkunden (also während dieser mit Einkaufen als „Aktivität" beschäftigt ist) betreten weitere Kunden den Supermarkt

Quasi-Parallelismus: Beispiel



Ereignisverwaltung



Ereignisnotizen benennen den Eintrittszeitpunkt und die Ereignisroutine (d.h. den Ereignistyp) eines eingeplanten (d.h. noch auszuführenden) Ereignisses

Die Ereignisliste speichert alle Ereignisnotizen und liefert auf Anforderung die früheste

Ereignisroutinen enthalten Anweisungen zur Änderung des Modellzustands (und evtl. zum Einplanen weiterer Ereignisse, d.h. Einfügen neuer Ereignisnotizen in die Ereignisliste)

Implementierung von priority queues

2. Implementierung: Unsortierte verkettete Liste

- → insert benötigt O(1) Schritte
  (z.B. vorne anfügen)
- → get_min benötigt O(n) Schritte
  (kleinstes Element suchen und ausketten)

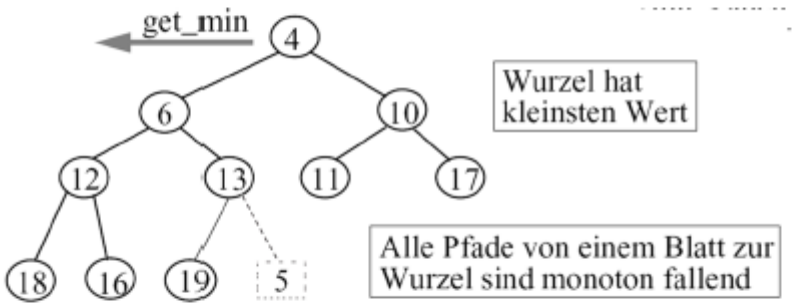
---

3. Implementierung: Heap-Datenstruktur ← „partiell" sortiert

- → insert und get_min benötigen *beide* nur O(log n) Schritte

Die Heap-Datenstruktur

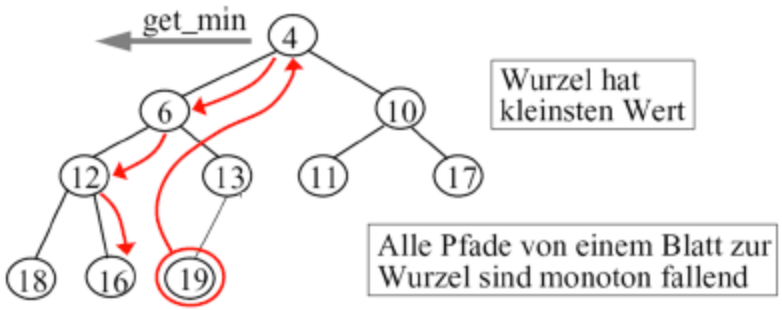
- Heap = Binärbaum mit
  - Alle Niveaus (bis evtl. auf das letzte) sind vollständig besetzt
  - Für alle Knoten k ≠ Wurzel: Wert(Vorgänger(k)) < Wert(k)
    (alternative Def. mit ≤, >, ≥ auch möglich)



Min-Heap

- Zwei Operationen: insert und get_min

Heap: get_min und insert



Man überlege sich genau, dass das der Fall ist!

Für get_min *und* insert gilt:
- Lässt Heap-Eigenschaft invariant
- Benötigt O(log n) Schritte

Im Unterschied zur Implementierung von priority queues als sortierte / unsortierte verkettete Liste!

get_min:

- Wurzel entfernen
- Letzten Knoten des untersten Niveaus an die Wurzelposition setzen
- Neue Wurzel so weit wie möglich nach unten sinken lassen: Mit kleinerem der beiden Nachfolger vertauschen; dies rekursiv (oder iterativ) auf entsprechenden Unterbaum anwenden

insert:

- Als neues Blatt auf unterstem Niveau einfügen
- Soweit wie möglich nach oben wandern lassen: Iterativ mit Vorgänger vertauschen, wenn dieser grösser

Heaps

- Wir haben ein Ding, in das ich in O(log n) Dinge einfuegen kann und dann in O(log n) jeweils das kleinste Element finden kann...

- Irgendwelche Anwendungsideen?

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Simulation, Heaps
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Übung 10

1. Sortieren mit Suchbaeumen
   Koennt ihr schon...
   Bisschen Gedanken darueber machen...

2. Komplexitaet/O-Notation

3. Komplexitaet

Übung 10

Und ein Springer auf dem Schachbrett...

- Berechne alle Felder, die ausgehend von einer Startposition in maximal x Schritten erreicht werden koennen
- Berechne einen moeglichst langen Pfad, wobei jedes Feld nur einmal besucht wird
  - Maximal: Jedes Feld einmal besucht

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Simulation, Heaps
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Informatik II

Übungsstunde 11

simon.mayer@inf.ethz.ch
Distributed Systems Group, ETH Zürich