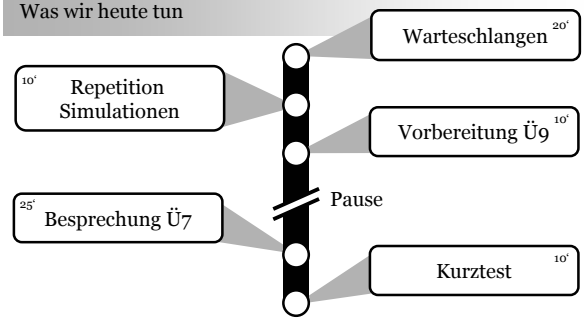
- Übungsserie 8 jetzt abgeben, wenn Ihr wollt
- Alle haben jetzt das Testat

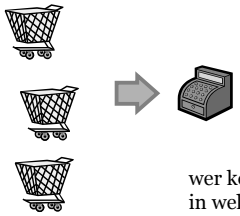
Was wir heute tun



Einkaufen im Migros

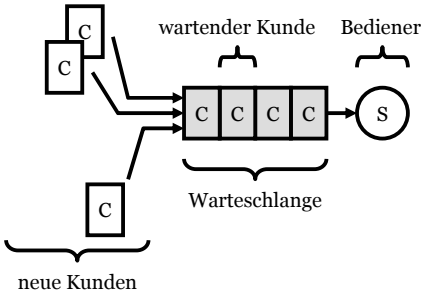
Einkaufswagen  Kasse 

Mehrere Einkaufswagen, aber nur eine Kasse



wer kommt zuerst dran?
in welcher Reihenfolge?

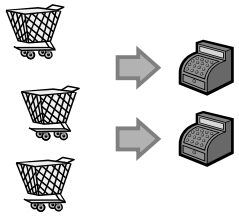
Das Prinzp der Warteschlange



Kunde = Client
Bediener = Server
Warteschlange = Queue

Neue Kasse wird geöffnet

plötzlich zwei Bediener

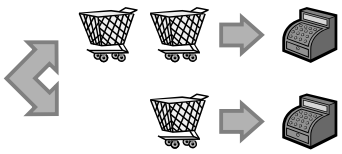


was passiert jetzt?

Was in Europa passiert

„Least Queuing“

die Leute rennen wild herum und
es bilden sich für jede Kasse eine Warteschlangen



als neu Ankommender geht man
zur kürzesten Warteschlange

Was in Amerika passiert

„amerikanisches System“

die Leute bleiben in einer Warteschlange und
der vorderste geht zur jeweils freien Kasse

ist alles, was aus Amerika kommt, besser?

Zeiten

*Die **Wartezeit** ist die Zeit, bis der Kunde bedient wird*

*Die **Bedienzeit** ist die Zeit, während der Kunde bedient wird*

*Die **Verweilzeit** ist die Zeit, bis der Kunde fertig bedient wurde; also Wartezeit plus Bedienzeit*

Was ist eine Simulation?

Simulation = Durchführung eines Vorganges in einem gewählten Modell

Echte Welt –
Auto fährt in einen Baum

Simulation –
mit dem Modell der Physik

Auto: v = 4 m/s
Material: Metall, stabil
Form: eckig, sperrig

Formel für Stoss:
$\frac{2m_2}{m_1+m_2}\vec{v}_2+\frac{m_1-m_2}{m_1+m_2}\vec{v}_1$

Baum: v = 0 m/s
Material: Holz, weich
Form: lang, rund

Simulieren auf dem Computer

Unsere Welt ist die Erde
Als Welt der Simulation nehmen wir Oberon

Auf der Erde werden Autos gebaut
In Oberon macht das eine Prozedur

```
PROCEDURE FabriziereAuto( G: Grösse; M: Material );
```

Auf der Erde wachsen Bäume

```
PROCEDURE PlaziereBaum( B: Baum; X, Y: LONGINT);
```

Auf der Erde wird herumgefahren

```
PROCEDURE PlaziereAuto( A: Auto; X, Y: LONGINT);
PROCEDURE BeschleunigeAuto( V: Vector );
```

Was passiert, wenn man auf einen Baum beschleunigt?
physikalisch ein gerader zentraler Stoss $\frac{2m_2}{m_1+m_2}\vec{v}_2+\frac{m_1-m_2}{m_1+m_2}\vec{v}_1$

```
2*M2 DIV (M1+M2) * V2 + (M1-M2) DIV (M1+M2) * V1;
```

Das Modell einer Simulation

Wie sich die Welt der Simulation verhalten soll, beschreibt das Modell

Wann aber läuft das Modell ab?

Bis jetzt sind nur Bäume & Autos plaziert!

Ausgangslage; eine Art Eingabe
Jetzt können wir also die virtuelle Welt laufen lassen!
Dazu braucht man eine Prozedur

```
PROCEDURE SimulationStarten()*;
```

In dieser Prozedur ist das Modell umgesetzt:

- Autos fahren herum
- Bäume stehen herum
- Wenn ein Auto irgendwo durchfährt, wo ein Baum steht, gibt es einen Zusammenstoss

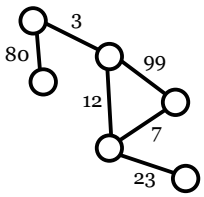
Vorbereitung Übungsserie 8

Aufgabe 1. Simulation von Warteschlangen.
Wir betrachten ein Serversystem mit Zufallsankunft der Kunden (Exponentialverteilung) und Zufallsbedienzeit (zufällige Auswahl aus einer diskreten Menge möglicher Bedienzeiten) durch die Servers. Erstellen Sie je ein Oberonprogramm zu Simulation der folgenden beiden Systemvarianten und vergleichen Sie die Leistung (mittlere Verweilzeit der Kunden).

- Zwei gleiche Servers, zwei Warteschlangen (mit "Least Queuing")
  `PROCEDURE EuroKasseOeffnen();`
- Zwei Servers, eine Warteschlange ("amerikanisches System")
  `PROCEDURE AmerikanischeKasseOeffnen();`

Notizen

Minimum Spanning Tree - minimaler spannender Baum



Graphen mit
Gewichten auf den Kanten

„Ortschafen mit Strassenverbindungen; die Gewichte geben die Länge der Strassen an“

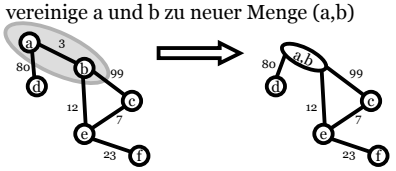
ein minimaler spannender Baum T für Graph G besteht aus allen Knoten V von G, enthält aber nur eine Teilmenge E‘ der Kantenmenge E von G, die alle Knoten des Graphen miteinander verbindet und die Eigenschaft hat, dass die Summe aller Kantengewichte den minimal möglichen Wert hat unter allen Teilmengen von E, die alle Knoten des Graphen G miteinander verbindet.

Minimum Spanning Tree finden

Kruskals Verfahren aus dem Jahre 1956

❶ sortiere Kanten in einer Liste nach Gewicht

❷ nimm kleinste Kante &
vereinige a und b zu neuer Menge (a,b)



❸
- wiederhole das mit allen Kanten, bei welchen zwei Mengen jeweils vereint werden können
- stoppe, wenn alle Knoten des Graphen verbunden sind

Kruskals Verfahren in Pseudocode

```
PROCEDURE Kruskal( g : GRAPH, VAR mst : LISTE )
    VAR kantenliste, resultat: LISTE;
BEGIN
    mst := leer;
    sortiere alle Kanten aus g in Liste kantenliste;
    FOR alle Knoten DO
        MakeSet( Knoten );
    END;
    WHILE noch mehrere Mengen vorhanden DO
        nimm kleinste Kante (v, w);
        entferne Kante aus kantenliste;
        IF Find(v) # Find(w) THEN
            Union( Find(v) , Find(w) );
            mst := mst & (v, w);
        END
    END
END Kruskal;
```

Besprechung Übungsserie 7 Aufgabe 1 – Teil 1

„Minimum Spanning Tree“

```
TYPE Kante = RECORD
     von, nach : INTEGER;
     gewicht : LONGINT;
END;
```

Anzahl Kanten im Graph $\sum_{i=1}^{n-1} i$ *oder* $\frac{1}{2}(-1+n)\,n$

```
VAR Graph : ARRAY 496 OF Kante; (* bei 32 Knoten *)
```

„Union-Find braucht Mengen“

```
Sets : ARRAY 32 OF SET;
```

Besprechung Übungsserie 7 Aufgabe 1 – Teil 2

„Graph aufbauen“

```
z := 0;

FOR k := 0 TO 30 DO

 FOR m := k + 1 TO 31 DO

  Graph[z].von = k;
  Graph[z].zu = m;
  Graph[z].gewicht = RandomNumbers.Uniform();
  INC(z);

 END;

END;
```

Besprechung Übungsserie 7 Aufgabe 1 – Teil 3

„Kruskals Algorithmus“

```
QuickSort;       (* Graph sortieren nach Gewicht *)
FOR i := 0 TO 31 DO     (* jeder ist eine kleine *)
 Sets[i] := {i}               (* Menge für sich *)
END;
kleinster := 0;
WHILE mstgroesse < 32 DO      (* bis alle Knoten *)
 k1 := Graph[kleinster].von;   (* kleinste Kante *)
 k2 := Graph[kleinster].zu;     (* herausgreifen *)
 INC(kleinster);          (* Kante herausnehmen *)
 IF Sets[k1] * Sets[k2] = {} THEN
  Sets[k1] := Sets[k1] + Sets[k2];      (* Union *)
  Sets[k1] := Sets[k1] + Sets[k2];
  (* Kante mit Out ausdrucken *)
  INC(mstgroesse);    (* neuer Knoten in MST tun *)
 END;
END;
```