



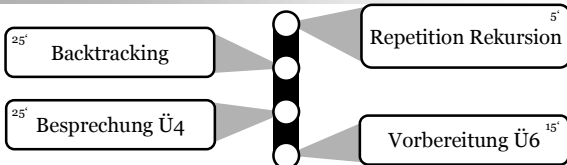
Übungsserie 5 abgeben

- Email, wenn Testatpunkt gewünscht
- Gedruckt, wenn Korrektur & Testatpunkt gewünscht

Lerntip

<http://www.educeth.ch/informatik/>

Was wir heute tun



Repetition Rekursion



Was ist Rekursion?

```

PROCEDURE Fakultael(n : INTEGER): INTEGER;
BEGIN
    IF n <= 1 THEN
        RETURN 1;
    ELSE
        RETURN n * Fakultael(n-1);
    END
END Fakultael;

```

Repetition Rekursion



Was passiert bei der Rekursion?

Ablauf bei Aufruf `a := Fakultael(3);`

```

→ ③ IF n <= 1 THEN (* n = 3 *)
    ELSE RETURN n * Fakultael(n-1); (* n = 3 *)
→ ② (* 1. rekursiver Aufruf Fakultael(2) *)
    IF n <= 1 THEN (* n = 2 *)
    ELSE RETURN n * Fakultael(n-1); (* n = 2 *)
→ ① (* 2. rekursiver Aufruf Fakultael(1) *)
    IF n <= 1 THEN (* n = 1 *)
        RETURN 1;
    ② ← zurück zur 1. rekursiven Prozedur Fakultael(2)
        RETURN 2 * 1;
    ③ ← zurück zur 2. rekursiven Prozedur Fakultael(3)
        RETURN 3 * 2;

```

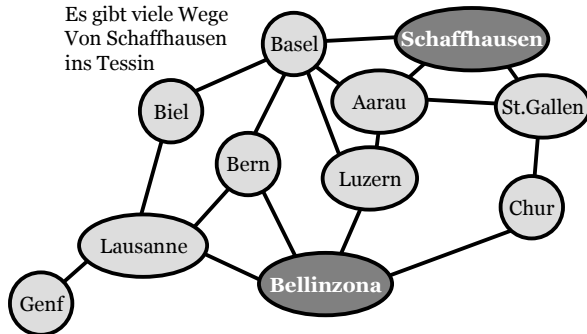
Am Schluss: `a := 6;`

Suche mit Backtracking



Wir suchen einen Weg von einem Ort zu einem Anderen!

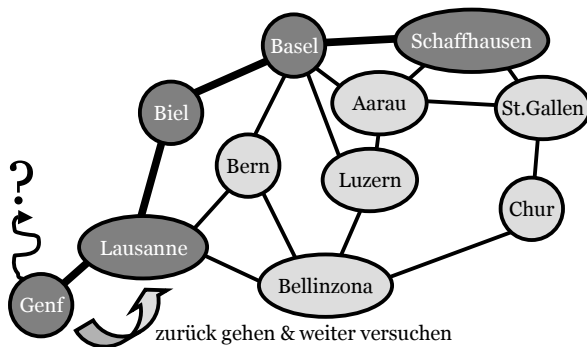
Es gibt viele Wege
Von Schaffhausen
ins Tessin



„zurückgehen“ – Backtracking



Naiv müssen wir alle Möglichkeiten durchtesten:



Backtracking & Heuristiken



Systematische Art der Suche in einem Graphen

Wenn eine Teillösung in eine Sackgasse führt,
geh zurück & versuche andere Wege

Hellsehen, ob Sackgasse kommt: Heuristiken!

*Heuristiken sind „Strategien, die mit höherer
Wahrscheinlichkeit (jedoch ohne Garantie) das
Auffinden einer Lösung beschleunigen sollen.“*

Heuristiken finden ist beinahe so schwer wie Hellsehen!

siehe auch http://www.educeth.ch/informatik/vortraege/backtracking/docs/backtracking_folien.pdf

Backtracking in Pseudocode



```

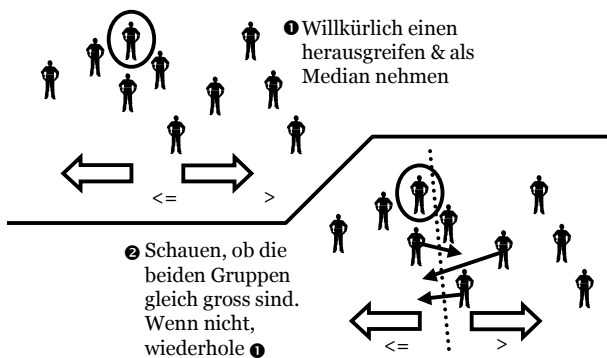
PROCEDURE findeLoesung( schritte : INTEGER,
VAR bisher : WEG ) : BOOLEAN;
WHILE es noch Teillösungen gibt DO
  wähle einen noch nicht getesteten schritt;
  IF schritt = erlaubt THEN
    bisher := bisher & schritt;
    IF bisher = vollständig THEN RETURN TRUE;
  ELSE
    IF findeLoesung( schritte + 1, bisher ) THEN
      RETURN TRUE;
    ELSE
      bisher := bisher ohne schritt;
    END;
  END;
END;
RETURN FALSE;

```

Besprechung Übungsserie 4 Aufgabe 1



Median Berechnung



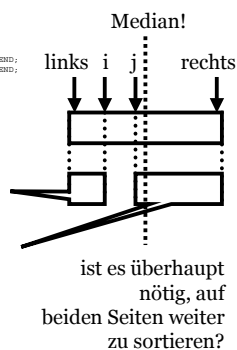
Komplettes Quicksort nötig?



```

PROCEDURE Quicksort(links, rechts: LONGINT; VAR werte: ARRAY OF LONGINT);
VAR
  i, j, cur, swap: LONGINT;
BEGIN
  i := links;
  j := rechts;
  cur := werte[(links + rechts) DIV 2];
  LOOP
    WHILE werte[i] < cur DO INC(i); END;
    WHILE werte[j] > cur DO DEC(j); END;
    IF i >= j THEN EXIT; END;
    swap := werte[i];
    werte[i] := werte[j];
    werte[j] := swap;
    INC(i); DEC(j);
  END;
  IF i = j THEN
    INC(i); DEC(j);
  END;
  IF links < j THEN
    Quicksort(links, j);
  END;
  IF i < rechts THEN
    Quicksort(i, rechts);
  END;
END;
END Quicksort.

```



Modifiziertes Quicksort für Medianbestimmung



```

VAR median: LONGINT;

PROCEDURE Median*(links, rechts: LONGINT; VAR werte: ARRAY OF LONGINT);
VAR
  i, j, cur, swap: LONGINT;
  ok: BOOLEAN;
  mitte: LONGINT;
BEGIN
  i := links;
  j := rechts;
  cur := werte[(links + rechts) DIV 2];
  LOOP
    WHILE werte[i] < cur DO INC(i); END;
    WHILE werte[j] > cur DO DEC(j); END;
    IF i >= j THEN EXIT; END;
    swap := werte[i];
    werte[i] := werte[j];
    werte[j] := swap;
    INC(i); DEC(j);
  END;
  IF i = j THEN
    INC(i); DEC(j);
  END;
  IF (links < median) & (median < j) & (links < j) THEN
    Median(links, j);
  END;
  IF (i < median) & (median < rechts) & (i < rechts) THEN
    Median(i, rechts);
  END;
END Median;
    
```

Sortiere weiter, wenn Median im linken Teil liegt

Sortiere weiter, wenn Median im rechten Teil liegt

Median!

ist Rekursion überhaupt nötig?

Modifiziertes Quicksort für Medianbestimmung – Besser



```

PROCEDURE MedianII*(links, rechts: LONGINT; VAR werte: ARRAY OF LONGINT);
VAR
  i, j, cur, swap: LONGINT;
  ok: BOOLEAN;
  mitte: LONGINT;
BEGIN
  mitte := (links + rechts) DIV 2; ok := FALSE;
  REPEAT
    i := links;
    j := rechts;
    cur := werte[(links + rechts) DIV 2];
    LOOP
      WHILE werte[i] < cur DO INC(i); END;
      WHILE werte[j] > cur DO DEC(j); END;
      IF i >= j THEN EXIT; END;
      swap := werte[i];
      werte[i] := werte[j];
      werte[j] := swap;
      INC(i); DEC(j);
    END;
    IF i = j THEN
      INC(i); DEC(j);
    END;
    IF mitte <= j THEN
      rechts := j;
    ELSIF i <= mitte THEN
      links := i;
    ELSE
      ok := TRUE;
    END;
  UNTIL ok;
END MedianII;
    
```

statt einer Rekursion nehmen wir REPEAT

(* Quicksort(links, j) *)

(* Quicksort(i, rechts) *)

Besprechung Übungsserie 4 Aufgabe 2



5% Barriere = Median, einfach mehr rechts

Besprechung Übungsserie 4 Aufgabe 3a

```

Datei öffnen;
erstes zeichen lesen;
WHILE zeichen # „#“ DO
  REPEAT
    Out.Char(zeichen); Lies nächstes Zeichen
  UNTIL zeichen = „#“;
  Lies nächstes Zeichen;
  REPEAT
    Out.Char(zeichen); Lies nächstes Zeichen
  UNTIL zeichen = „#“;
  Out.Ln; Lies nächstes Zeichen;
END;
    
```

Besprechung Übungsserie 4 Aufgabe 3b



```

Datei a öffnen; Datei b öffnen;
lies beide Volumen vola und volb;
WHILE ~a.eof & ~b.eof DO
  IF vergleiche(vola, volb) = 0 THEN
    Out.String( vola );
  ELSIF vergleiche(vola, volb) > 0 THEN
    lies nächstes volb;  (* I *)
  ELSIF vergleiche(vola, volb) < 0 THEN
    lies nächstes vola;  (* II *)
  END;
END;

```

mit drei Dateien eigentlich gleich, nur zusätzliche Rekursion bei (* I *) und (* II *)

Vorbereitung Übungsserie 6 Aufgabe 1



Aufgabe 1. Permutationen.
Entwickeln Sie ein rekursives Programm zur Auflistung aller n!
Permutationen von n Elementen.
Beispiel: Elemente 1, 2, 3
Permutationen 1 2 3, 1 3 2, 2 1 3, 2 3 1, 3 1 2, 3 2 1.

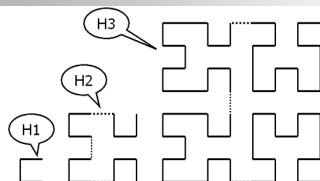
Strategie herausfinden!

```

PROCEDURE Perm(elem: ARRAY OF INTEGER; schritt: INTEGER);
BEGIN
  IF schritt = Anzahl Elemente THEN
    drucke alle übrigen Varianten;
  ELSE
    für alle übrigen Varianten i
      Perm( elem[schritt] := i, schritt + 1);
  END;
END Perm;

```

Vorbereitung Übungsserie 6 Aufgabe 2



```

PROCEDURE hilbert( stufe : INTEGER, richtung : INTEGER );
  IF stufe = 1 THEN
    zeichnen;
    90° nach rechts drehen;
    zeichnen;
    90° nach rechts drehen;
    zeichnen;
  ELSE (* hmmm? *) END;
END hilbert;

```

Vorbereitung Übungsserie 6 Aufgabe 3



Reihen zu dritt, wobei nie zweimal die gleichen hintereinander



maximal einmal
nebeneinander!



Ende einer Gruppe &
Anfang anderer Gruppe sind
nicht nebeneinander

alle erlaubten Permutationen sind gefragt!

also ähnlich wie Aufgabe 1, nur müssen mehr
Regeln beachtet werden

„für alle übrigen Varianten“ muss man also anpassen!

alle bisherigen Nachbarschaften protokollieren!

Vorbereitung Übungsserie 6 Aufgabe 4



Springertouren, linke untere Ecke als Start & Ziel

einfaches Problem, weil
es 33'439'123'484'294 Lösungen hat!

um beim 8x8 alle Möglichkeiten zu testen, benötigen
wir beinahe unendlich lange!

deshalb versucht es zuerst
mit 6x6 Schachbrett – ungemein einfacher!

Lösungsansatz: Backtracking

Heuristiken:

<http://www.borderschess.org/G-KTsimple.htm>
<http://www.ubka.uni-karlsruhe.de/vvv/1992/informatik/11/11.text>
[http://www.inf.hs-zigr.de/~wagenkn/TI/
Komplexitaet/Referate/Graphen/problem2.html](http://www.inf.hs-zigr.de/~wagenkn/TI/Komplexitaet/Referate/Graphen/problem2.html)
