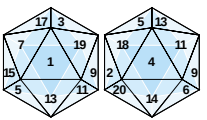
Musteraufgabe:

1 kosaeder

Jedes Käferkind hat im Alter von einem Jahr eine Prüfung zu bestehen. Es muß eine möglichst schnelle Reise über alle Seitenflächen des im Bild dargestellten Ikosaeders machen. Ein Ikosaeder ist ein regelmäßiges Polyeder mit 20 Seitenflächen. Die Seitenflächen tragen die Nummern 1 bis 20:



Die Reise darf nur über jeweils benachbarte Seitenflächen des Ikosaeders erfolgen. Zwei Seitenflächen sind benachbart, wenn sie eine gemeinsame Kante besitzen. Das Käferkind muß jede Seitenfläche genau einmal betreten. Jede Fläche kann Startfläche sein

Auf jeder Seitenfläche muß das Käferkind eine Pause einlegen. Die Länge der Pausen (in Sekunden) errechnet sich durch Multiplikation der Schrittnummer (fortlaufend von 1 bis 20) mit der Nummer der betretenen Seitenfläche. Der Wechsel von einer Seitenfläche zur nächsten dauert eine Sekunde.

Beispiel:
Für eine Reise mit der Route
[12, 2, 18, 5, 15, 7, 17, 10, 8, 20,14, 4, 11, 13, 1, 19, 3, 16, 6, 9]
benötigt das Käferkind 2176 Sekunden.

Aufgabe:
Hilf dem Käferkind und ermittle die günstigsten Startflächen und günstigsten Routen für die schnellsten Reisen.

Lösungsidee

Es überlagern sich zwei Probleme:
1. Ermitteln der zulässigen und
2. Heraussuchen der schnellsten Reisen.

Das 1. wird mit dem Standardverfahren 'backtracking' gelöst: Die 20 Flächen werden nacheinander als Startfläche ausprobiert. Von dieser geht es zu einer benachbarten vorher noch nicht betretenen Fläche, solange wie möglich und bis alle Flächen betreten sind. Zur Lösung des 2. Problems wird Schritt für Schritt die benötigte Zeit ermittelt. Die Suche wird abgebrochen, wenn das bisher gefundene Minimum überschritten wurde (Methode: branch and bound). Eine Lösung wird ausgegeben, wenn alle 20 Flächen betreten wurden.

Programm-Dokumentation

Die Nachbarschaftsbeziehungen werden in der Konstanten *"nachbar"* gespeichert. Das Feld *"frei"* gibt an, welche Flächen bereits betreten wurden. Der aktuelle Weg ist im Feld *"weg"* gespeichert. Die Prozedur *"gehe"* ist rekursiv.
Der Parameter *"feld"* beinhaltet die Ausgangsfläche für den nächsten Schritt. Der Parameter *"tiefe"* nimmt Werte (1..19) an. Die Schrittnummer läuft von 20 bis 1, damit der Abbruch beim branch and bound möglichst schnell erfolgt. Vom Programm wird die Menge der Wege und das Minimum ausgegeben, siehe Ablaufprotokoll. Aus dieser sind die Wege mit der kürzesten Dauer herauszusuchen. Es gibt deren zwei.

Halbformale Programmbeschreibung:

```
Initialisiere Minimum;
initialisiere alle Felder;
Für jedes Feld k der 20 Felder
    beginne den Weg mit k;
    besetze das Feld k;
    Summe := Anfangskosten;
    gehe (Tiefe=1, Feld=k);
    befreie das Feld k;
schreibe das Minimum.

gehe (Tiefe, Feld);
    Für alle 3 Richtungen
      Neufeld := nachbar [Feld];
      wenn frei und bezahlbar
        nimm dieses Feld;
        besetze es;
        summiere die Kosten;
        wenn am Ende
          gib Weg aus;
          halte Minimum fest
        sonst
          gehe (Tiefe+1, Neufeld);
        bei Rückzug
          erstatte die Kosten;
          befreie das Feld.
```

Programm-Ablaufprotokoll

```
01 07 15 05 13 11 04 14 20 08 16 06 09 19 03 17 10 12 02 18  2086
01 07 15 05 13 11 04 18 02 12 10 17 03 16 08 20 14 06 09 19  1994
01 07 15 05 13 11 04 18 02 12 10 17 03 19 09 06 14 20 08 16  1988
01 07 15 05 13 11 04 18 02 12 10 17 03 19 09 06 16 08 20 14  1982
02 12 15 07 01 13 05 18 04 11 09 19 03 17 10 08 16 06 14 20  1994
04 11 13 01 07 15 05 18 02 12 10 17 03 19 09 06 14 20 08 16  1998
04 11 13 01 07 15 05 18 02 12 10 17 03 19 09 06 16 08 20 14  1992
05 13 01 07 15 12 02 18 04 11 09 19 03 16 06 14 20 08 10 17  1998
05 13 01 07 15 12 02 18 04 11 09 19 03 17 10 08 16 06 14 20  1988
07 01 13 05 15 12 02 18 04 11 09 19 03 16 06 14 20 08 10 17  1992
07 01 13 05 15 12 02 18 04 11 09 19 03 17 10 08 16 06 14 20  1982
                                                  Minimum:   1982
```

Programm-Text

```
PROGRAM Ikosaeder;
CONST nachbar:
          ARRAY[1..20, 1..3] OF integer =
        ( (07,13,19), (12,18,20), (16,17,19),
          (11,14,18), (13,15,18), (09,14,16),
          (01,15,17), (10,16,20), (06,11,19),
          (08,12,17), (04,09,13), (02,10,15),
          (01,05,11), (04,06,20), (05,07,12),
          (03,06,08), (03,07,10), (02,04,05),
          (01,03,09), (02,08,14) );
VAR  frei:  ARRAY[1..20] OF boolean;
     weg:   ARRAY[1..20] OF integer;
     k, sum, min : integer;
PROCEDURE gehe (tiefe, feld : integer);
  VAR neufeld, richtung, h : integer;
  BEGIN
    FOR richtung := 1 TO 3 DO BEGIN
      neufeld := nachbar[feld, richtung];
      IF frei[neufeld] AND (sum < min)
      THEN BEGIN
        weg[tiefe+1] := neufeld;
        frei[neufeld] := false;
        sum :=sum+1+(20-tiefe)*neufeld);
        IF tiefe = 19 THEN BEGIN
          FOR h := 1 TO 20 DO
              write (weg[h]:3);
          writeln (sum:10);
          IF sum < min THEN min := sum;
        END ELSE gehe (tiefe+1, neufeld);
        sum :=sum-1-(20-tiefe)*neufeld);
        frei[neufeld] := true
  END END END;
BEGIN
  min := maxint;
  FOR k := 1 TO 20 DO frei[k]:=true;
  FOR k := 1 TO 20 DO BEGIN
    weg[1] := k;
    frei[k] := false;
    sum := 20 * k; gehe (1, k);
    frei[k] := true;
  END; writeln ('Minimum: ', min)
END.
```