Info2 Uebung vom 3.6.02

Gruppe F, Thomas Briner, http://n.ethz.ch/student/brinerth/info2

3. Juni 2002

1 Nachbesprechung U6

1.1 Fibonacci-Zahlen

$$Fib(i) := Fib(i-1) + Fib(i-2)$$

$\Rightarrow$Induktion braucht Verankerung für 2 Werte !

1.2 Minimaler AVL

Idee: Rekursiv aufbauen

2 Dynamische Programmierung

2.1 Rucksackproblem

Ein Wanderer packt seinen Rucksack. Er weiss, dass er im Ganzen nicht mehr als eine bestimmte Limite an Gewicht tragen kann. Er ordnet jedem Gegenstand zusätzlich noch einen bestimmten Wert zu. Welches ist die optimale Bepackung, um mit einem Gewicht <= limit die maximale Summe an Werten mitzutragen?

Gegenstand	1	2	3	4	5	6	7
Gewicht	5	9	4	7	1	2	7
Wert	9	3	8	7	7	5	9

2.1.1 Backtracking Lösung - branch and bound

Grundentscheidung bei jedem Gegenstand: Nehmen oder nicht? Was brauche ich an Wissen um diese Entscheidung zu treffen?

Wie speichere ich die benötigten Informationen?

Rekursive Prozedur auscodiert:

```
PROCEDURE check(a:ARRAY OF BOOLEAN; i,weight,value:INTEGER);
                          IF (i=n) THEN
                                  IF (weight<limit) THEN
                                          IF (value>maxValue) THEN
                                                  maxValue:=value; maxSolution:=a;
                                          END;
                                  END;
                          ELSE
                                  a[i]:=TRUE;
                                  check(a,i+1,weight+w[i],value+v[i]);
                                  a[i]:=FALSE;
                                  check(a,i+1,weight,value);
                          END;
                  END check;
```

mögliche Bounds:
- wenn die Gewichtslimite erreicht ist:

- wenn das bisherige Maximum nicht mehr erreicht werden kann

2.1.2 Dynamische Programmierung

Ist das Optimalitätsprinzip gegeben?

Annahme: Optimale Lösung ist gegeben. Aus dieser optimalen Lösung schaue ich nur einen bestimmten Teil an. Ist diese Teillösung dann auch optimal?

Was ist in diesem Beispiel die Teillösung?

Gegenstand	1	2	3	4	5	6	7
Gewicht	5	9	4	7	1	2	7
Wert	9	3	8	7	7	5	9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1																										
2																										
3																										
4																										
5																										
6																										
7																										

Rekursive Formel:

Vorgehen für Ausfüllen:

2.2 Summe bilden

9 6 9 2 5 0 2 1 3 = 100
An beliebigen Stellen '+' setzen

2.2.1 Branch and bound

An jeder Stelle entscheiden: '+' setzen odern nicht?

Ausprogrammiert:

```
PROCEDURE check(i,sum,aktsum:INTEGER):BOOLEAN;
        IF (i=n) OR (sum+aktsum>limit) THEN
                RETURN (sum+aktsum=limit);
        ELSE
                RETURN (check(i+1,sum+aktsum,a[i]) OR
                        check(i+1,sum,aktsum*10+a[i]);
        END;
END check;
```

2.2.2 Dynamische Programmierung

Induktiver Ansatz: Alle bisher erreichbaren Summen bekannt, dann kommt neue Zahl dazu

Verankerung: keine Zahlen -> einzig mögliche Summe: 0

1. Zahl: nur Summe 6 kann erreicht werden

2. Zahl: entweder '+' setzen -> 6+9=15 oder kein '+' -> 69

3. Zahl: entweder '+' setzen -> 15+2=17 oder 69+2=71 oder kein '+' -> 692
oder 6+92=98

3 Selbstanordnende Listen

Beim Zählen unbedingt darauf achten, dass Zugriff auf 1.Element Kosten 1 ergibt, nicht Kosten 0

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	9	0	0	0	3	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	8	9	0	0	0	17	0	0	0	11	12	0	0	0	20	0	0	0	0	0	0	0
4	0	0	0	0	8	9	0	7	0	17	0	15	16	11	12	0	24	0	20	0	18	19	0	0	0	27
5	0	7	0	0	8	15	16	7	14	17	24	15	22	23	18	19	24	31	20	27	18	25	26	0	0	27
6	0	7	5	12	8	15	16	20	21	17	24	22	29	23	27	28	24	31	29	36	25	32	26	30	31	27
7	0	7	5	12	8	15	16	20	21	17	24	22	29	25	29	30	26	33	31	38	32	36	37	31	40	38