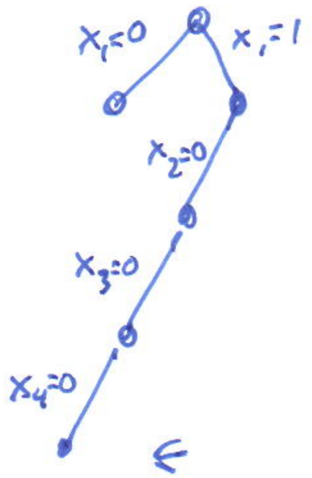
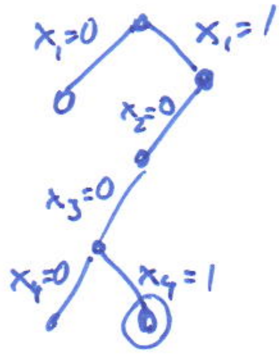


Berechne UB: $x_1 = 1$, $x_5 = 1$, $x_6 = \frac{2}{9}$, $\sum_{i=1}^{7} x_i p_i = 13$

Das ist kleiner als 15, also Sackgasse!

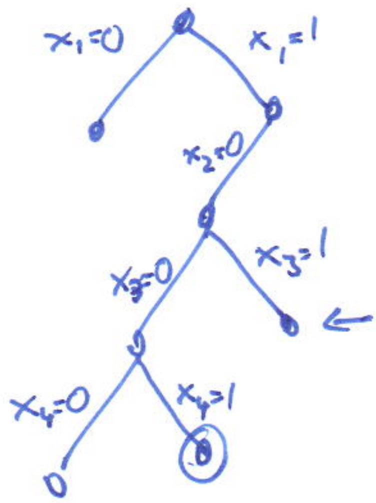


Hier (wie gesehen)
$UB = LB = 15$ für $S = \{1, 4\}$

~~Verwendung von anderen Objekten liefert nur schlechtere Lösungen~~

Außerdem erfordert $x_1 = x_4 = 1$ dass
$Z - \sum x_i z_i = 0$, d.h. $x_5 = \dots = x_7 = 0$.

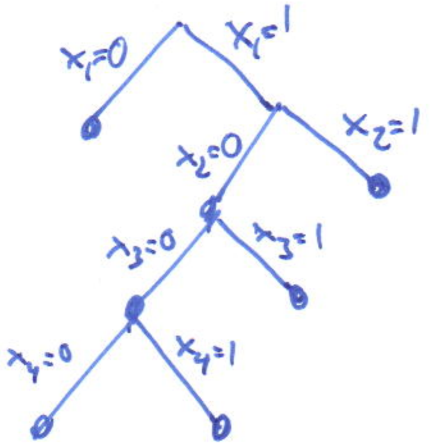
Damit:



$Z - \sum_{i=1}^{3} x_i z_i = 1$ ← zu klein für weitere Objekte!

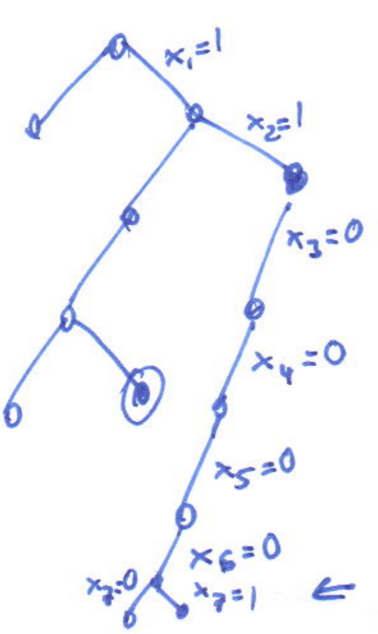
Damit $UB = 14 < 15$!

Damit



$Z - \sum_{i=1}^{2} x_i z_i = 4$,

Dann passt nur noch Objekt 7, also $x_3 = \dots = x_6 = 0$



Damit $UB = 14 < 15$

← $x_1 = x_2 = x_7$

$\boxed{\text{Also Optimum: } 15 \text{ ! für } S = \{1,4\}}$

ALGORITHMUS 1.14 (BRANCH-AND-BOUND)

Übergeben: $z_1, \dots, z_n, Z, p_1, \dots, p_n$ (global: Kostenwerte, Kostenschranke, Nutzenwert)

P (bester bekannter Lösungswert)

ℓ (nächster Index, über den verzweigt wird)

$x_j = b_j$ für $j = 1, \dots, \ell-1$ mit $b_j \in \{0,1\}$ (bislang fixierte Binärvariable)

Ausgabe: $\max \left\{ \sum_{j=1}^{\ell-1} b_j p_j + \sum_{j=\ell}^{n} x_j p_j \;\middle|\; \sum_{j=1}^{\ell-1} b_j z_j + \sum_{j=\ell}^{n} x_j z_j \leq Z, \; x_j \in \{0,1\} \right\}$

(Also: Lösung des Knapsackproblems mit den ersten $\ell-1$ Variablen fixiert)

Branch-and-Bound (ℓ)

(1) IF $\left(\sum_{j=1}^{\ell-1} b_j z_j > Z \right)$ THEN RETURN // Unzulässig

(2) IF $\left(\sum_{j=1}^{\ell-1} b_j z_j > P \right)$ THEN // Lösung verbessert

$P := \sum_{j=1}^{\ell-1} b_j p_j$;

(3) IF $(\ell > n)$ RETURN // Blatt im Enumerationsbaum war erreicht

(4) Compute $U := UB(b_1, \dots, b_{\ell-1})$; // Obere Schranke berechnen

(5) IF $(U > P)$ THEN {

$b_\ell := 1$; Branch-and-Bound $(\ell+1)$;

$b_\ell := 0$; Branch-and-Bound $(\ell+1)$;

}

(6) RETURN

Dabei ist

$$UB(b_1,\dots,b_{\ell-1})$$

eine geeignete obere Schranke:

$$\max\left\{ \sum_{j=1}^{\ell-1} b_j p_j + \sum_{j=\ell}^{n} x_j p_j \;\middle|\; \sum_{j=1}^{\ell-1} b_j z_j + \sum_{j=\ell}^{n} x_j z_j \le Z,\ x_j \in [0,1] \right\}$$

Berechnung mit Greedy-Algorithmus 1.4 !

Satz 1.20

Algorithmus 1.19 (als rekursiv arbeitende Unterroutine) berechnet in endlicher Zeit eine Optimallösung für das Knapsackproblem. Die Worst-Case-Laufzeit beträgt $O(n2^n)$.

Beweis:

Es werden systematisch alle möglichen Teilmengen durchprobiert (und dabei Teilmengen nur dann ausgelassen, wenn sie unzulässig (1) sind oder keine Verbesserung bringen (2)).
Die Zahl der Rekursionsaufrufe in (5) ist insgesamt 2^n, die sonstigen Berechnungen benötigen jeweils $O(n)$.

$\square$