Aufgabe 6.1 - Vitrine

In einem dynamischen Ansatz wählen wir hier zwei Laufvariablen: die Anzahl der möglichen Trophäen, wobei T_i Die Menge der ersten i Trophäen meint und rb ist die verfügbare Restbreite der Vitrine nach platzieren der gewählten Trophäen.
a)
Gesucht ist maxAnsehen(T_n,B)
Wir nutzen eine Art Orakel Ansatz. Wenn wir die Trophäe i nicht ausstellen, dann ist das Restproblem maxAnsehen(T_n $\setminus\{i\}$, B). Falls wir sie ausstellen ist noch maxAnsehen(T_n $\setminus\{i\}$, B-b_i) zu lösen.
b)
Die Rekursionsgleichung sieht für die ersten i Trophäen wie folgt aus:
Falls $rb > b_i$ $maxAnsehen(i, rb) = max\{maxAnsehen(i-1, rb)\ ,\ (maxAnsehen(i-1, rb-b_i) + a_i)\}$
Sonst einfach:
$maxAnsehen(i, rb) = maxAnsehen(i-1, rb)$
wobei Die Basisfälle $maxAnsehen(j, 0) = maxAnsehen(0, b) = 0$,für beliebige j und b.
c) Pseudocode:

```
Array((n+1),(B+1)) // Speicherzellen fuer alle zwischenwerte.
// a(i) und b(i) sind ansehen und breite von Trophae i
for i = 0 to n:
  Array[i][0] = 0
for j = 0 to B:
  Array[0][j] = 0

for i = 1 to n:
  for j = 1 to B:
    if j < b(i) :
      Array[i][j] = Array[i-1][j]
    else:
      Array[i][j] = max ( Array[i-1][j] , (a(i)+Array[i-1][j-b(i)])
return A[n][B]
```

Laufzeit:
Im Algortihmus wird jede Zelle genau einmal berechnet, damit haben wir (n+1)(B+1) $= O(nB)$ Operationen.

6.2 - Würfeln

Wenn wir die Zahl Z mit n Würfeln mit je m Seiten würfeln möchten, ergibt sich für die Anzahl der Möglichkeiten eine Berechnungsmöglichkeit mit n-1 Würfeln. Hierbei Summiere ich alle Möglichkeiten die Zahlen Z-1, Z-2,...,Z-m mit n-1 Würfeln zu bilden, da alle diese Möglichkeiten mit dem zusätzlichen Würfel Z erreichen können. Falls Wir 0 Würfel haben, ist die Anzahl stets 0. Falls Z$<$ n gibt es auch keine Möglichkeit diese Zahl zu erreichen ebenso für Z $> m \cdot n$. Damit bleiben noch 2 relevante Fälle. Falls Z $=$ n, dann muss jeder Würfel die 1 zeigen, es gibt also genau eine Möglichkeit. Der letzte Fall ist die Rekursion:
$Anzahl_m(Z, n)$ für $n \leq Z \leq n \cdot m \quad = \sum_k Anzahl(Z-k, n-1), \quad$ wobei $Z - k \geq 0$.
Die letztgenannte Bedingung ist mit summe von 1 bis min(Z,m) erfüllt.

b)
Wie oben beschrieben realisieren wir diesen Algorithmus wieder über eine Matrix ähnlich der 6.1. Dabei zählen die Zeilen die Würfel hoch und die Spalten die Zahl Z. Wir erhalten für ein spezifisches M jeweils eine spezifische Tabelle.

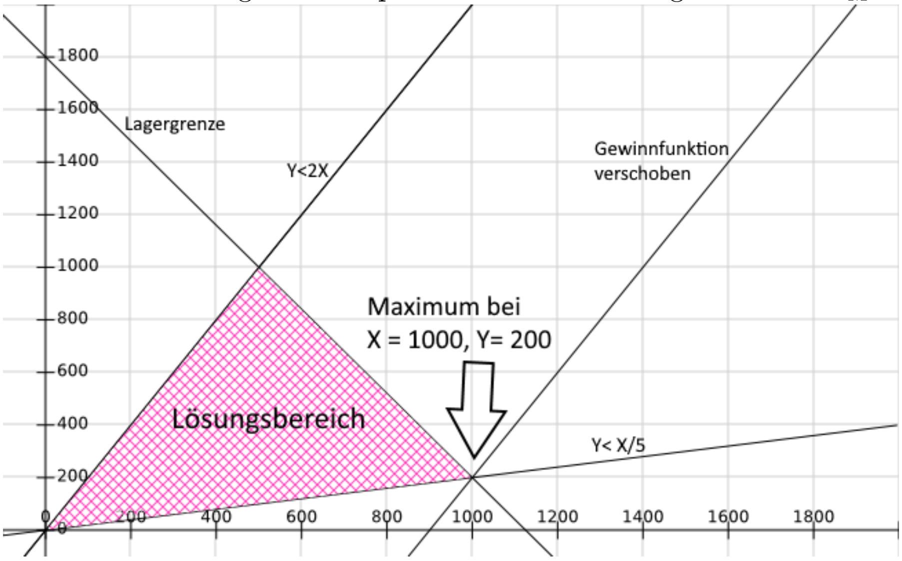
```
// gegeben sind m, Z, n
vlaues = Array[n][Z] // a m-specific Table of Z*n cells
values.fill(0)

for i = 1 to n:{
  for j = 1 to Z:{
    if i <= Z and j <= m*i:
      for k=1 to (min(Z,m)):
        values[i][j] += values[i-k][j-1]
    else:{
      if i == j:
        values[i][j] = 1
        }
    }
    return values[n][Z]
}
```

Auch hier speichern wir quasi in Zellen, der Speicherbedarf ist also $O(n \cdot Z)$. Für jede Zelle haben wir einen Zugriff beim initialisieren und maximal m Berechnungen in der for k schleife. Also pro Zelle m Berechnungen und somit $O(m \cdot n \cdot Z)$

6.3 - Kaffee-Optimierung

Um eine Gewinnfunktion zu bauen, wählen wir X als die Menge von Tassen von Variante A und Y als Tassen von B. Dann ist die Gewichtsmenge von K und M in Gramm mit $m_K = 4X + 10Y, m_M = 12X$ gefordert. Zusätzlich gilt $m_M \leq 2m_K$ und $m_K \leq 2m_M$ sowie $m_M + m_K \leq 18000$. Einsetzen und umformen dieser Beziehungen ergibt $X \leq 5Y$ und $Y \leq 2X$ sowie $16X + 10Y \leq 18000$. Gewinn einer Tasse A = 210 - (0.012*500 + 0.004*1000) = 200 und einer Tasse B = 110 - (0.01*1000) = 100 (beides in Cent). Somit lässt sich mit der Gewinnfunktion 2X+ Y in € folgende Graphik darstellen. Es ergeben sich $m_M = 12000; m_K = 6000$



6.4 - Lager-Optimierung

Um bei der Optimierung die Kontrolle über die Kosten zu behalten führen wir zwei Wahrheitsvariablen ein: Die Variable L_i ist 0, wenn das Lager nicht gebaut wird, und ist 1, wenn das Lager gebaut wird. Die zweite Variable hierfür ist $V_{i,j}$, die 1 ist , wenn das Lager i das Geschäft j versorgt, ansonsten 0. Damit können wir in der Berechnung der Kosten die spezifischen Kosten für Eröffnung und Versorgung bestimmen.
Die zu minimierenden Kosten sind dann:
$\sum_{i=1}^{n} L_i c_i + \sum_{i=1}^{n} \sum_{j=1}^{m} V_{i,j} d_{i,j}$, mit den Einschränkungen: $L_i, V_{i,j} \in \{0,1\}$
Und den weiteren Bedingungen:
(1) $\sum_{j=1}^{m} V_{i,j} = 1$, damit jedes Geschäft von mindestens einem Lager versorgt wird.
(2)$L_i \geq V_{i,j}$, damit das Geschäft nur von Lager i versorgt wird, wenn es auch offen ist.
Anmerkung: Laut Aufgabe steht bei (1) ein größer-gleich, das ist aber nicht minimiert, da mehrere Lager zu einem Geschäft laut Aufgabe unnötig wären.

6.5 - Turing-Maschinen

a)
Idee: in q0:Prüfe ob das letzte Bit, wenn ja, dann bleibe mit qa und akzeptiere.
Wenn nein, dann lösche dieses und fange beim nächsten wieder an.
$Q = \{q_0, q_1, q_2, q_a\}$, $\Sigma = \{0,1\}$, δ_a , q_0 , $\Gamma = \{0,1,B\}$, $F = \{q_a\}$
$\delta_a(q_0, i) = (q_1, i, rechts), \quad i \in \{0,1\}$ (letztes bit?)
$\delta_a(q_1, i) = (q_2, i, links), \quad i \in \{0,1\}$ (nicht letztes bit!)
$\delta_a(q_1, B) = (q_a, B, bleib)$ (ist letztes bit)
$\delta_a(q_2, i) = (q_0, B, rechts), \quad i \in \{0,1\}$ (lösche vorgänger starte neu)
Die fehlenden Definitionen dienen für ungültige Eingaben oder Akzeptanz.
b)
Idee: in q0: finde erste 1, dann q1, dann inklusive erster 1 insgesamt 11 oder 12 bits, sonst ungültig.
$Q = \{q_0, q_1, q_2, ..., q_{11}, q_{12}, q_{13}, q_a\}$, $\Sigma = \{0,1\}$, δ_b , q_0 , $\Gamma = \{0,1,B\}$, $F = \{q_a\}$
$\delta_b(q_0, 0) = (q_0, 0, rechts),$ (führende nullen)
$\delta_b(q_0, 1) = (q_1, 1, rechts),$ (erstes bit)
$\delta_b(q_k, i) = (q_{k+1}, i, rechts), \quad i \in \{0,1\}, \quad k \in [1,12]$ (hauptsache bits)
$\delta_b(q_{11}, B) = (q_a, B, bleib)$ (>=1024 , <4096)
$\delta_b(q_{12}, B) = (q_a, B, bleib)$ (>=1024 , <4096)
$\delta_b(q_{13}, i) = undefined$ (13 bits: >4095)
Die fehlenden Definitionen dienen für ungültige Eingaben oder Akzeptanz.
c)
Idee: in q0: laufe bis Ende, füge 0 hinten an(x2) dann -1.
Das minus 1 sorgt dafür, dass die letzten nullen zur 1 werden und erst bei einer 1 der carry genullt wird. $Q = \{q_0, q_1, q_2, ..., q_{11}, q_{12}, q_{13}, q_a\}$, $\Sigma = \{0,1\}$, δ_c , q_0 , $\Gamma = \{0,1,B\}$, $F = \{q_a\}$
$\delta_c(q_0, 0) = (q_0, 0, rechts),$ (führende nullen)
$\delta_c(q_0, 1) = (q_1, 1, rechts),$ (Zahl > 0)
$\delta_c(q_1, i) = (q_1, i, rechts), \quad i \in \{0,1\}$ (bis Ende)
$\delta_c(q_1, B) = (q_2, 1, links)$ (rückwärts mit carry)
$\delta_c(q_2, 0) = (q_2, 1, links)$ (rückwärts mit carry)
$\delta_c(q_2, 1) = (q_a, 0, bleibe)$ (carry genullt, fertig)
Die fehlenden Definitionen dienen für ungültige Eingaben oder Akzeptanz.