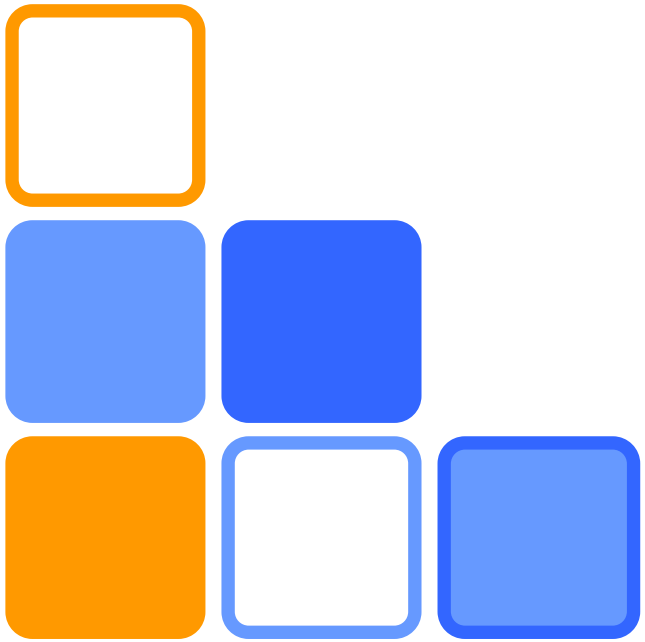
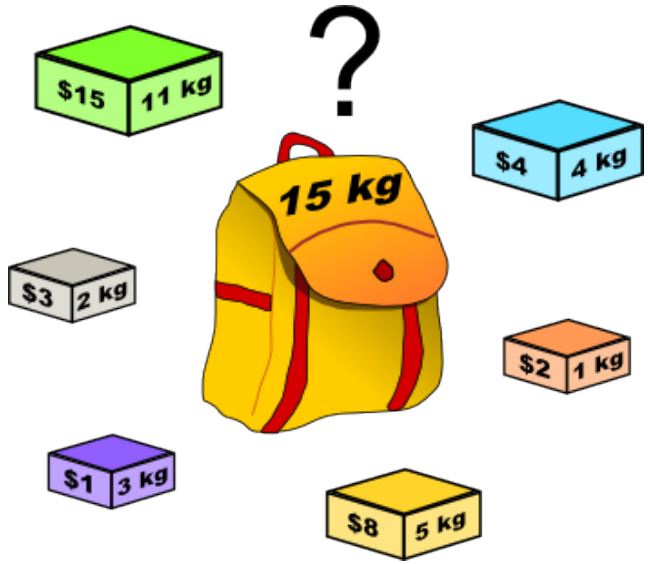
ナップサック問題の
近似解を求めるモデル

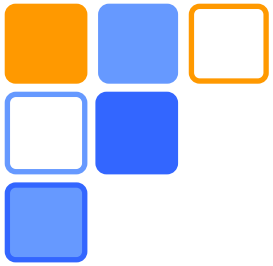


ナップサック問題

- N個のアイテム(各アイテムで重さと価格がばらばら)と容量Cのナップサックが与えられCる。
- 容量Cを超えない範囲でナップサックにアイテムを詰め、ナップサックに入れたアイテムの価格を最大にするにはどのアイテムを選べば良いか。



http://dai1741.github.io/maximum-algo-2012/docs/dynamic-programming/



ナップサック問題の問題点

- NP困難
  - ナップサック問題は、選ぶアイテムの組み合わせをすべて確かめれば最適解を算出可能
    - アイテムの数が増えれば組み合わせの数は膨大な量
    - 最適解を求めるのは困難

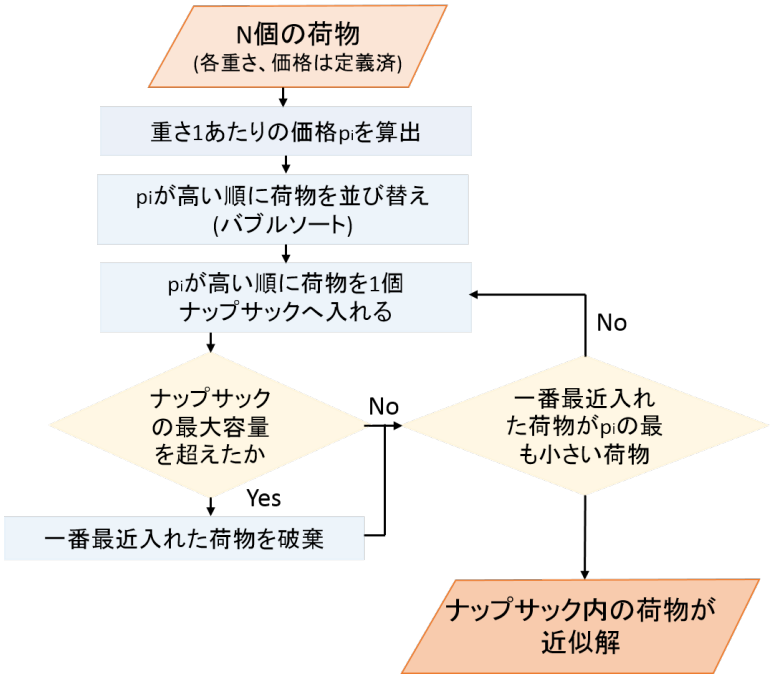
⇒ 近似アルゴリズムでナップサック問題の近似解を求める

作成したモデル

- ナップサック問題を解く近似アルゴリズム
  - 重さ1(kg)あたりの価格を算出し、大きい順にソート
  - ソート後の1番目(1番大きいもの)から順にナップサックに入れていく
  - ナップサックの最大容量を超えたらそれまでに入ったアイテムが問題の近似解

初期状態

順番	アイテム	価格($)	重さ(kg)
1	アイテム1	15	11
2	アイテム2	3	2
3	アイテム3	1	3
4	アイテム4	8	5
5	アイテム5	2	1
6	アイテム6	4	4



モデルの動作1

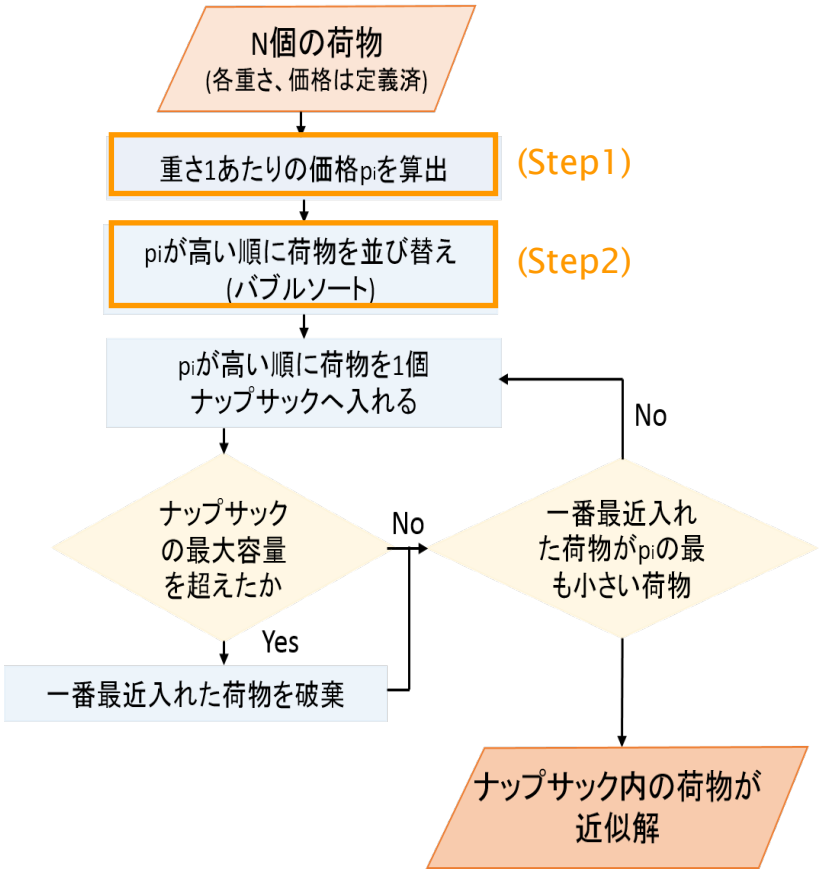
初期状態

(Step1) 価格/重さの算出

順番	アイテム	価格($)	重さ(kg)	価格/重さ
1	アイテム1	15	11	1.36
2	アイテム2	3	2	1.50
3	アイテム3	1	3	0.33
4	アイテム4	8	5	1.60
5	アイテム5	2	1	2.00
6	アイテム6	4	4	1.00

(Step2) 価格/重さの大きい順にソート
(バブルソート利用)

順番	アイテム	価格($)	重さ(kg)	価格/重さ
1	アイテム5	2	1	2.00
2	アイテム4	8	5	1.60
3	アイテム2	3	2	1.50
4	アイテム1	15	11	1.36
5	アイテム6	4	4	1.00
6	アイテム3	1	3	0.33



モデルの動作2

最大容量(15kg)

ナップサック

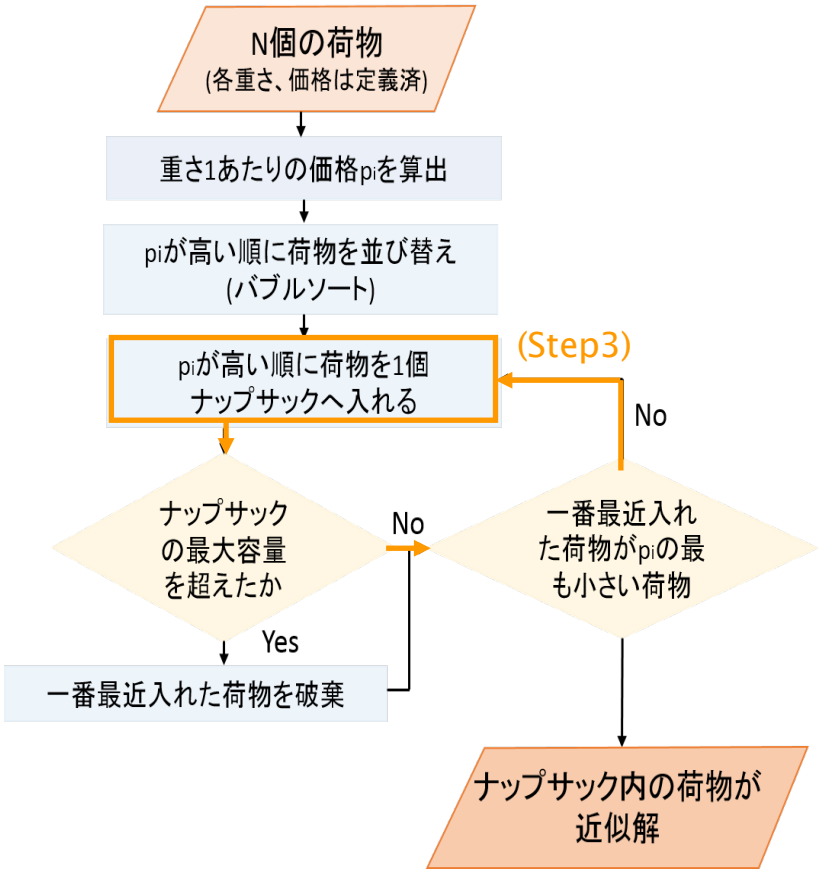
- アイテム5

価格:2($)
容量:1(kg)

容量＜最大容量 の間
(Step3)を繰り返す

(Step3) 1番目のアイテムから順に
ナップサックに入れる

順番	アイテム	価格($)	重さ(kg)	価格/重さ
1	アイテム5	2	1	2.00
2	アイテム4	8	5	1.60
3	アイテム2	3	2	1.50
4	アイテム1	15	11	1.36
5	アイテム6	4	4	1.00
6	アイテム3	1	3	0.33



モデルの動作3

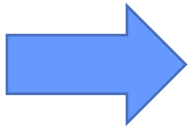
最大容量(15kg)

ナップサック

- アイテム5
- アイテム4
- アイテム2
- アイテム1

価格:28($)
容量:19(kg)

アイテム1を入れると
最大容量を超えた



ナップサック

- アイテム5
- アイテム4
- アイテム2
- ~~アイテム1~~

価格:13($)
容量:8(kg)

(Step3) 1番目のアイテムから順に
ナップサックに入れる

順番	アイテム	価格($)	重さ(kg)	価格/重さ
1	アイテム5	2	1	2.00
2	アイテム4	8	5	1.60
3	アイテム2	3	2	1.50
4	アイテム1	15	11	1.36
5	アイテム6	4	4	1.00
6	アイテム3	1	3	0.33

- アイテム1を破棄
- (Step3)に戻り、アイテム6を
ナップサックへいれる

モデルの動作4

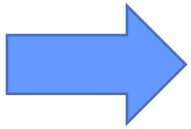
最大容量(15kg)

ナップサック

- アイテム5
- アイテム4
- アイテム2
- アイテム6
- アイテム3

価格:18($)
容量:15(kg)

アイテム3を入れたとき
容量＝最大容量



ナップサック

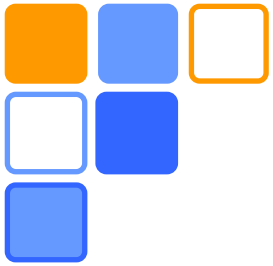
- アイテム5
- アイテム4
- アイテム2
- アイテム6
- アイテム3

価格:18($)
容量:15(kg)

(Step3) 1番目のアイテムから順に
ナップサックに入れる

順番	アイテム	価格($)	重さ(kg)	価格/重さ
1	アイテム5	2	1	2.00
2	アイテム4	8	5	1.60
3	アイテム2	3	2	1.50
4	アイテム1	15	11	1.36
5	アイテム6	4	4	1.00
6	アイテム3	1	3	0.33

今ナップサックに入っている
アイテムが得られた近似解



まとめ

- ナップサック問題の近似解を求めるプロダクションシステムを作成

- *plusや*divideといった関数の存在をモデル作成後に気づいてしまった
  - 今回のモデルでは各アイテムの重さと価格を15以下で設定しなければならない
  - ナップサックの最大容量も15以下で制限
  - *plusや*divideを使うと重さや価格の制限をつけずに動くモデルを作成できたと思う